2008

MULTIPLE CHANNEL COHERENT AMPLITUDE MODULATED (AM) TIME DIVISION MULTIPLEXING (TDM) SOFTWARE DEFINED RADIO (SDR) RECEIVER

Veerendra Bhargav Alluri
University of Kentucky, vballu2@uky.edu

ABSTRACT OF THESIS

MULTIPLE CHANNEL COHERENT AMPLITUDE MODULATED (AM) TIME DIVISION MULTIPLEXING (TDM) SOFTWARE DEFINED RADIO (SDR) RECEIVER

It is often required in communication and navigation systems to be able to receive signals from multiple stations simultaneously. A common practice to do this is to use multiple hardware resources; a different set of resources for each station. In this thesis, a Coherent Amplitude Modulated (AM) receiver system was developed based on Software Defined Radio (SDR) technology enabling reception of multiple signals using hardware resources needed only for one station. The receiver system architecture employs Time Division Multiplexing (TDM) to share the single hardware resource among multiple streams of data. The architecture is designed so that it can be minimally modified to support any number of stations. The Verilog Hardware Description Language (HDL) was used to capture the receiver system architecture and design. The design and architecture are initially validated using HDL post-synthesis and post-implementation simulation. In addition, the receiver system architecture and design were implemented to a Xilinx Field Programmable Gate Array (FPGA) technology prototyping board for experimental testing and final validation.

KEYWORDS: Software Defined Radio, Time Division Multiplexing Architecture, Multi-Channel Coherent AM Receiver, Hardware Descriptive Language Simulation Testing, FPGA Experimental Prototype.

Veerendra Bhargav Alluri

1/4/2008

MULTIPLE CHANNEL COHERENT AMPLITUDE MODULATED (AM) TIME DIVISION MULTIPLEXING (TDM) SOFTWARE DEFINED RADIO (SDR) RECEIVER

By

Veerendra Bhargav Alluri

Dr. J. Robert Heath
(Director of Thesis)

Dr.YuMing Zhang
(Director of Graduate Studies)

1/4/2008

THESIS

Veerendra Bhargav Alluri

The Graduate School
University of Kentucky
2007

MULTIPLE CHANNEL COHERENT AMPLITUDE MODULATED (AM) TIME DIVISION MULTIPLEXING (TDM) SOFTWARE DEFINED RADIO (SDR) RECEIVER

THESIS

A thesis submitted in partial fulfillment of the

requirements for the degree of Master of Science in Electrical Engineering in the

College of Engineering at the University of Kentucky

By

Veerendra Bhargav Alluri

Lexington, Kentucky

Director: Dr. J. Robert Heath, Associate Professor of Electrical and Computer Engineering

Lexington, Kentucky

2007

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my thesis chair, Dr. J.Robert Heath. Without his constant support, encouragement and suggestions this work would not be possible. I would also like to thank Dr. Michael Lhamon from Lexmark Inc. for his guidance and comments. I would also like to extend my thanks to Dr. William R. Dieter and Dr. Laurence G. Hassebrook for serving on my thesis committee and providing their invaluable comments and suggestions for improving the thesis.

I also extend my gratitude to my family and friends for their constant encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FILES

Thesis_Veerendra_Alluri.pdf (File Size: 1825 KB)

Chapter 1

Introduction and Background

Wireless communication protocols and standards are changing rapidly. Making a device, based on old standards, work with the new standards, requires costly hardware upgrades. Sometimes, devices based on old standards, might be completely rendered useless, with the arrival of a new standard. Furthermore, different geographic regions use different standards, making it difficult for regional and global roaming. To address the above problems J. Mitola proposed the concept of Software Define Radio (SDR) [16]. In a SDR most radio receiver functions would be defined in software, to be run on a general purpose programmable processor rather than the functions being implemented strictly in non programmable hardware. The functionality of a SDR receiver can be changed via reprogramming.

The subject of this thesis is the development, testing and validation of a Multiple Channel Coherent Amplitude Modulated (AM) Time Division Multiplexing (TDM) radio receiver architecture based on SDR technology. Several SDR Architectures have been proposed and studied in [17-20 and 25-53]. A multiple channel SDR implementation can be developed by placing several single channel implementations in parallel. This however would be a waste of digital logic resources. A much more hardware efficient implementation approach would be to time division multiplexing a single set of hardware resources across the data streams of several different channels.

In [21], a Time Division Multiplexing Multistage Transmultiplexing downconverter architecture was proposed for implementation of a multi-channel Radio Frequency Identifier (RFID) reader/writer. Unlike the multiple channel coherent AM TDM SDR receiver architecture of this thesis, in the Multistage Transmultiplexing downconverter architecture, as the number of channels increase, additional filtering components must be added prior to the Transmultiplexing stage, increasing the digital hardware gate count significantly. Furthermore, as the number of channels increase, the input sample rate must also be increased. Reference [21] was the only TDM architecture found for a multiple channel radio.

The focus of this thesis is the development, simulation and experimental hardware prototype testing and validation of a Multi-Channel SDR receiver architecture which is very efficient in terms of hardware resource utilization. A goal is that the architecture would be upgradeable to support any number of channels with minimal design alteration and increase in hardware required for implementation. TDM of hardware functional units is to be used within the receiver architecture to minimize hardware; and to accommodate design alterations, the multi-channel SDR receiver architecture will be implemented to Programmable Logic Design (PLD) technology. Evolving faster and cheaper PLD technologies enable the use of TDM. To the authors knowledge, no other multiple channel coherent AM TDM radio receiver architecture based on SDR technology implemented to PLD technology appear in the literature. Other radio architectures based on SDR techonology have been implemented to PLD technology as reported in [17, 21, 26, 27, 30, 36, 41, 42, 43, 47, 48].

Chapter 2

Overview of Amplitude Modulation and Demodulation

2.1 Amplitude Modulation

Modulation is a process in which some property of the carrier is modified relative to the modulating signal [1]. This is also called frequency translation since it effectively translates the modulating signal from one frequency range to the other. Such frequency translations have many advantages in addition to allowing frequency multiplexing [2]. In amplitude modulation the amplitude of the carrier is varied linearly with the baseband signal. The mathematical representation of an amplitude modulated signal, $AM(t)$, is given by Equation 2.1.

$$AM(\mathrm{t}) = \mathrm{A_c}[1 + m(t)]\cos(W_c t) \tag{2.1}$$

The term '$A_c \cos(W_c t)$' in the above Equation is the carrier signal and '$m(t)$' the message signal. From the above Equation it can seen that, in AM, in addition to the product signal '$A_c \cos(W_c t) * m(t)$', the carrier signal itself is being transmitted. This makes the demodulation of AM signals at the receiver end very easy. Figures 2.1(a), 2.1(b) and 2.1(c) show a simple modulating signal, carrier and corresponding amplitude modulated signal respectively.

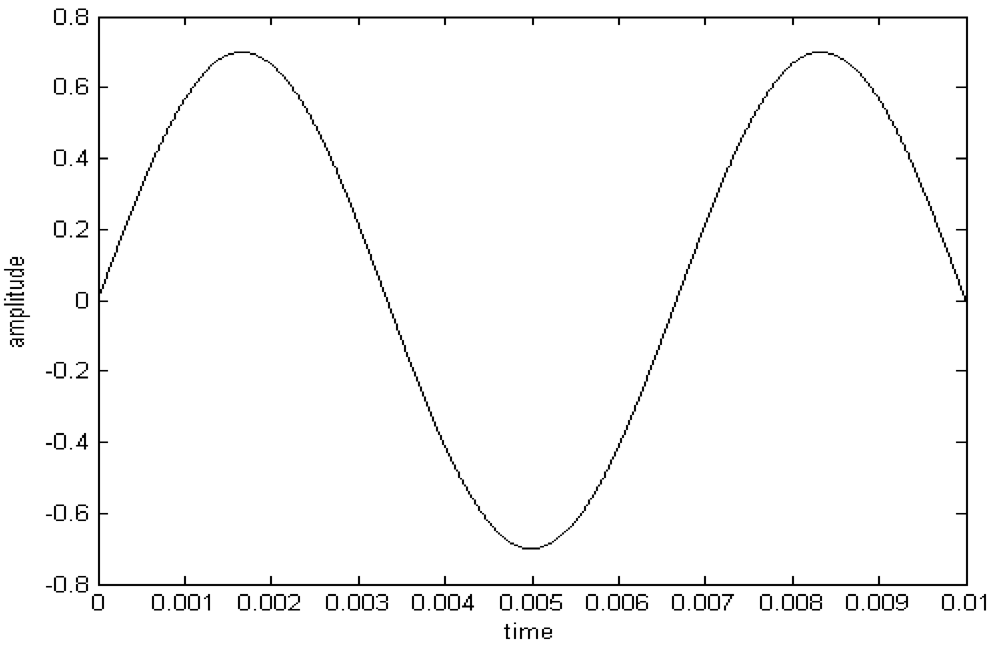


Figure 2.1(a) Modulating Signal

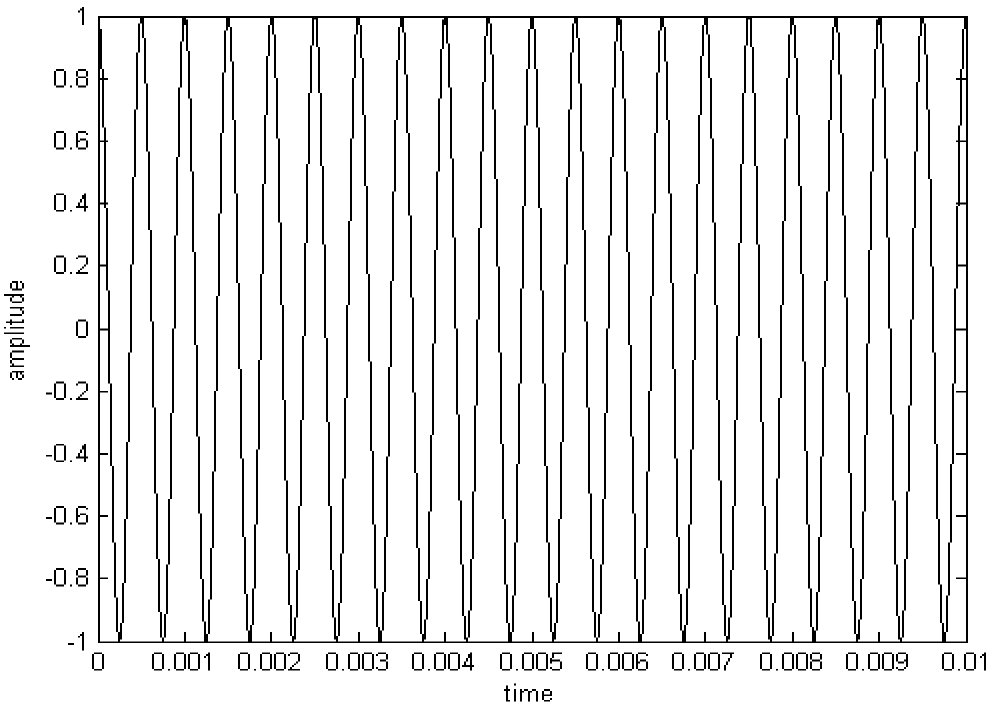


Figure 2.1(b) Carrier Signal

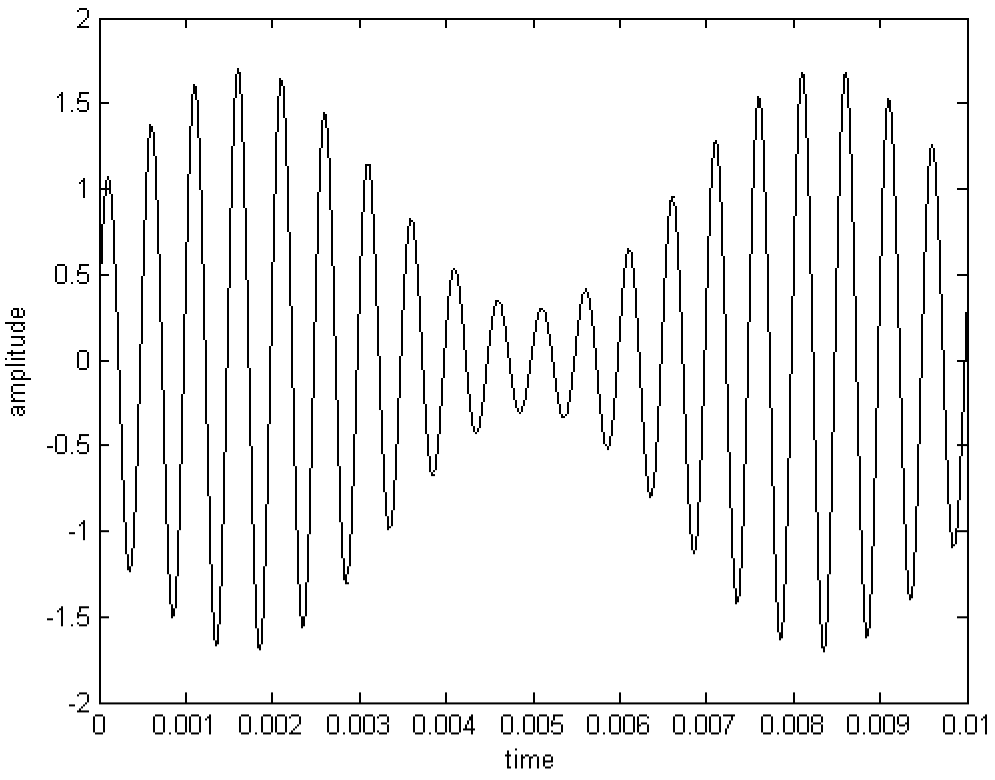


Figure 2.1(c) Amplitude Modulated Signal.

2.2 AM Demodulation

Demodulation is the opposite process of modulation in which the modulating signal is extracted from the modulated carrier. Several demodulators such as diode detector or product demodulator can be used to demodulate AM waves, but, prior to demodulation, the desired carrier frequency must be separated from the spectrum. A straightforward way to do this would be to construct an extremely sharp variable Band-pass filter. Constructing a filter with such sharp selectivity at radio frequencies is both difficult and costly. An efficient alternative is to use the Superheterodyne principle.

2.3 Superheterodyne Principle

The advantage of a Superheterodyne receiver is that most of the receiver components need be sensitive only to a narrow band of frequencies [3]. A functional block diagram of a Superheterodyne receiver is shown in Figure 2.2. In Figure 2.2, only the components preceding the RF amplifier need to be sensitive to a wider band of frequencies. The carrier frequency of interest is converted to an Intermediate Frequency (IF) signal which is then demodulated to obtain the message signal.

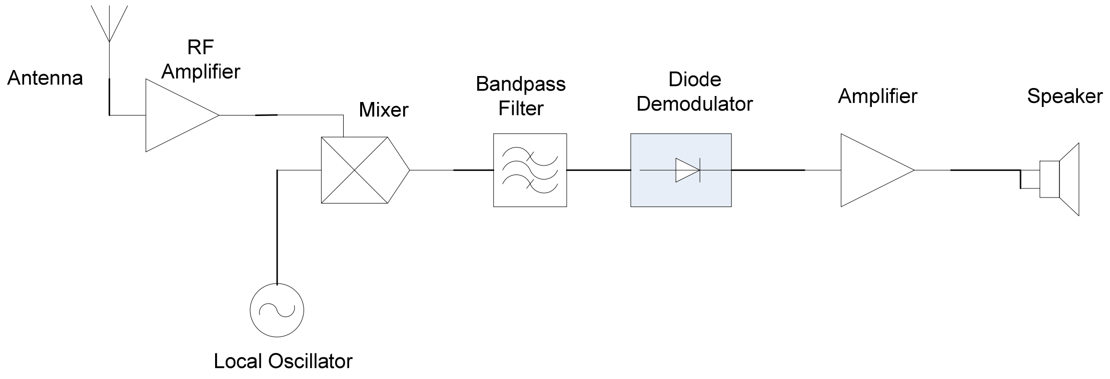


Figure 2.2 Block diagram of Superheterodyne Receiver.

The Radio Frequency (RF) amplifier only allows the frequency band of interest and rejects all other bands. A station is selected by varying the local oscillator frequency. The local oscillator is tuned to the station frequency of interest plus the intermediate frequency. The mixer multiplies the RF amplifier signal and the local oscillator. The Mixer output contains both sum and difference frequency components.

$$
\begin{aligned}
& A_c(1+m(t))\cos(W_c t)*\cos((W_c+W_{if})t) \\
&= A_c(1+m(t))[\cos(W_c+W_{if}+W_c)t+\cos(W_c+W_{if}-W_c)t]*\frac{1}{2} \\
&= \underbrace{[A_c(1+m(t))\cos(W_{if}t)]*\frac{1}{2}}_{\text{Intermediate Frequency Component}}+[A_c(1+m(t))\cos((2W_c+W_{if})t)]*\frac{1}{2}
\end{aligned}
\tag{2.2}
$$

The Intermediate Frequency Component shown in Equation 2.2 is selected by using a band-pass filter centered at the intermediate frequency and all other components are filtered out. This intermediate frequency component is then demodulated to obtain the message signal.

The problem with the above receiver is that it can also receive an image frequency separated from the desired station frequency by twice the intermediate frequency. For example, if its needed to tune into a station at 590 KHz and the IF is set to 455 KHz, the local oscillator should be tuned to 1045 KHz since 1045 – 590 = 455 KHz. But now the receiver can also tune into a station at 1500 KHz because 1500 – 1045 = 455 KHz. To avoid this problem, the image frequency should be filtered out before the mixer using a band-pass filter centered at the frequency of interest. The frequency response of this filter need not be very sharp, but should be sharp enough to filter out the image frequency. Implementing such a filter digitally requires us to store the coefficients for all the frequencies that we need to support [4]. This may be costly if there are a large number of frequencies to be supported.

2.4 Coherent Demodulation

The problem of image frequency can be circumvented by directly downconverting the carrier signal of interest to baseband instead of the intermediate frequency. This can be done by tuning the local oscillator to the carrier frequency of interest and using a Low-Pass Filter (LPF) at the output of the mixer in Figure 2.2 instead of a band-pass filter. The cutoff frequency of the low-pass filter should equal the maximum bandwidth of the modulating signal. Phase difference between the local oscillator signal and the carrier signal plays an important role when downcoverting directly to baseband. Let "Φ" be the phase difference between the local oscillator signal and the station carrier signal, then the product signal at the output of the mixer is given as

$$\begin{aligned} &A_c(1+m(t))\cos(W_c t)\cos((W_c t+\Phi)) \\ &= \frac{1}{2} * A_c(1+m(t))[\cos(\Phi)+\cos(\ 2W_c t+\Phi \ )] \end{aligned} \qquad (2.3)$$

The output of the low-pass filter which filters out all the high frequency components is given by Equation 2.4.

$$\frac{1}{2}A_c\left(\ 1+m(t)\ \right)\cos(\Phi) \qquad (2.4)$$

Equation 2.4 shows how Φ effects the demodulated signal amplitude. For example if Φ = 90, the demodulated signal will be Zero. To avoid this, the local oscillator signal and the carrier signal must always be phase synchronized. Synchronization can be achieved without a Phase Locked Loop (PLL) by using a coherent receiver. A coherent receiver utilizes In-Phase (**I**) and Quadrature-Phase (**Q**) streams to achieve phase synchronization. The block diagram of a coherent receiver is shown in Figure 2.3.

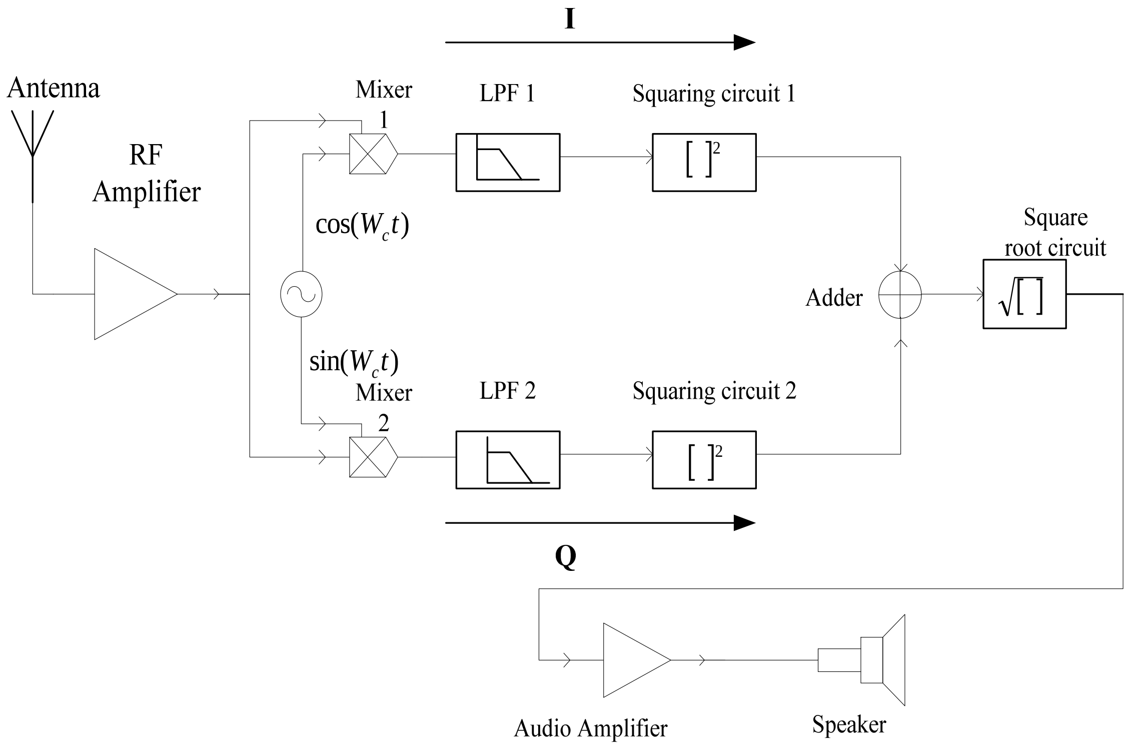


Figure 2.3 Block Diagram of Coherent Receiver

Again let “Φ” be the phase difference between the local oscillator signal and the station carrier signal. The outputs of LPF 1 and LPF 2 are given by Equations 2.5 and 2.6 respectively. It should be noted that both LPF 1 and LPF 2 are identical.

$$\frac{1}{2}A_c\left(\ 1+m(t)\ \right)\cos(\Phi) \tag{2.5}$$

$$\frac{1}{2}A_c\left(\ 1+m(t)\ \right)\sin(\Phi) \tag{2.6}$$

Squaring circuits produce at their output the square of the input signal. The output of the adder, which adds outputs of both the squaring circuits, is given by

$$\left[\frac{1}{2}A_c\left(\ 1+m(t)\ \right)\cos(\Phi)\right]^2+\left[\frac{1}{2}A_c\left(\ 1+m(t)\ \right)\sin(\Phi)\right]^2$$
$$=\left[\frac{1}{2}A_c\left(\ 1+m(t)\ \right)\right]^2\left\{\sin^2(\Phi)+\cos^2(\Phi)\right\}$$

$$=\left[\frac{1}{2}A_c\left(\ 1+m(t)\ \right)\right]^2 \tag{2.7}$$

It can be observed that the output of the adder, Equation 2.7, is free of Φ. This implies that the phase difference between the local oscillator signal and the station carrier signal will have no effect on the output of the coherent receiver circuit. The square root circuit computes the square root of the input signal and produces it at the output. The output of the square root circuit is given by Equation 2.8.

$$\frac{1}{2}A_c\left(\ 1+m(t)\ \right) \tag{2.8}$$

The DC component $\frac{1}{2}A_c$ in Equation 2.8 can easily be removed using a simple mean remover block. The resulting demodulated signal can be suitably amplified and fed to a speaker.

Chapter 3

Coherent AM Multiple Channel TDM SDR Architecture

As shown in Figure 2.3 a single station coherent receiver needs to process two streams of data, an In-phase stream (I) and a Quadrature stream (Q). Thus a receiver which can tune into two stations simultaneously will need to process 4 streams of data. One straightforward way to implement such a receiver would be to use dedicated hardware to process each data stream. Instead of duplicating hardware to process each stream, TDM is used to develop an architecture which will take the hardware needed to process one data stream and use it on a time shared basis to process multiple streams. On every clock edge the time shared hardware will process a different data stream. The tradeoff involved in such an implementation is one of logic resources versus the frequency of operation. For example, if the logic was to be shared among two streams of data, each with sample rate $f_{s,}$ then it should be operated at twice the sample rate i.e., 2 f_s.

In this work the set of hardware components which carry information relating to a particular stream of data is called a channel. As there are four data steams, two for each station, there will be 4 channels, one for each data stream. The channels are assigned numbers so they may be easily identified. Channels 0 and 2 correspond to in-phase (I) streams of stations 1 and 2 respectively. Similarly channels 1 and 3 correspond to quadrature-phase (Q) streams of channels 1 and 2 respectively.

Figure 3.1 shows the top level view of the multiple channel TDM architecture. The functionality of each functional unit shown in the Figure 3.1 will be described now.

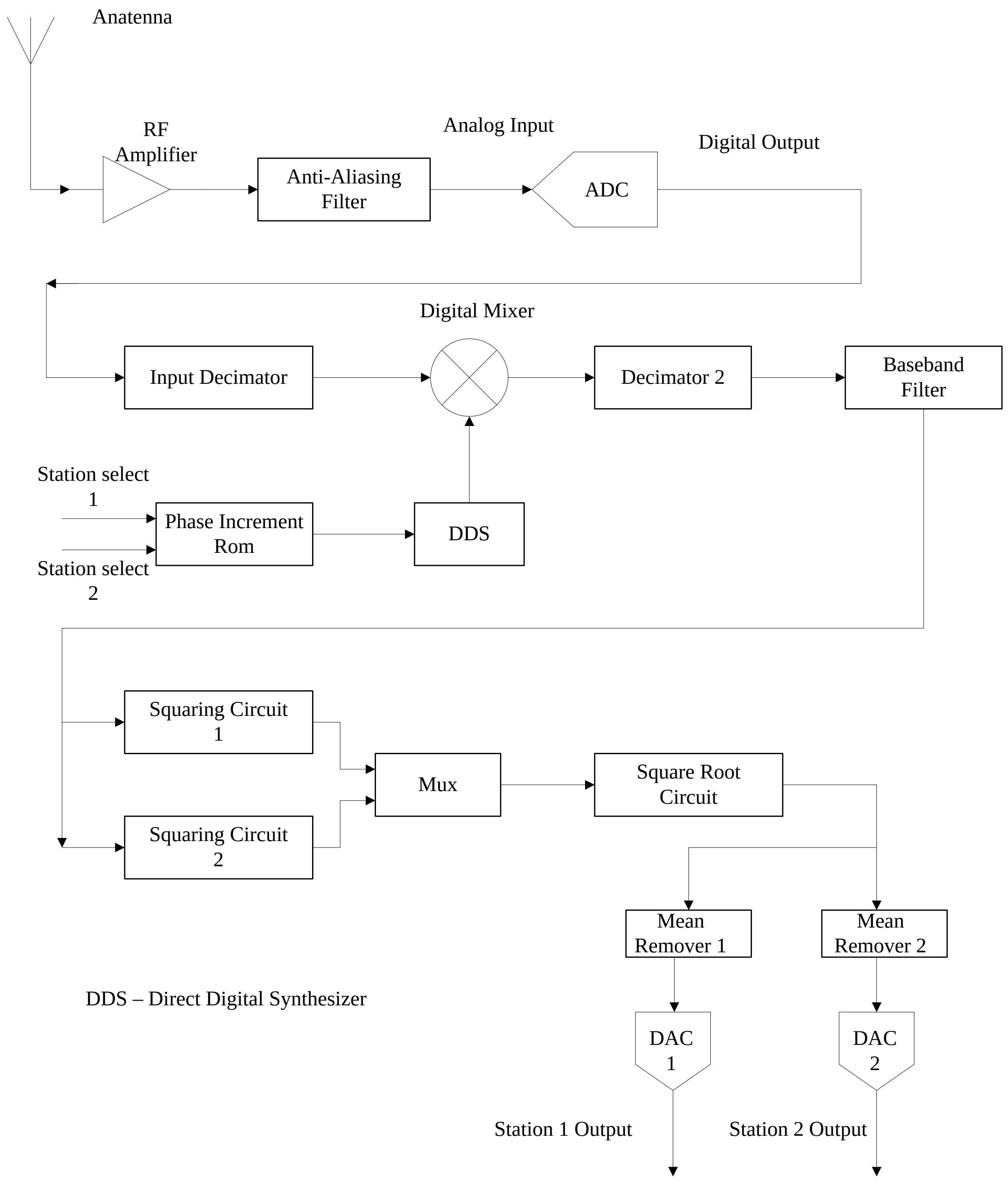


Figure 3.1 Top Level View of Multiple Channel TDM SDR Receiver

3.1 Analog to Digital Converter

The Analog to Digital Converter (ADC) of Figure 3.1 converts continuous time analog signals to discrete time digital signals. The analog to digital conversion process is illustrated in Figure 3.2

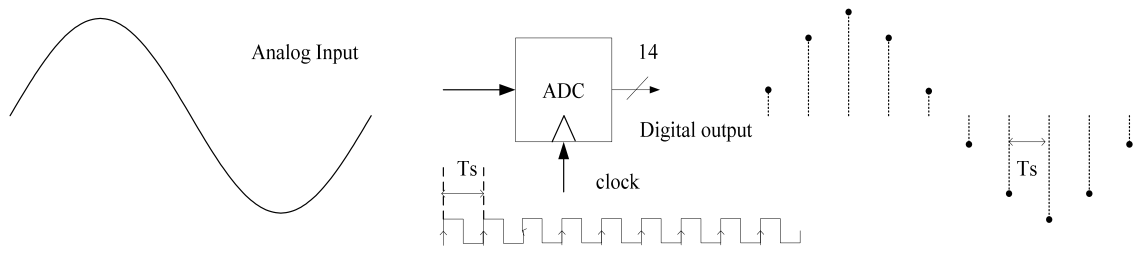


Figure 3.2 Analog to Digital Conversion

As seen in Figure 3.2, the ADC samples the continuous analog signal at every rising edge of the clock signal, converts it to digital and presents it at the output. The number of bits used to represent the digital output is called the resolution of the ADC. The resolution of an ADC indicates the number of discrete quantized values that it can produce over the range of analog values. An increase in resolution of the design will result in proportional increase in the accuracy of the results produced but at the expense of increased logic resource utilization.

According to Nyquist’s Sampling theorem, if a continuous time signal is sampled at a rate greater than twice its bandwidth then it can be reconstructed from its discrete samples [2]. In the USA, AM band extends from 535 to 1705 KHz [4], so the ADC should be run at least at a frequency greater than (1705 – 535) * 2 = 2340 KHz. If the ADC is run at a frequency Fs, then the analog signal applied to the input of the ADC should not contain any frequency components above Fs/2 to avoid aliasing [5]. This filtering is done by the RF amplifier which essentially acts as an Anti-Aliasing filter for the ADC.

An Analog Devices AD6645 [7] ADC is used in the implementation. The ADC has a 14 bit resolution but only the least significant 12 bits are utilized in the system and the rest of the bits are truncated off. The ADC presents the output in a 2’s complement fixed point fractional format.

3.2 Input Decimator

As mentioned earlier, if a logic resource has to be shared among ‘N’ streams of data each with an input sample rate ‘fs’, then the logic resource should be operated at frequency ‘N*fs’ to maintain the sample rate ‘fs’ for each stream. As shown in Figure 3.2 the input sample rate equals the frequency at which theADC is operated. The logic resource utilization of a DSP system is proportional to the sample rate at which it has to operate. The sample rate of a signal can be decreased to reduce logic resource utilization. The sample rate of a signal can be decreased by using decimation. Decimation is a process of filtering followed by downsampling. Downsampling is a process of retaining some samples of the input signal and throwing others away. Filtering is required before downsampling to prevent aliasing. So, in Figure 3.3 the filter before the downsampler effectively acts as an anti-aliasing filter for the downsampler. If it is required to decimate the sampling rate down to f_D, then the anti-aliasing filter should be designed to remove all the frequency components above $f_D/2$.

Both the ADC and the design are driven are operated at a frequency of 25 MHz. Since the input signal is oversampled, the input decimator is used to reduce the sample rate of the input signal by a factor of 4. Figure 3.3 shows the block diagram of the input decimator. Since the highest frequency of interest in an AM band is 1.7 MHz, the cutoff frequency of the anti-aliasing filter, before the downsampler, is set to that value. The frequency response of the anti-aliasing low-pass filter is shown in Figure 3.4. The anti-alias filter, before the downsampler is implemented using a parallel Distributed Arithmetic (DA) Finite Impulse Response (FIR) filter core. The number of Taps required to implement the filter are 46. The FIR filter is explained in detail in section 3.6.

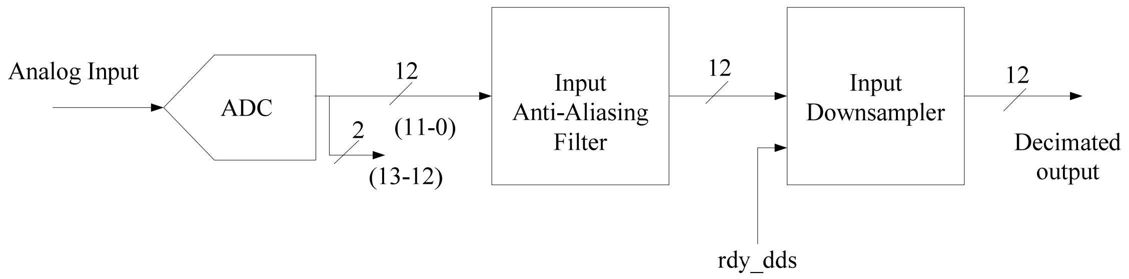


Figure 3.3 Block Diagram of Input Decimator

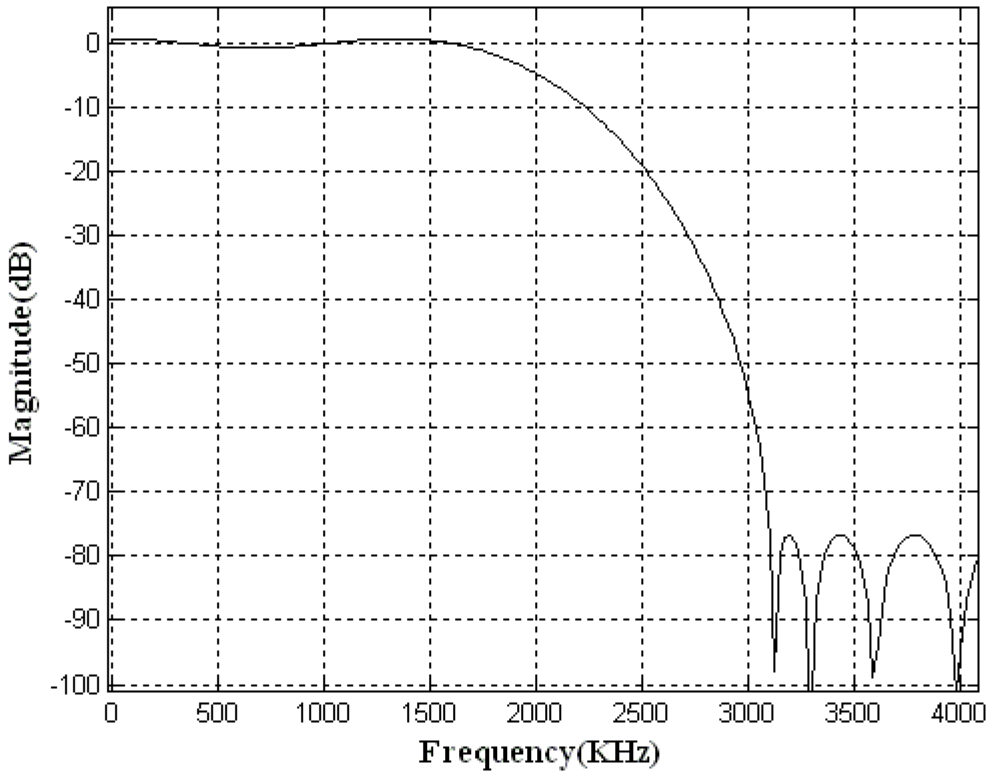


Figure 3.4 Frequency Response of Input Anti-Aliasing Low-Pass FIR Filter

The block diagram of an input downsampler is shown in Figure 3.5. The downsampler receives samples from the anti-aliasing filter. The downsampling factor depends upon the counter, the larger the counter, the larger the downsampling factor.

Downsampling is enabled when the ‘rdy_dds’ signal from the Direct Digital Synthesizer goes high. When ‘rdy_dds’ is low, the register is enabled and then latches all samples from the anti-aliasing filter, hence there is no downsampling. When ‘rdy_dds’ is high, the counter is enabled, and samples from the anti-aliasing filter are latched into register only when count reaches 3. Therefore when the counter is enabled only one in 4 samples are retained downsampling the input signal sample rate by a factor of 4.

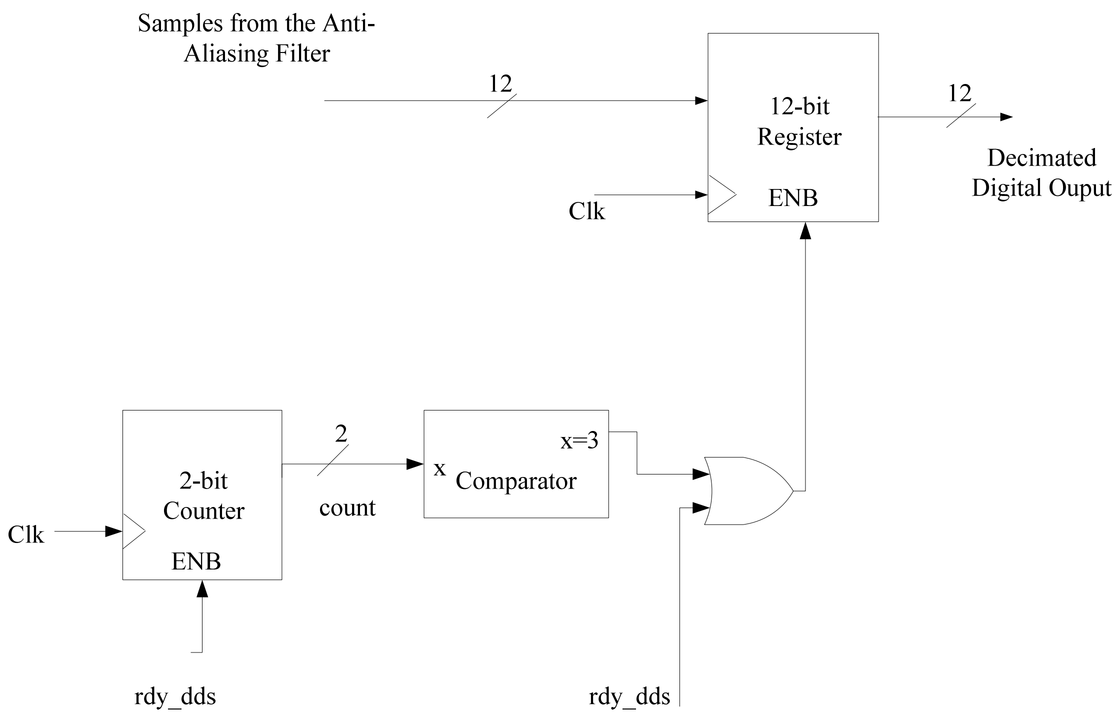


Figure 3.5 Input Downsampler

3.3 Direct Digital Synthesizer (DDS)

Direct Digital Synthesizers (DDS) are used to digitally generate sinusoidal signals. DDS offer such benefits as fast switching between output frequencies and fine frequency resolution [6]. The mathematical representation of a sinusoidal signal is given by Equation 3.1.

$$y(t) = \cos(2\Pi ft + 2\Pi\Psi) \tag{3.1}$$
$$\Psi \in \{-1,+1\}$$

In Equation 3.1, f is the frequency of the sinusoidal signal, t the time and $2\Pi\Psi$ the initial phase. In digital systems, t can only assume discrete values which are integral multiples of clock signal's time period, T. Substituting t = $n * T$ ($n = 0,1,2,3\ldots\infty$) in Equation 3.1,

$$y(nT) = \cos(2\Pi f * nT + 2\Pi\Psi) \tag{3.2}$$
$$T = 1/f_{clk}$$

For a given sinusoidal signal frequency f, $f * T$ would be constant and is called the phase increment for that frequency. Replacing $f * T$ in Equation 3.2 by $\Delta\Theta$ resulting,

$$y(nT) = \cos(2\Pi * \Delta\Theta * n + 2\Pi\Psi) \tag{3.3}$$

If 'r' bits are used to represent $\Delta\Theta$ and Ψ in fixed point arithematic, then they can be expressed as shown in Equation 3.4.

$$\Delta\Theta \cong \frac{\Delta\Theta_{dec}}{2^r}\ , \Psi \cong \frac{\Psi_{dec}}{2^r} \tag{3.4}$$

$\Delta\Theta_{dec}$ and Ψ_{dec} are the decimal equivalents of the binary numbers stored in the 'r' bit registers. Substituting $\Delta\Theta$ and Ψ from Equation 3.4 into Equation 3.3 results in,

$$y(nT) = \cos\left(2\Pi * \frac{\Delta\Theta_{dec}}{2^r} * n + 2\Pi * \frac{\Psi_{dec}}{2^r}\right) \tag{3.5}$$

Equation 3.5 can be implemented using the DDS architecture shown in Figure 3.6

3.3.1 Single Channel DDS

A functional block diagram of a single channel DDS is shown in Figure 3.6.

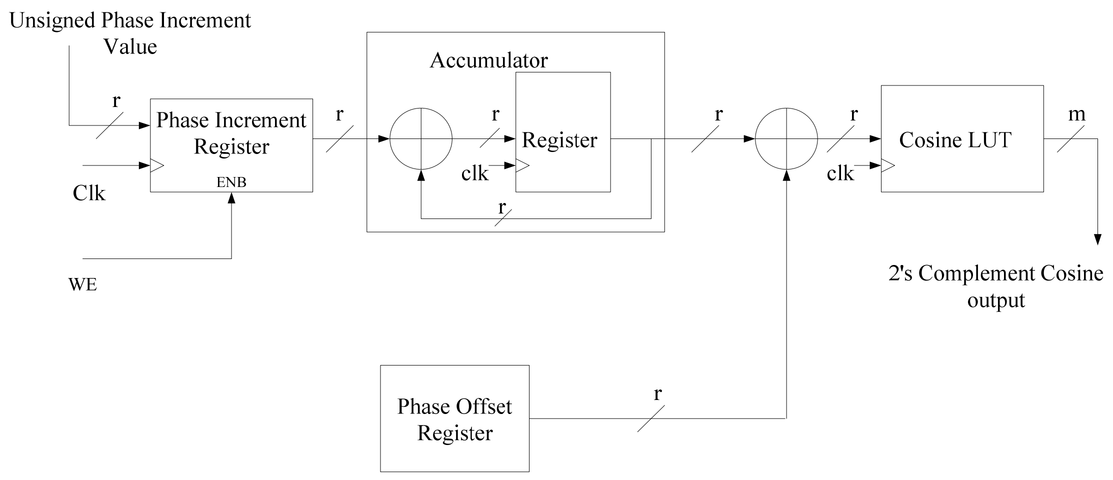


Figure 3.6 Block Diagram of a Single Channel DDS

The phase increment register holds the phase increment value corresponding to a given frequency. The frequency of the generated cosine wave can be changed by loading a different phase increment value into the phase increment register (PIR). A new value can be loaded into the phase increment register by making the write enable signal ‘WE’ go high. The accumulator generates the phase argument for the cosine Look Up Table (LUT) by integrating the phase increment register value. The cosine LUT takes the phase argument value from the accumulator and generates the corresponding cosine value. A constant phase offset value can be added to the accumulator output to generate sinusoidal signal with an initial phase. The Phase Offset Register (POR) is used to hold the phase offset value.

The cosine LUT is nothing but a ROM which stores the sampled values of a cosine wave [8]. The phase argument serves as the address for the ROM. The width of the ROM is equal to the number of bits used to represent the output cosine value and the depth of the ROM is equal to 2^r, where r is the number of bits used to represent the input phase argument value.

Frequency resolution of the DDS depends on the number of bits used to represent the phase increment value (r) and the clock frequency (f_{clk}) [9]. The frequency resolution of the DDS is given by Equation 3.6.

$$\Delta f = \frac{f_{clk}}{2^r} \tag{3.6}$$

The frequency resolution of the DDS can be improved by increasing the number of bits used to represent the phase increment value. Increasing the number of bits used to represent the phase increment value will increase the depth of the cosine LUT ROM. By comparing Equations 3.2 and 3.5, the frequency of the generated sinusoidal signal can be obtained as,

$$f = \frac{f_{clk} * \Delta\Theta_{dec}}{2^r} Hz \tag{3.7}$$

From Equation 3.7 it can be seen that larger the phase increment value stored in the phase increment register larger would be the frequency of the generated output signal would be and vice versa.

3.3.2 Multiple Channel DDS

The multiple channel DDS architecture is shown in Figure 3.7. The multiple channel DDS architecture shares the sine LUT across multiple channels on a time division basis. The output of the counter which counts from 0 to 3 is used as the select input for the multiplexer. Every clock edge the cosine LUT receives a phase argument from a different channel. So, on every clock edge the cosine LUT produces a sinusoidal output corresponding to a different channel. The cosine LUT takes one clock cycle to compute the output corresponding to a phase argument. So the output of the counter can be delayed using a register to indicate which channel is presently available at the output of the LUT.

As mentioned earlier, channels 0 and 1 correspond to I and Q streams of station 1. So channel 0 and channel 1 should produce sinusoidal signals which are of the same frequency but differ by a phase of $\Pi/2$ radians. Channels 0 and 1 can be made to produce sinusoidal signals of the same frequency by programming the same phase increment value into both phase increment registers, PIR – 0 and PIR – 1. The value that needs to be programmed into a phase offset register to produce a phase offset of Ψ radians is given by Equation 3.8.

$$offset_{POR} = \frac{2^r * \Psi}{2\Pi} \tag{3.8}$$

Now channels 0 and 1 can be made to differ in phase by $\Pi/2$ radians, by programming phase offset registers POR – 0 and POR – 1 with values corresponding to 0 radians and $\Pi/2$ radians respectively.

The same explanation holds for channels 2 and 3.

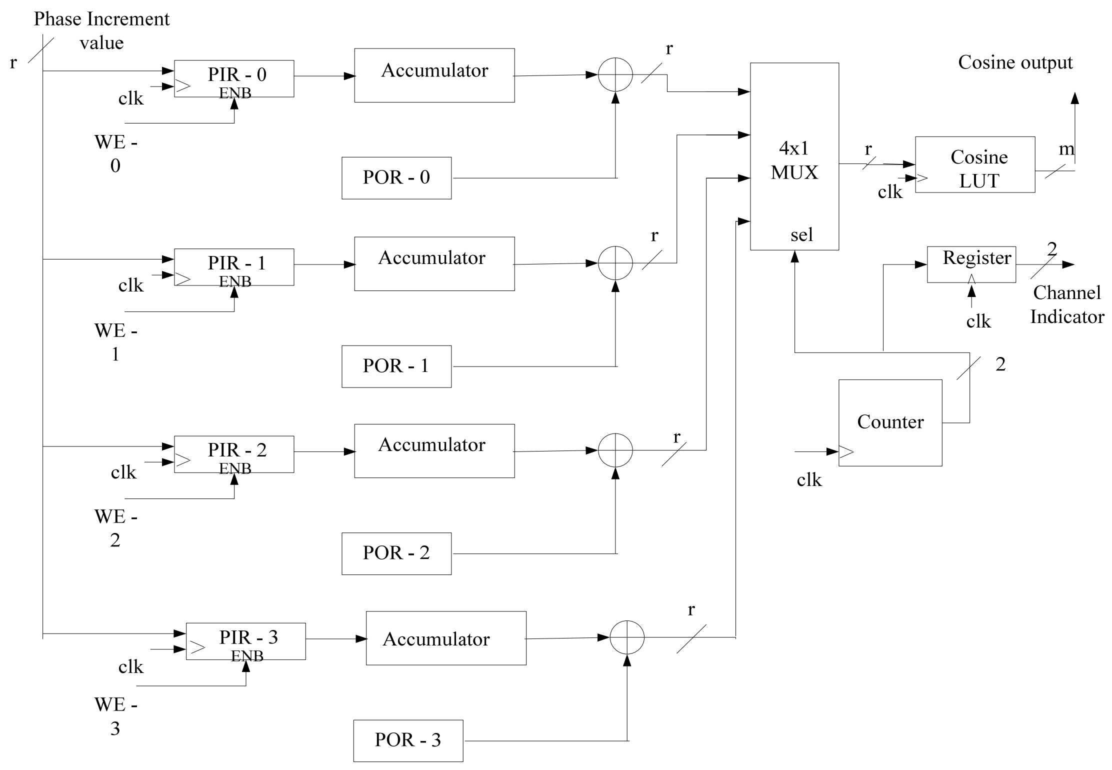


Figure 3.7 Block Diagram of Multiple Channel DDS

3.3.3 Xilinx DDS Core

The multiple channel DDS functionality will be implemented using a DDS Intellectual Property (IP) Core from Xilinx [9]. 12 bits were used to represent the sinusoidal output and 32 bits for the phase increment value. Figure 3.8 shows a top level view of the Xilinx DDS IP Core. All the control signals shown are active high.

Phase increment values are supplied at the 'data' port. Signal 'we' is used to latch the phase increment value available at the 'data' port into one of the 4 phase increment registers. One of the 4 phase increment registers can be selected using port 'a'. When the synchronous clear signal 'clr' is asserted all the registers in the core are cleared and output signal 'rdy_dds' deasserted. The sinusoidal output is presented at port 'cosine'. The output signal 'rdy_dds' indicates when the outputs from the core become available.

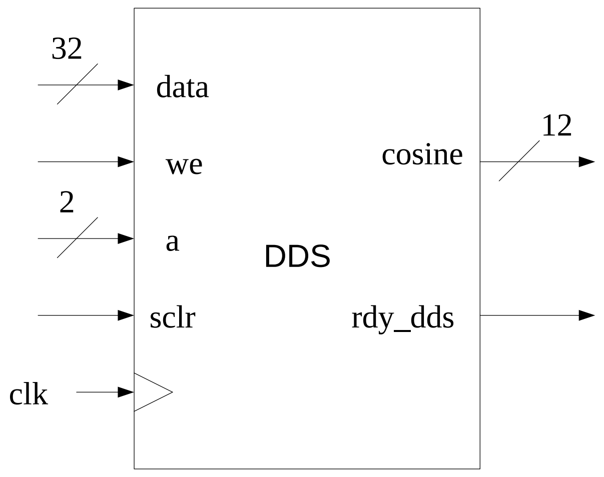


Figure 3.8 Top level view of Xilinx DDS Core

3.4 Phase Increment ROM

The receiver can be made to tune into a frequency by adjusting the DDS to that frequency. The output frequency of the DDS can be varied by changing phase increment value. The phase increment value $\Delta\Theta_{dec}$ that would need to be programmed into the phase increment register to obtain an output sinusoidal signal of frequency (f) can be obtained from Equation 3.9,

$$\Delta\Theta_{dec} = \frac{f * 2^r}{f_{clk}} \tag{3.9}$$

Instead of a user supplying the phase increment values for every frequency, the pre-computed phase increment values for all the supported frequencies can be calculated (using Equation 3.9) and stored in a ROM. The phase increment value for a particular frequency can now be obtained by supplying the appropriate address to the phase increment ROM.

A single phase increment ROM can be shared among the select inputs of stations, station 0 and station 1, using a multiplexer as shown in Figure 3.9.

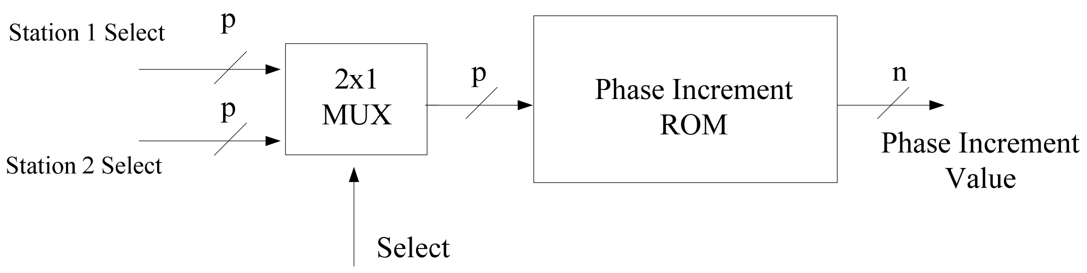


Figure 3.9 Phase Increment ROM

The multiplexer select signal is controlled by the FSM controller.

3.5 Digital Mixer

A digital mixer is a multiplier which multiplies signals from the input donwsampler and DDS. The mixer output contains the desired downconverted baseband component and unwanted high frequency components. The unwanted high frequency components can be removed using a low-pass FIR filter.

3.6 FIR Filter

Filtering is a method by which some frequency components of the input signal are passed to the output without attenuation while the others are highly attenuated. Filtering can be implemented digitally by using a finite impulse response (FIR) filter. The impulse response of a FIR filter is finite and hence the name. The advantages of FIR filters are they are stable, have linear phase response and require no feedback. The input output relation of an FIR filter is given by Equation 3.10.

$$y[n] = \sum_{k=0}^{N} a_k x[n-k] \tag{3.10}$$

In Equation 3.10 $x[n]$ is the nth input, $y[n]$ the nth output and a_k are the filter coefficients. The filter coefficients are constant for a particular filter configuration. The filter is said to be of Nth order and contain N+1 taps. Any type of frequency response can be implemented using Equation 3.10 by choosing the order of the filter and the filter coefficients. Matlab’s Filter Design and Analysis Tool (FDATool) was used to calculate the filter order and coefficients.

Both filters, FIR 1 and FIR 2, used in the design, are low-pass filters. Figure 3.10 shows the frequency response of a low-pass filter.

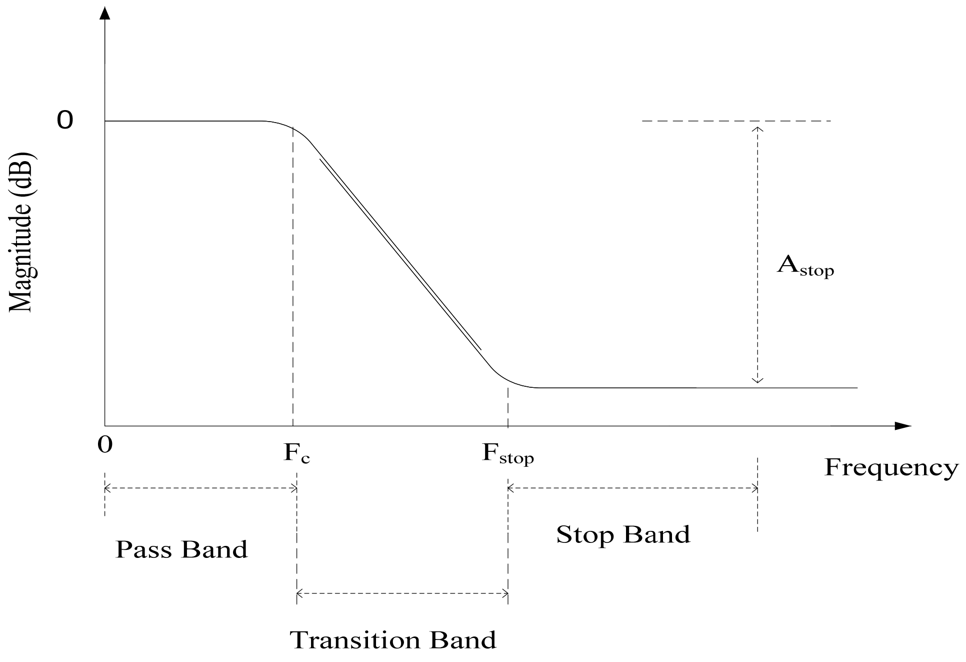


Figure 3.10 Frequency Response of an Ideal Low-Pass Filter

The band of frequencies from 0 to F_c is called pass band. Frequencies in the pass band suffer Zero or very small attenuation. The frequency F_c is often called the cutoff frequency of the filter. The band of frequencies lying above F_{stop} is called stop band. Frequencies in the stop band are highly attenuated. Compared to signals in the pass band, signals in the stop band are attenuated by a factor 'A_{stop}'. The band of frequencies lying between F_c and F_{stop} is called transition band. Frequencies in the transition band suffer attenuation between 0 dB and - A_{stop} dB. It is desirable that the width of the transition band should be small. The width of the transition band can be decreased by increasing the order of the filter.

3.6.1 Distributed Arithmetic (DA) FIR Filter

Equation 3.10 can be implemented by using multipliers, adders and delay elements as shown in Figure 3.11. The delay elements shown in Figure 3.11 are implemented using memory elements, at any time only N most recent inputs need to be stored.

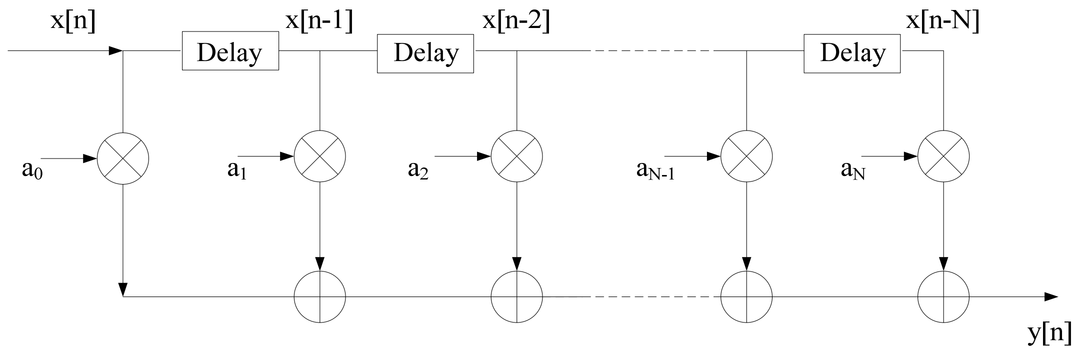


Figure 3.11 Conceptual View of an FIR Filter Implementation

The filter can be implemented either serially or in parallel. Implementing an Nth order FIR filter requires N+1 Multiply-And-Accumulate (MAC) units. Implementing the FIR filter using MAC units is expensive as it consumes lot of logic resources. Alternatively Distributed Arithmetic (DA) architecture can be used, which is very efficient in implementing Sum-Of-Products (SOP) [10]. DA implements MAC operations using LUTs instead of dedicated multipliers [11]. DA is bit serial in nature and parallel implementations can be developed by using serial DA FIRs in parallel.

Let the input variable $x[n-k]$, which is in 2's complement fixed point fractional format, contain 'R' bits and let $|x[n-k]| < 1$. It can then be expressed as shown in Equation 3.11.

$$x[n-k] = -x_{k,0} + \sum_{r=1}^{R-1} x_{k,r} 2^{-r} \tag{3.11}$$

In Equation 3.11, $x_{k,0}$ is the Most Significant Bit (MSB) or sign bit and $x_{k,R-1}$ is the Least Significant Bit (LSB) of the 'R' bit variable x [n-k]. It must be noted that $x_{k,r}$ is a binary variable and can only assume values 0 or 1.

Substituting Equation 3.11 in Equation 3.10 gives,

$$y[n] = -\sum_{k=0}^{N} x_{k,0} \bullet a_k + \sum_{k=0}^{N} \sum_{r=1}^{R-1} x_{k,r} \bullet a_k 2^{-r} \tag{3.12}$$

Expanding and rearranging Equation 3.12 gives [12],

$$
\begin{aligned}
y[n] = &-[x_{0,0} \bullet a_0 + x_{1,0} \bullet a_1 + x_{2,0} \bullet a_2 + \ldots\ldots + x_{N,0} \bullet a_N] \\
&+ [x_{0,1} \bullet a_0 + x_{1,1} \bullet a_1 + x_{2,1} \bullet a_2 + \ldots\ldots + x_{N,1} \bullet a_N]2^{-1} \\
&+ [x_{0,2} \bullet a_0 + x_{1,2} \bullet a_1 + x_{2,2} \bullet a_2 + \ldots\ldots + x_{N,2} \bullet a_N]2^{-2} \\
&\quad \circ \\
&\quad \circ \\
&\quad \circ \\
&\quad \circ \\
&+ [x_{0,R-1} \bullet a_0 + x_{1,R-1} \bullet a_1 + x_{2,R-1} \bullet a_2 + \ldots\ldots + x_{N,R-1} \bullet a_N]2^{-(R-1)} \quad (3.13)
\end{aligned}
$$

In Equation 3.13, each term inside the square brackets denote a logical AND operation and the plus sign denote arithmetic addition. The negative powers of two which appear outside the brackets can be implemented simply by shifting the results of the computation to the right. So the MAC operations in Equation 3.10 are now converted to addition, subtraction, shifting and logical AND operations. Instead of computing the values inside the brackets online they may be computed offline and stored in a LUT. Bits of the input variable can be used to address the LUT. Figure 3.12 shows the contents of the LUT used to implement a simple Equation.

A serial DA FIR filter can be constructed using a single LUT and time sharing it to process all the bits. Input shift registers (ISR) are required to supply bits serially to the LUT in serial DA FIR filter. Bits are output from the ISR MSB first. To construct a parallel DA FIR filter, ‘R’ LUTs are required. The MSB bits of all inputs are connected to the 0th LUT; the 1st bits of all inputs are connected to the 1st LUT and so on. Figures 3.13 and 3.14 show block diagrams of serial and parallel implementations of a single channel DA FIR filter respectively. The parallel filter produces one output every clock cycle whereas the serial filter produces one output every R clock cycles.

$$F = x_{0,0} \bullet a_0 + x_{1,0} \bullet a_1 + x_{2,0} \bullet a_2$$

$x_{0,0}$	$x_{1,0}$	$x_{2,0}$	Contents
0	0	0	0
0	0	1	a_2
0	1	0	a_1
0	1	1	a_2+a_1
1	0	0	a_0
1	0	1	a_0+a_2
1	1	0	a_0+a_1
1	1	1	$a_0+a_1+a_2$

Figure 3.12 Address and Contents of an LUT

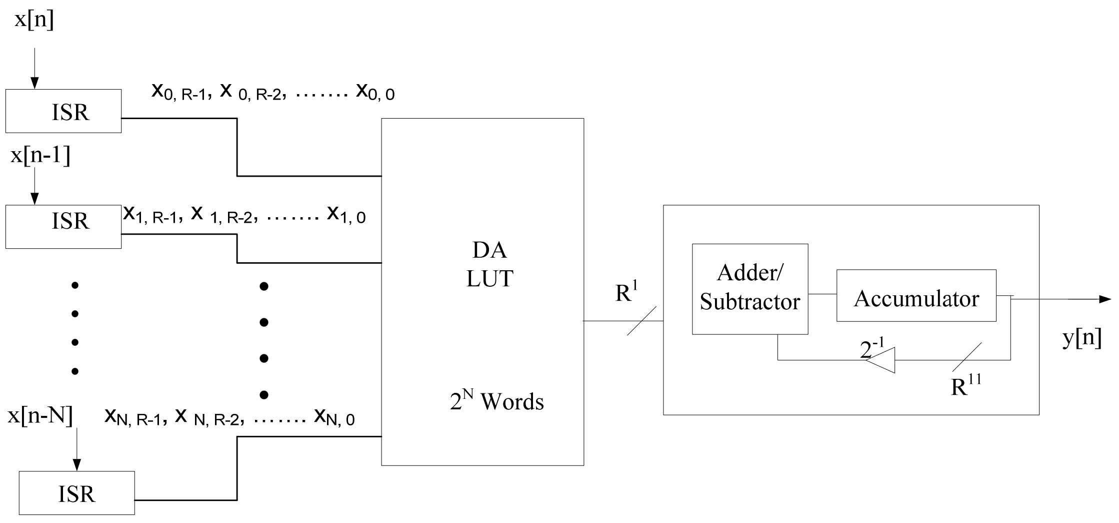

x[n]
ISR
x0, R-1, x 0, R-2, ……. x0, 0
x[n-1]
ISR
x1, R-1, x 1, R-2, ……. x1, 0
x[n-N]
xN, R-1, x N, R-2, ……. xN, 0
ISR
DA
LUT
2^N Words
R^1
Adder/
Subtractor
Accumulator
2^-1
R^11
y[n]

Figure 3.13 Serial DA FIR Filter

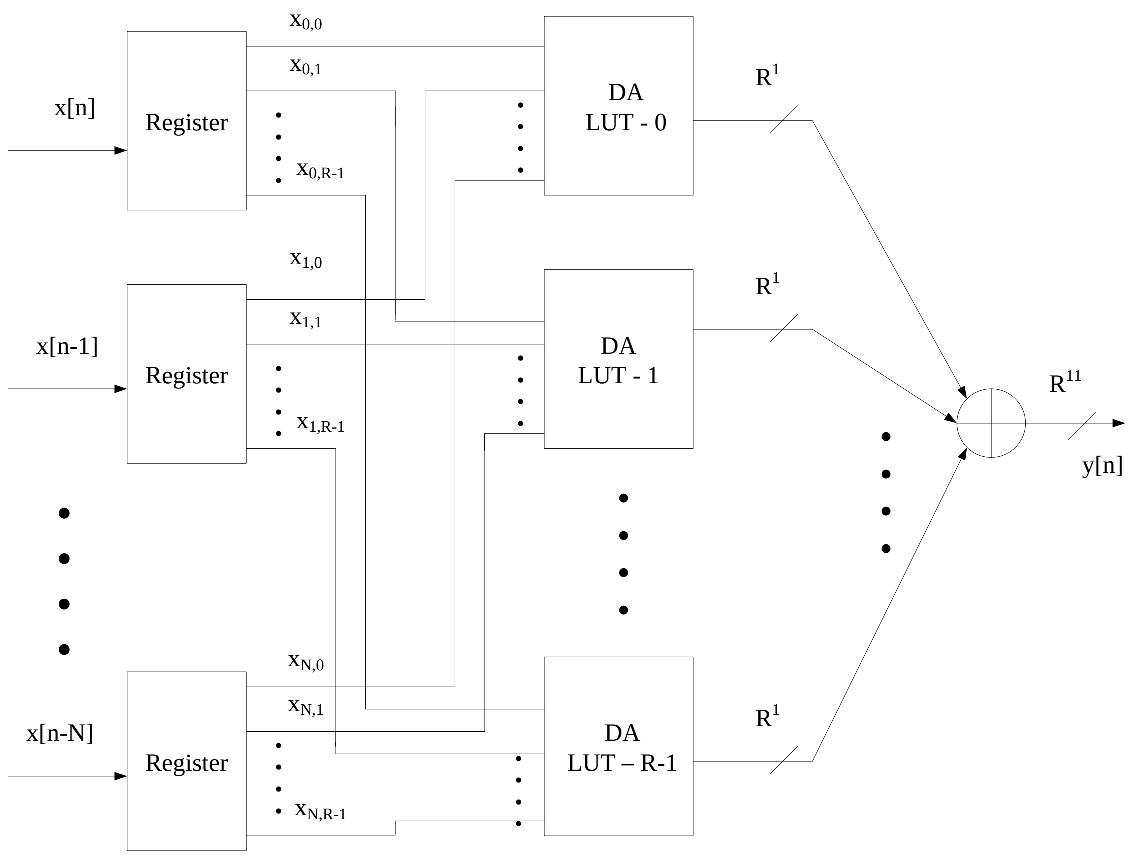


Figure 3.14 Parallel DA FIR Filter

Since all 4 channels have the same filtering requirements, a multi channel DA FIR filter can be constructed by time sharing DA LUTs across data from multiple channels. For a multi channel DA FIR filter, memory required the amount of memory required to store input variables will be more since it has to store input variables of multiple streams, but, the logic resources required to compute results would be the same as a single channel filter.

3.6.2 Xilinx DA FIR Core

Xilinx DA FIR IP Cores [13] will be utilized to implement the serial and parallel DA FIR operations. Both the serial and parallel implementations have the same external interface and pin definitions. Figure 3.15 shows a top level view of a Xilinx DA FIR Core. Both cores consist of 4 channels each. Twelve bits were used to represent the input data variables and 25 bits to represent the outputs.

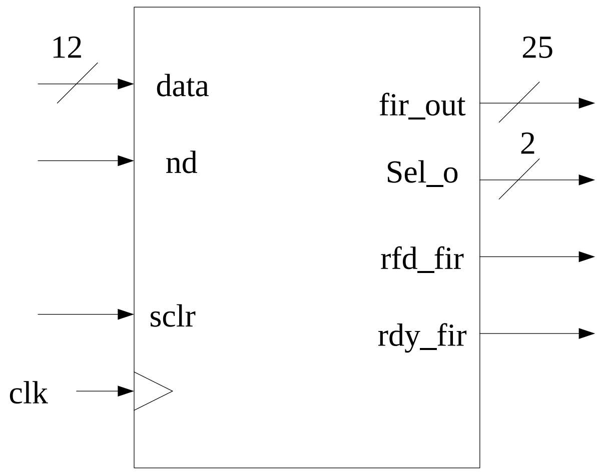


Figure 3.15 Top Level View of Xilinx DA FIR Core

All the shown control signals shown are active high. Input data must be supplied at the ‘data’ port. The input data supplied at the ‘data’ port is latched into an input data register at the rising edge of the clock if the ‘new data’ signal ‘nd’ is high. Signal ‘rfd_fir’ indicates that the core is ready to accept new data. New input should be supplied to the core only when the signal ‘rfd_fir’ is high. The output result of the core is presented at port ‘fir_out’. The signal ‘sel_o’ indicates which channel output is presently available at the port ‘fir_out’.

3.7 Decimator 2

The design is operated at a frequency of 25 MHz. So, the input sample rate for each channel would be 25/4 = 6.25 MHz. In the US, AM transmission channels are spaced every 10 KHz. The bandwidth of the audio signals transmitted over AM is usually within 4 KHz. So, a low-pass FIR filter with F_c = 4.5 KHz and F_{stop} = 9.5 KHz is required to filter out the baseband signal from the digital mixer output without any adjacent transmission channel interference.

A filter with such narrow transition band width, operating at such a large sample rate (6.25 MHz) will have a large order. For larger order filter more logic resources will be required to implement the filter. Table 3.1 compares filter orders for various sampling rates.

Table 3.1 Comparison of Filter Orders for Various Sampling Rates

F_c = 4.5 KHz , F_{stop} = 9.5 KHz	
Sample Rate (MHz)	Order of Filter
6.25	3163
3	1518
0.39	198

From table 3.1 it can be seen that as the sample rate decreases the order of the filter decreases. The sample rate of the signal at the mixer output can be decreased by using decimation. Figure 3.16 shows the block diagram of the decimator followed by a low-pass baseband filter.

The anti-aliasing low-pass filter is designed with F_c = 4.5 KHz and F_{stop} = 150 KHz. The frequency response of the anti-aliasing low-pass filter used in decimator 2 is shown in Figure 3.17. The number of taps required to implement this filter is 110. The anti-aliasing filter is implemented using a 4 channel parallel DA FIR core. Since the highest frequency component that can be present at the output of the anti-aliasing filter is 150 KHz, the sample rate can now be brought down to a value slightly greater than 300 KHz. The signal at the output of the anti-aliasing filter is downsampled by a factor of 16, bringing the sample rate down to 390 KHz.

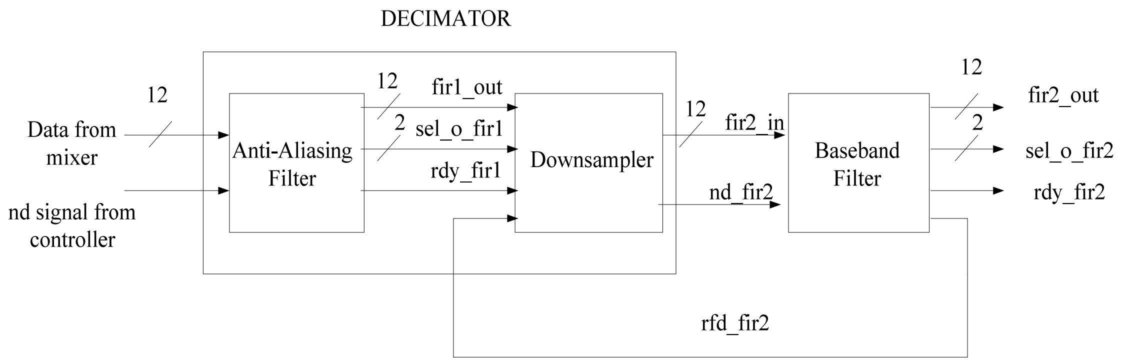


Figure 3.16 Block Diagram of a Decimator followed by Baseband Filter

The baseband filter, used to extract the baseband component is designed with Fc = 4.5 KHz and F_{stop} = 8.5 KHz. The frequency response of the baseband filter is shown in Figure 3.18. The number of taps required to implement this filter is 199. So, the total number of taps required now to filter the baseband component is, 110 + 199 = 309, instead of 3163 that would have been required if no decimation was used. Furthermore, input samples to the baseband filter are separated by 16 clock cycles and the serial DA FIR core takes only 12 clock cycles to compute a result, so the baseband filter can be implemented serially.

The downsampler shown in Figure 3.16, in addition to decreasing the sample rate by a factor of 16, also controls the baseband filter. The downsampler is responsible for providing proper input to the proper channel of the baseband filter every 12 clock cycles. The block diagram of downsampler is shown in Figure 3.19. The downsampler uses a FIFO to store samples from the anti-alias filter and supply them at the right time to the baseband filter. The counter counts from 0 to 63. The counter is enabled by the ready signal of the anti-aliasing filter ‘rdy_fir1’. A new sample is loaded into the FIFO if the ‘rdy_fir1’ signal is high and the count is less than or equal to 3. The empty signal from the FIFO indicates weather the FIFO is full or not. A new sample is read out of the FIFO if it is not empty (empty = ‘0’) and the baseband filter is ready to accept new data (rfd_fir2 = ‘1’). The read signal for the FIFO, ‘rd’ also serves as the ‘new data’ signal for the baseband filter, ‘nd_fir2’.

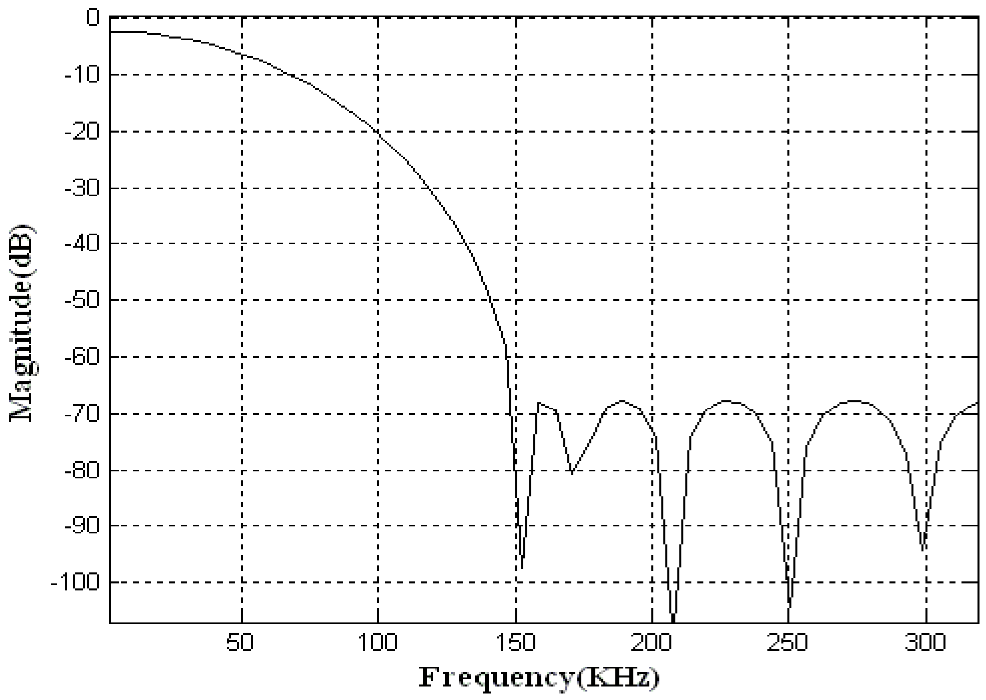


Figure 3.17 Frequency Response of Anti-Aliasing Filter used in Decimator 2.

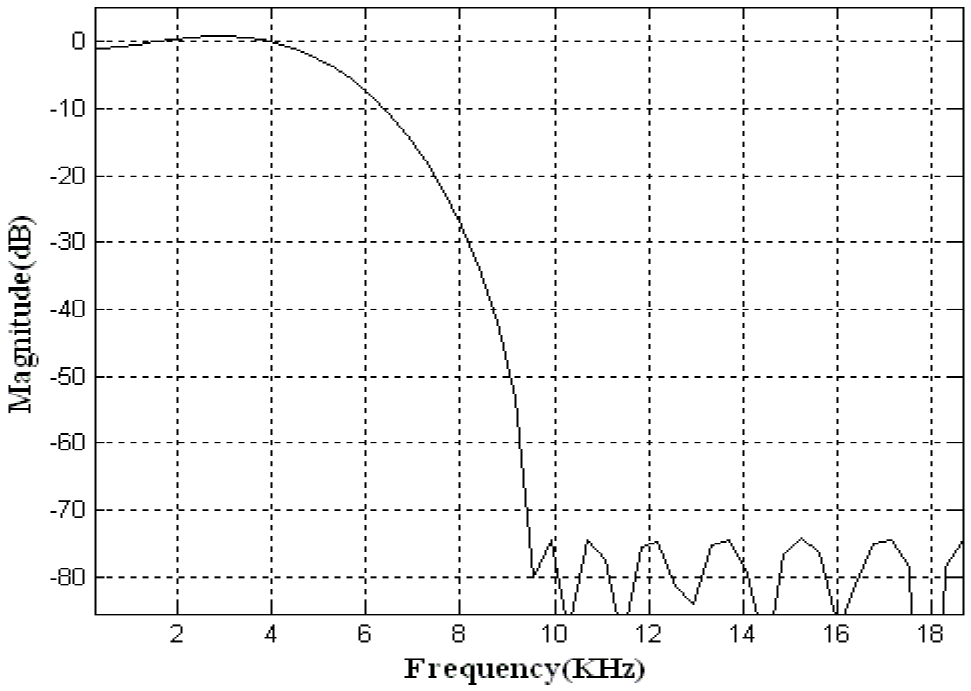


Figure 3.18 Frequency Response of Baseband Filter

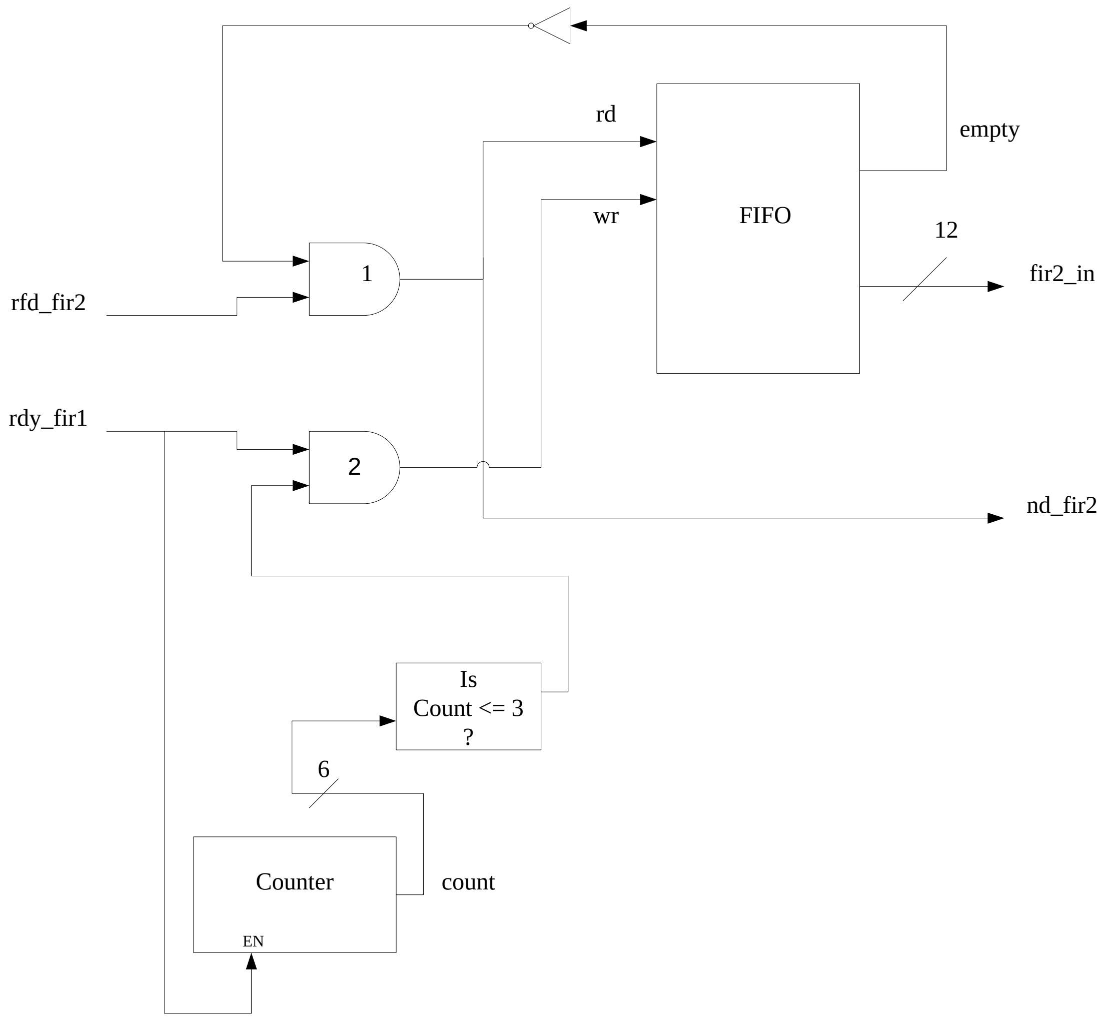


Figure 3.19 Block Diagram of Downsampler

3.8 Squaring Circuit

The squaring circuit of Figure 3.1 is used to square and add the I and Q channels of a station. Both the squaring circuits 1 and 2 used in the design are identical. Squaring circuits 1 and 2 are used to process the I and Q channels of stations 1 and 2 respectively. Figure 3.20 shows the conceptual view of a squaring circuit.

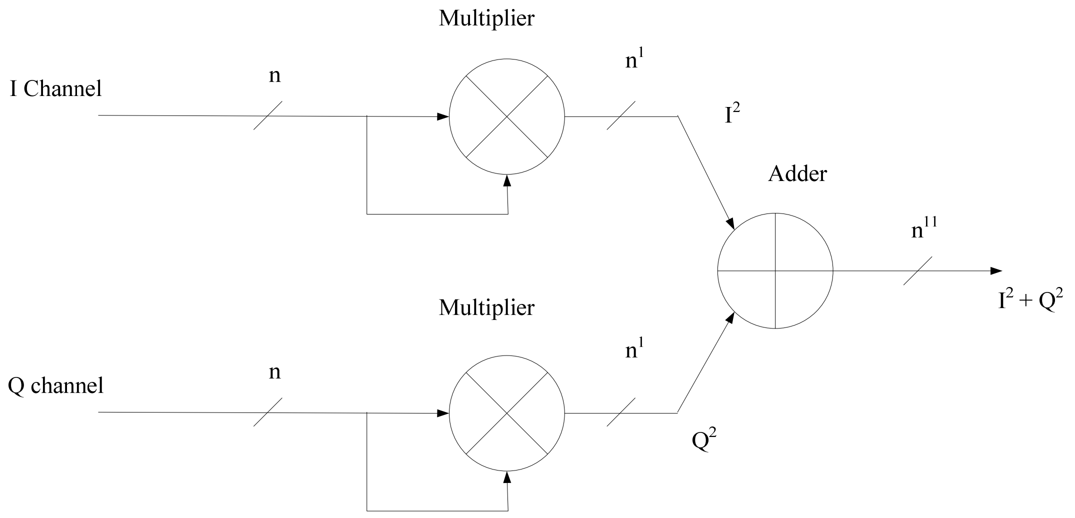


Figure 3.20 Conceptual View of a Squaring Circuit

The functional block diagram of pipelined squaring circuit 1 is shown in Figure 3.21. The squaring circuit operates on the outputs of the baseband filter. The output of the comparator block ‘comp 1’ is high when the signal ‘sel_o_fir2’ is 1. This signal is delayed through pipeline registers and used at the output of the squaring circuit to indicate when the output of the squaring circuit becomes ready. The output of the comparator circuit ‘comp 2’ is high when the signal ‘sel_o_fir2’ is either 0 or 1. This signal is Anded with the ready signal from the baseband filter, ‘rdy_fir2’, and the resultant signal is used as an enable input for pipeline register ‘REG 1’. So, channel 0 and channel 1 outputs can only be latched into pipeline register ‘REG 1’. The squaring is performed in stage 1. The output of the mixer is latched into one of the two registers, ‘A’ or ‘B’, depending upon the ‘SEL’ value. When ‘SEL’ is 0 register ‘A’ is enabled, when

‘SEL’ is 1 register ‘B’ is enabled. This implies that the squared value of channel 0 is stored in register ‘A’ and the squared value of channel 1 is stored in register ‘B’. The squared values of channel 0 and channel 1 are added in stage 2.

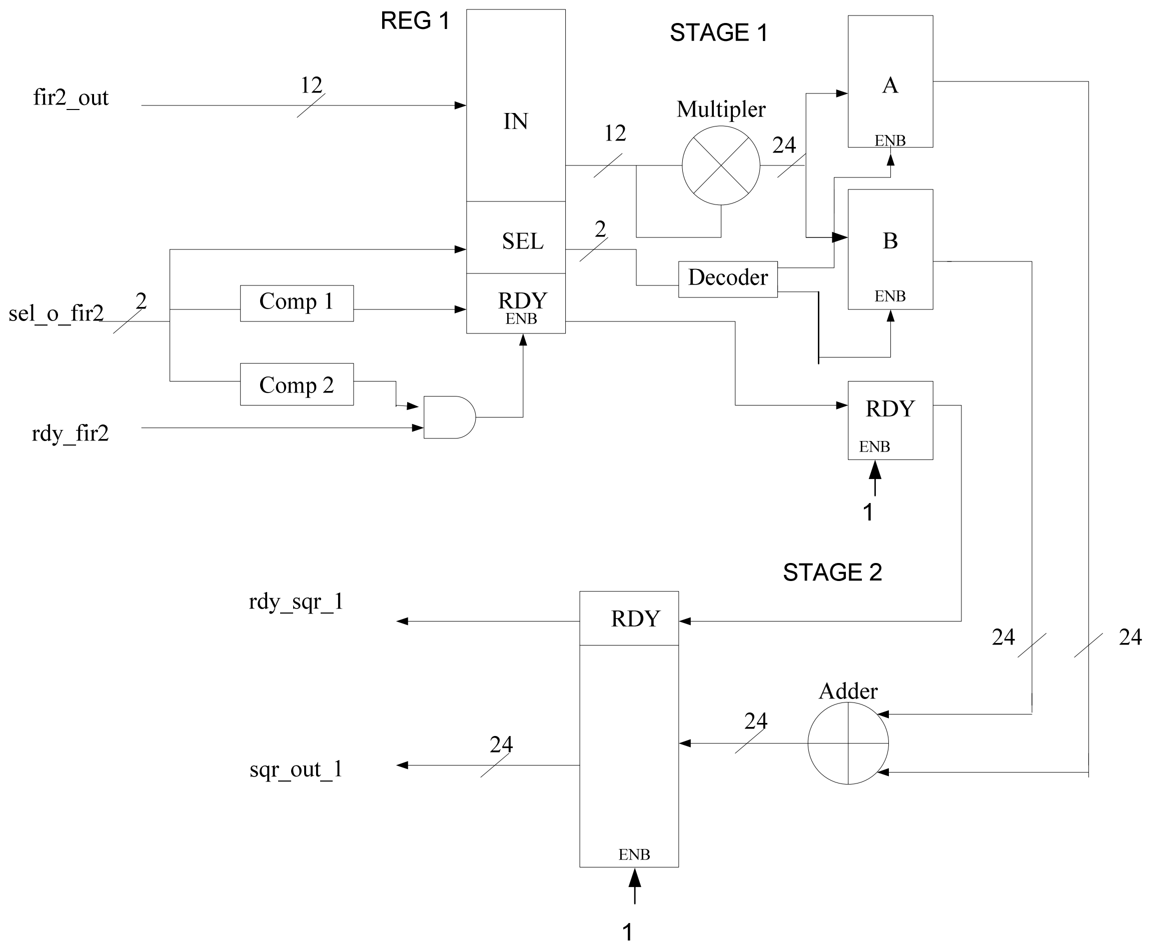


Figure 3.21 Functional Block Diagram of Pipelined Squaring Circuit 1.

3.9 Square Root Circuit

The square root circuit computes the square root of the squaring circuit output shown in Figure 3.21. The square root function is implemented using the modified Dijkstra’s algorithm developed by Tommiska [14]. This algorithm can be implemented without the use of multipliers or dividers. Calculating the square root of a n bit number requires n/2 clock cycles. The flow chart of the modified Dijkstra’s algorithm implemented by the Square Root Circuit of Figure 3.24 is shown in Figure 3.22.

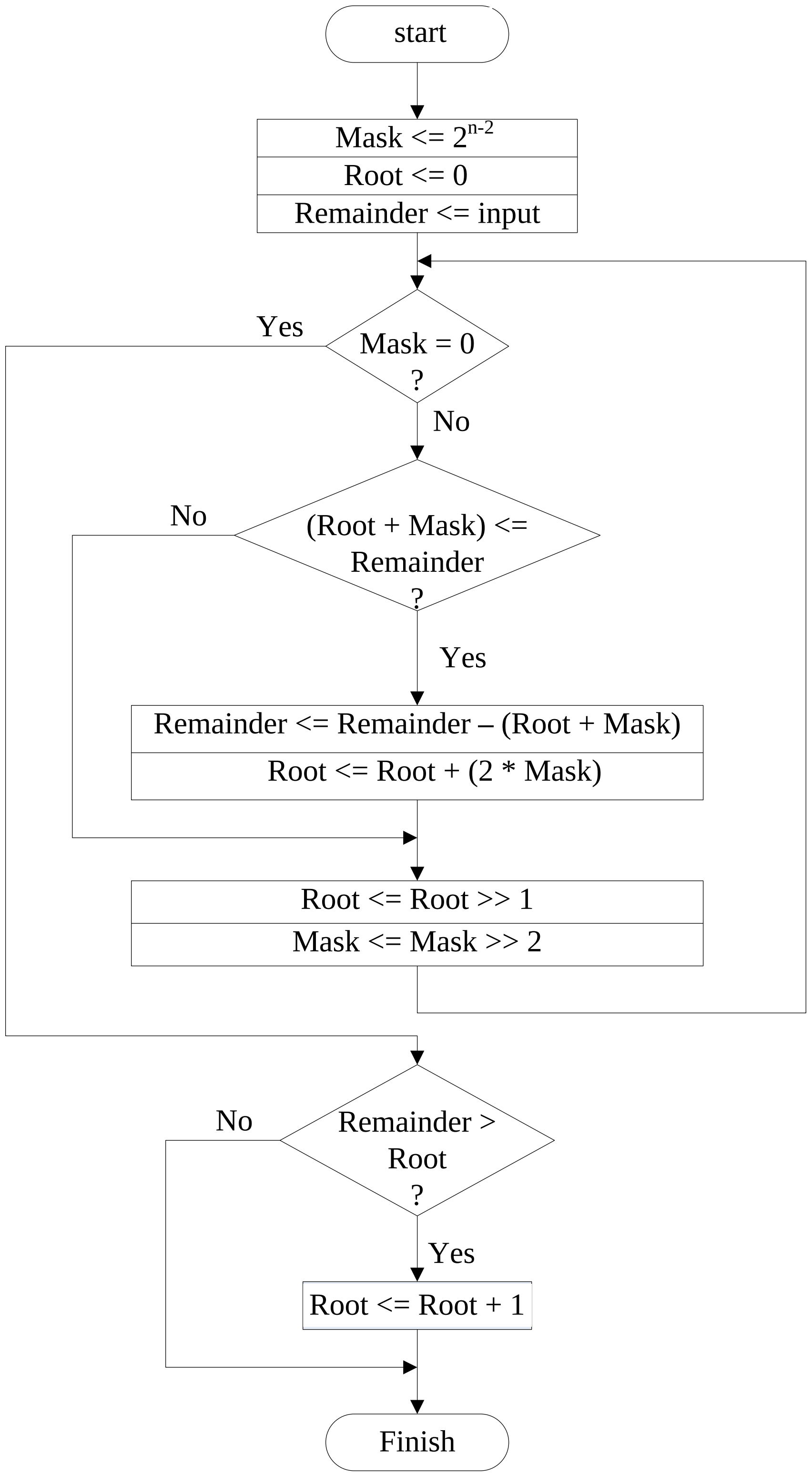


Figure 3.22 Flow Chart of Modified Dijkstra's Algorithm [14]

The algorithm uses three variables; ‘Mask’, ‘Root’ and ‘Remainder’ to calculate the square root of a number. It is assumed the number whose square root has to be calculated is an ‘n’ bit wide unsigned integer. At the beginning of the calculation the variables ‘Mask’ and ‘Root’ are loaded with values 2^{n-2} and 0 respectively. Also the number whose square root is to be calculated is assigned to the variable ‘Remainder’. At the end of the algorithm, the square root of the number would be present in the variable ‘Root’.

The above algorithm can be implemented either in a looped or a pipelined fashion. In either of the two implementations an iteration of the algorithm is performed every clock cycle. The basic block used in either of the two implementations (SQ_BIT) is shown in Figure 3.23.

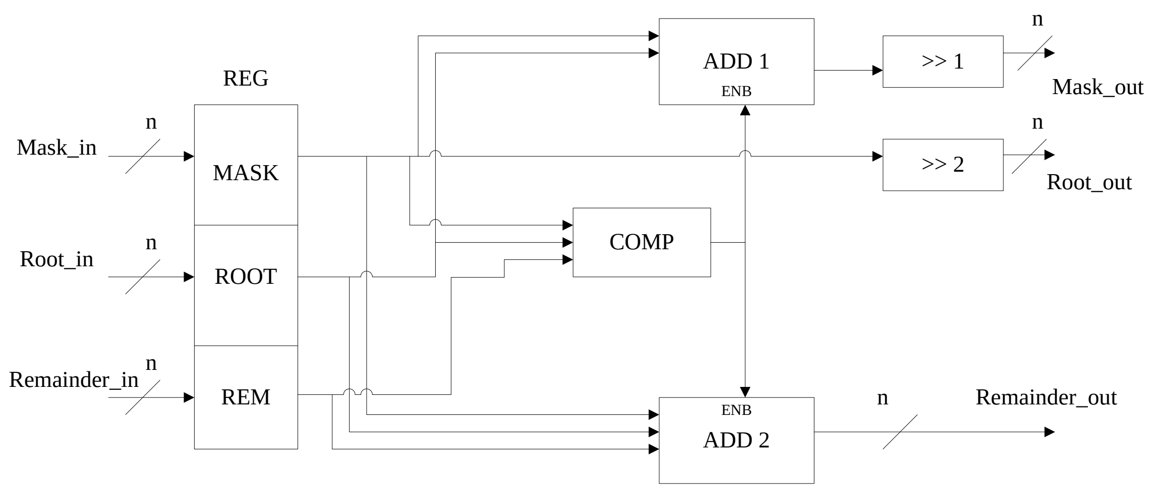


Figure 3.23 Block Diagram of SQ_BIT

The ‘COMP’ block checks weather the condition ‘(ROOT + MASK) <= REM’ is true or not. The ‘ADD 1’ block outputs the value ‘ROOT + (2 * MASK)’ when enabled, otherwise it outputs the value ‘ROOT’. Similarly the ‘ADD 2’ block outputs the value ‘REM – (ROOT + MASK)’ when enabled, otherwise it outputs the previous reminder value ‘REM’.

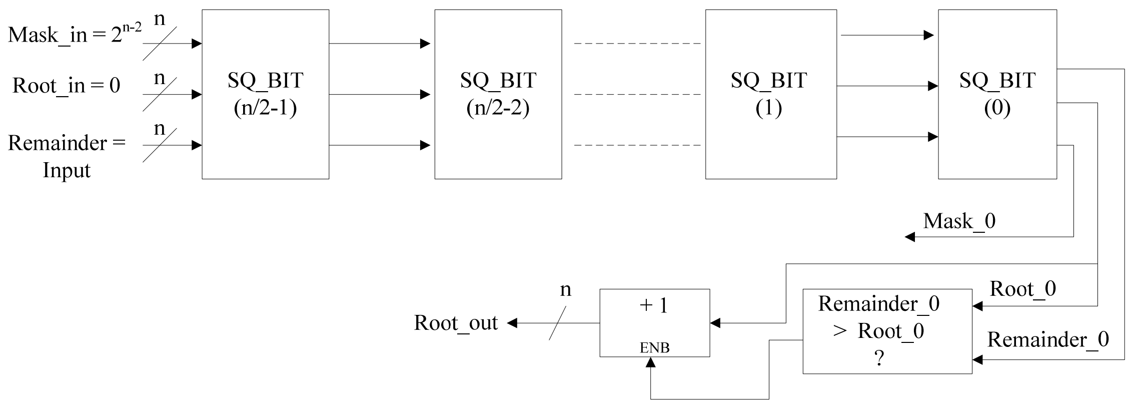


Figure 3.24 Fully Pipelined Implementation of a Square Root Circuit.

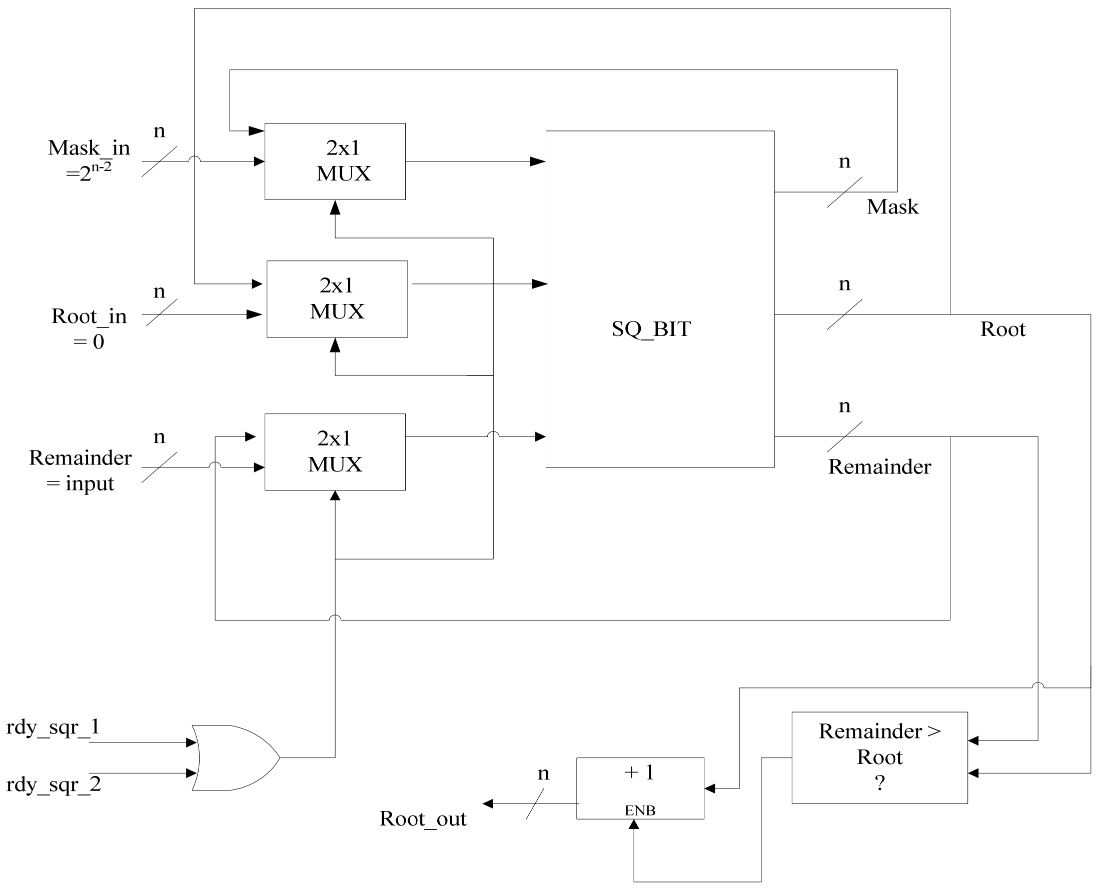


Figure 3.25 Loop Implementation of a Square Root Circuit.

A fully pipelined implementation is developed by cascading n/2 ‘SQ_BIT’ units as shown in Figure 3.24. A looped implementation is developed by using a single ‘SQ_BIT’ unit as shown in Figure 3.25. In the loop implementation, if the ready signal from either of the squaring circuits is high, input values are fed to the ‘SQ_BIT’ unit. Else, the previous outputs of the ‘SQ_BIT’ unit are fed back to its inputs. In both the loop and pipelined implementations, ready signals from the squaring circuits are delayed n/2 clock cycles and used at the output of the square root circuit to indicate which station output is presently available at the output.

3.10 Mean Remover

The mean remover circuit is used to remove any DC component that may be present at the output of the square root circuit. The presence of DC component will limit the range of voltage values over which the analog output of the DAC can swing. Therefore the DC component should be removed before a signal can be supplied to the DAC.

To remove the DC component from the square root circuit output, the mean value of the samples from square root circuit should be calculated and this value should be removed from the square root circuit samples. The mean value can be calculated using Equation 3.14 [24].

$$V_o = V_o^{1} + K(V_i - V_o^{1}) \qquad (3.14)$$

In Equation 3.14, ‘V_i’ is the input to the mean remover block, ‘V_o’ is the mean value and ‘V_o^{1}’ the previous mean value. The operation of the circuit is controlled by the constant ‘K’. The smaller this constant, the smoother will be the output but more number of iterations would be required initially to find the mean value and vice versa. The circuit which implements Equation 3.14 is shown in Figure 3.26.

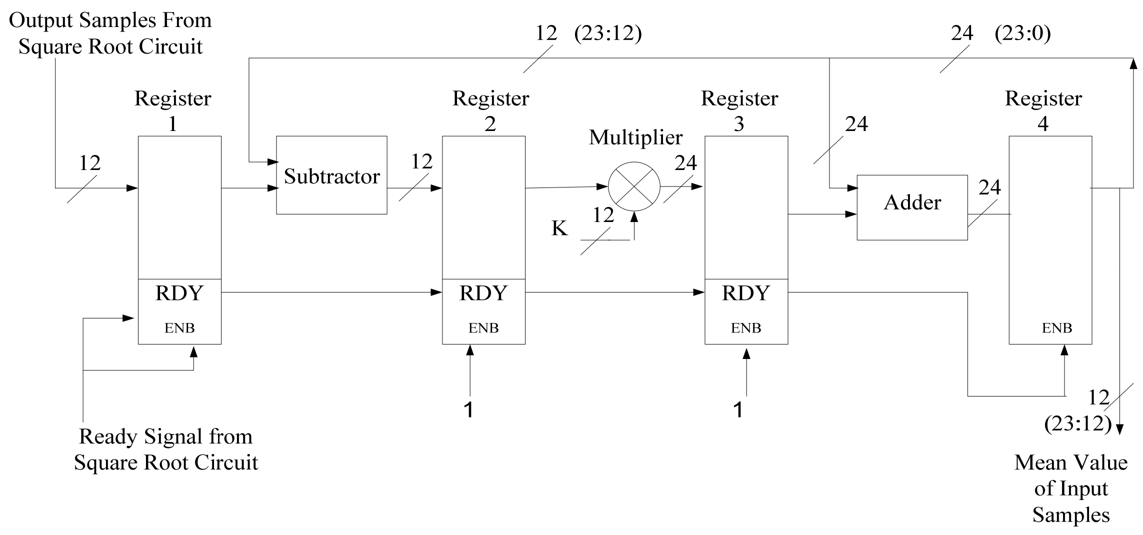


Figure 3.26 Mean Remover Circuit [24]

3.11 Digital to Analog Converter

The Digital to Analog Converters (DAC) of Figure 3.1 are used to convert discrete time digital samples to a continuous time analog signal. The DAC used in the implementation is analog devices AD9772A [15]. Two DACs were used as shown in Figure 3.1 one for each station.

3.12 Controller

The controller is only responsible for controlling the Phase Increment ROM, the DDS and the Anti-Alias Filter units. All the other units need no control inputs from the controller. The top level view of the controller is shown in Figure 3.27.

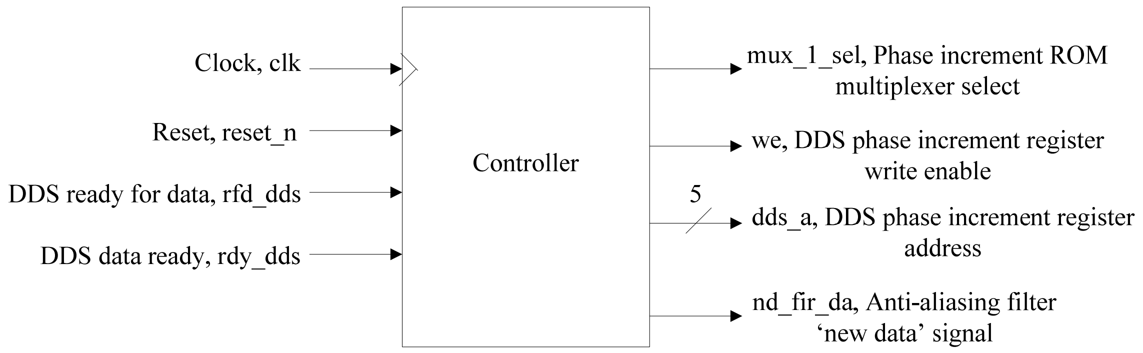

Clock, clk
Reset, reset_n
DDS ready for data, rfd_dds
DDS data ready, rdy_dds
Controller
mux_1_sel, Phase increment ROM multiplexer select
we, DDS phase increment register write enable
5
dds_a, DDS phase increment register address
nd_fir_da, Anti-aliasing filter 'new data' signal

Figure 3.27 Top Level View of Controller

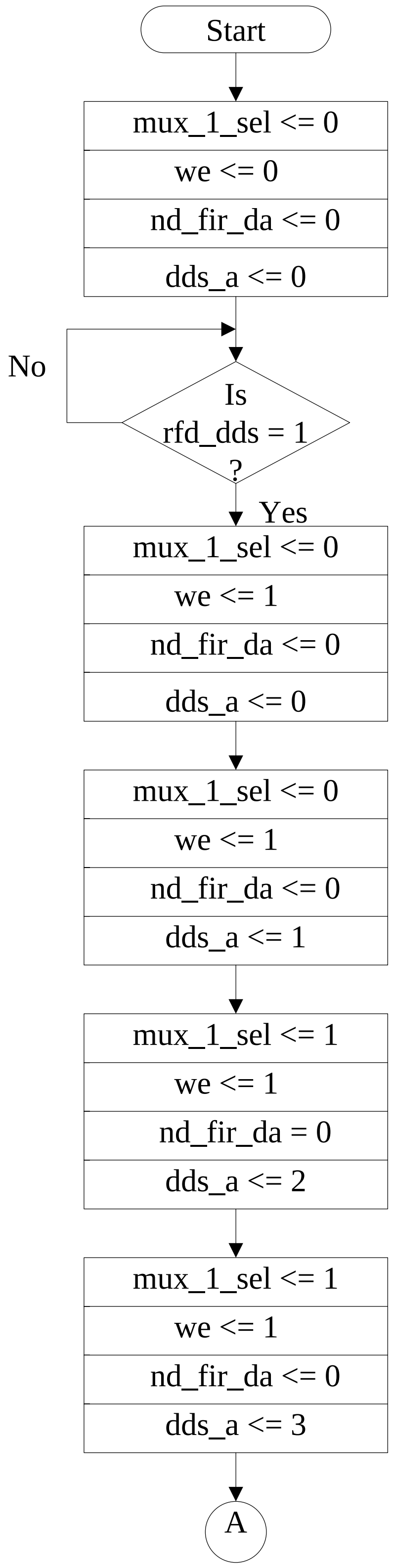


Figure 3.28 Controller Flow Chart

The system flow chart of the controller is shown in Figure 3.28. After the system comes out of reset the controller is responsible for latching in the appropriate phase increment values into the phase increment registers of the DDS. Once the cosine samples become ready at the output of the DDS the controller enables the ‘new data’ input of the anti-aliasing filter. After this point the system requires no intervention from the controller unless there is a reset.

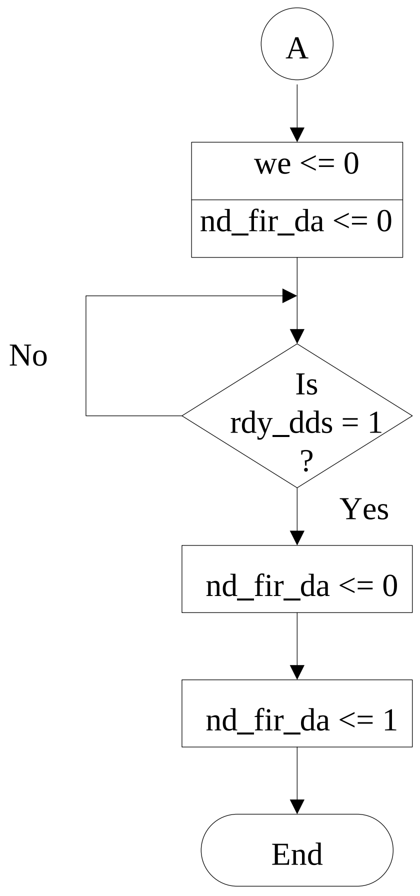


Figure 3.28 (Continued) Controller Flow Chart

In this chapter the Coherent TDM SDR architecture and various components that make up the architecture were described. The architecture will be validated using Post-

Synthesis and Post-Implementation HDL simulations in the next chapter. In a later chapter FPGA based hardware prototyping will be used to further validate the functionality and performance of the architecture.

Chapter 4

Post-Synthesis and Post-Implementation Simulation Testing and Validation of Coherent Multi-Channel TDM SDR Architecture

After a design has been captured using Hardware Descriptive Languages (HDL) and synthesized, it can be implemented on a Field Programmable Gate Array (FPGA) chip. If there are any bugs in the design it would be difficult to identify them once the design is implemented on a FPGA chip. This is because the FPGA chip has a limited number of I/O pins and not all the internal signals of the design can be brought out to the pins of the FPGA chip at the same time. Even if there are enough pins on the FPGA chip to bring out all the internal signals, debugging a design this way would be a tedious and time consuming process. Therefore, it is better to find and eliminate problems in the design using simulations before it is implemented on the FPGA chip. Post-Synthesis and Post-Implementation HDL simulations are two important simulation levels used to validate a design before it is implemented on a FPGA chip. Post-Synthesis simulation is used for functional validation and Post-Implementation simulation is used for both performance and timing validation of the design. This chapter describes the Post-Synthesis and Post-Implementation functional and performance simulation testing of the Coherent Multi-Channel TDM SDR architecture.

HDL Post-Synthesis and Post-Implementation simulation testing of the design was carried out using the Xilinx ISE 8.2i and Mentor Graphics Modelsim SE 6.1a CAD software tools. The computer system used to run the above mentioned CAD software has the following configuration; Intel Pentium 4 CPU, 1 GB RAM and Windows XP Professional Edition operating system. The design was implemented to a Xilinx Virtex II Pro FPGA XC2VP30.

The approach followed in testing the multi-channel TDM SDR architecture and design was a bottom up approach i.e., the lower level functional components were tested and validated first and then the validated lower level components were assembled to form higher level more complex functional units, which are again validated. The HDL simulation details of the lower level components are not presented. Only the Top level simulations are addressed in this chapter.

4.1 Post-Synthesis Simulation Testing and Validation

Synthesis is a process of transforming a design at Behavioral or Register Transfer Level (RTL) abstraction to gate level abstraction. The synthesized design may function erroneously either because of design errors made by the designer in capturing the design or because of errors in synthesis. Therefore, the synthesized design must be simulated in order to find and eliminate any design, HDL design capture or synthesis errors.

In order to validate the multi-channel coherent AM SDR receiver, two AM signals are generated, one at a frequency of 590 KHz and the other at a frequency of 600 KHz. These two AM waves are summed and the resultant sum signal samples are quantized to 12 bit 2’s complement values. The sample rate of the sum signal is 25 mega samples per second. These quantized samples are written to a text file ‘input_samples.txt’. The MATLAB m-file which performs the above operations can be found in Appendix B. The modulating signals for the 590 KHz station (modulating signal 1) and the 600 KHz station (modulating signal 2) are shown in Figures 4.1 and 4.2 respectively and their power spectral density plots are shown in Figures 4.3 and 4.4 respectively. The 590 KHz station and 600 KHz AM signals are shown in Figures 4.5 and 4.6 respectively. The quantized sum signal is shown in Figure 4.7.

The quantized sum signal shown in Figure 4.7 can be viewed as the output of the ADC of Figure 3.1. The samples corresponding to this signal are supplied as input to the design.

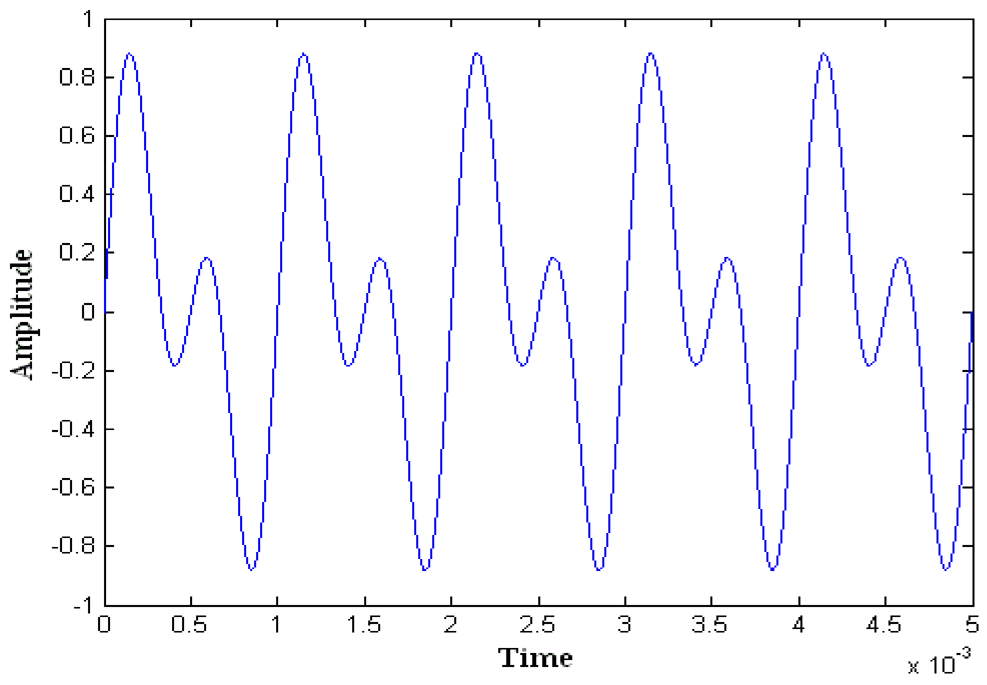


Figure 4.1 Modulating Signal 1

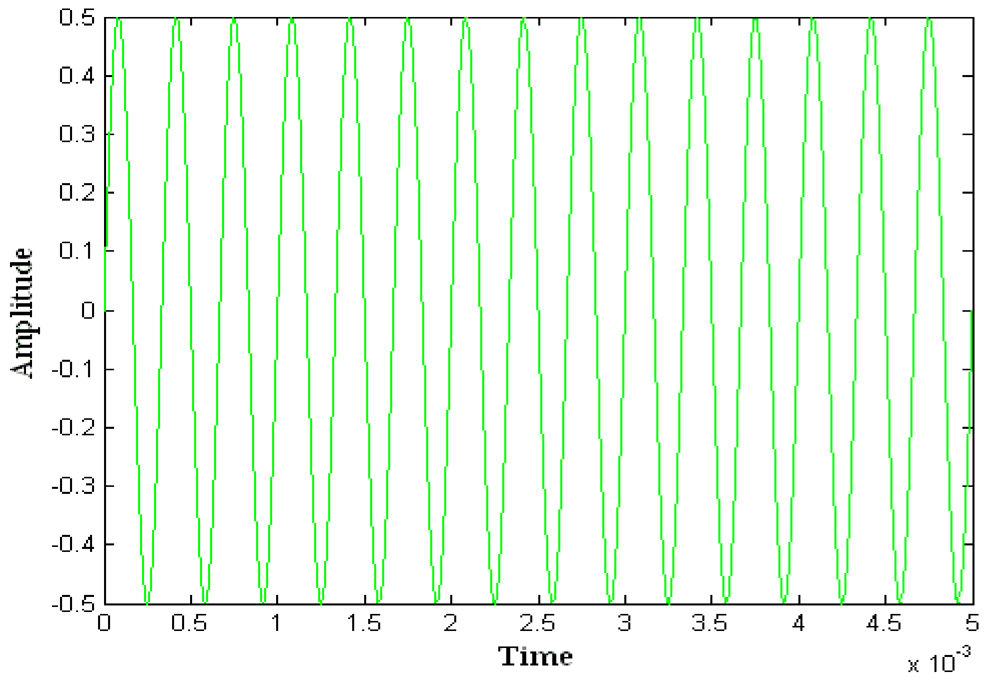


Figure 4.2 Modulating Signal 2

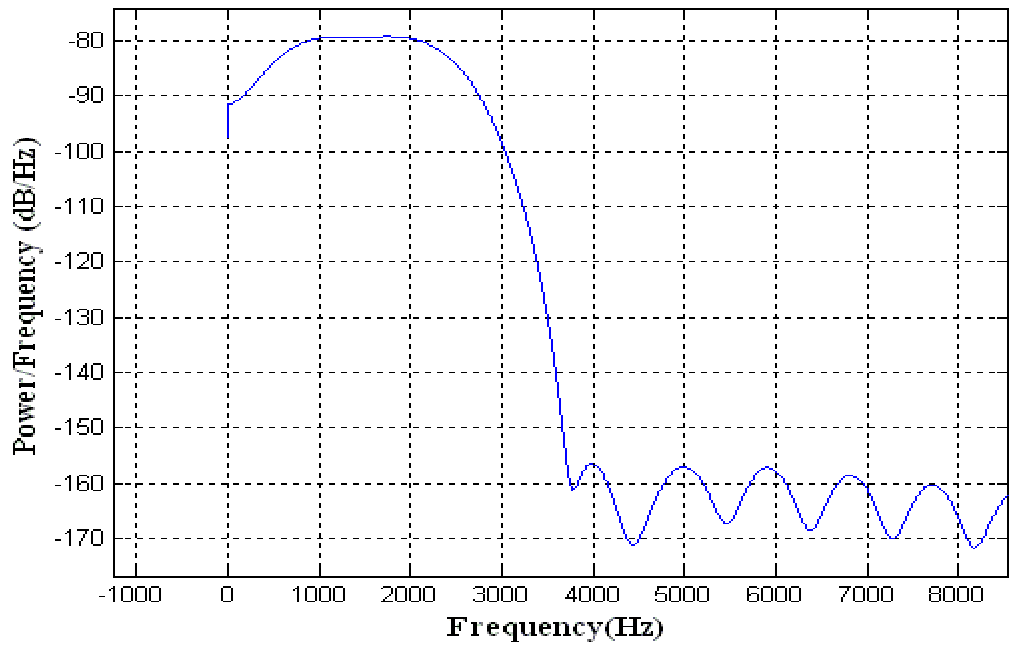


Figure 4.3 Power Spectral Density Plot of Modulating Signal 1

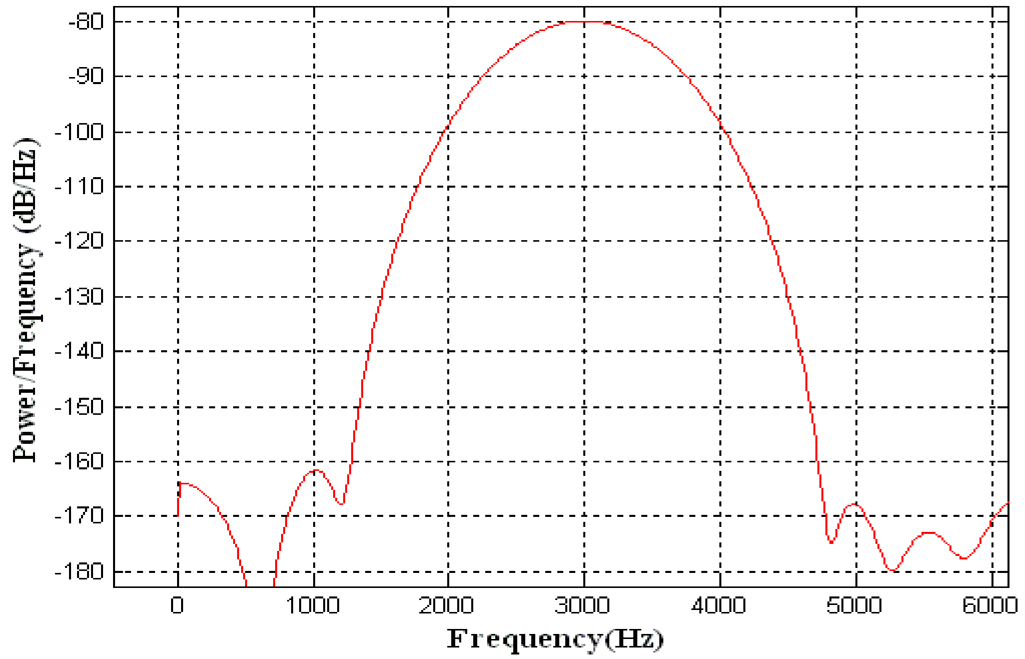


Figure 4.4 Power Spectral Density Plot of Modulating Signal 2

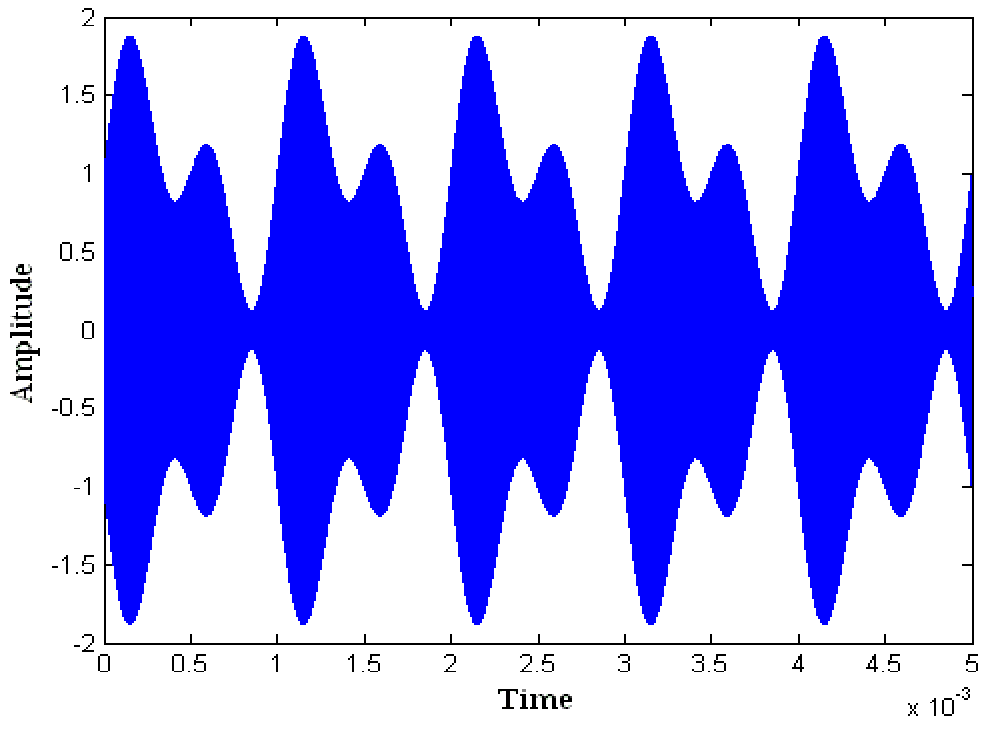


Figure 4.5 AM Signal corresponding to 590 KHz Station

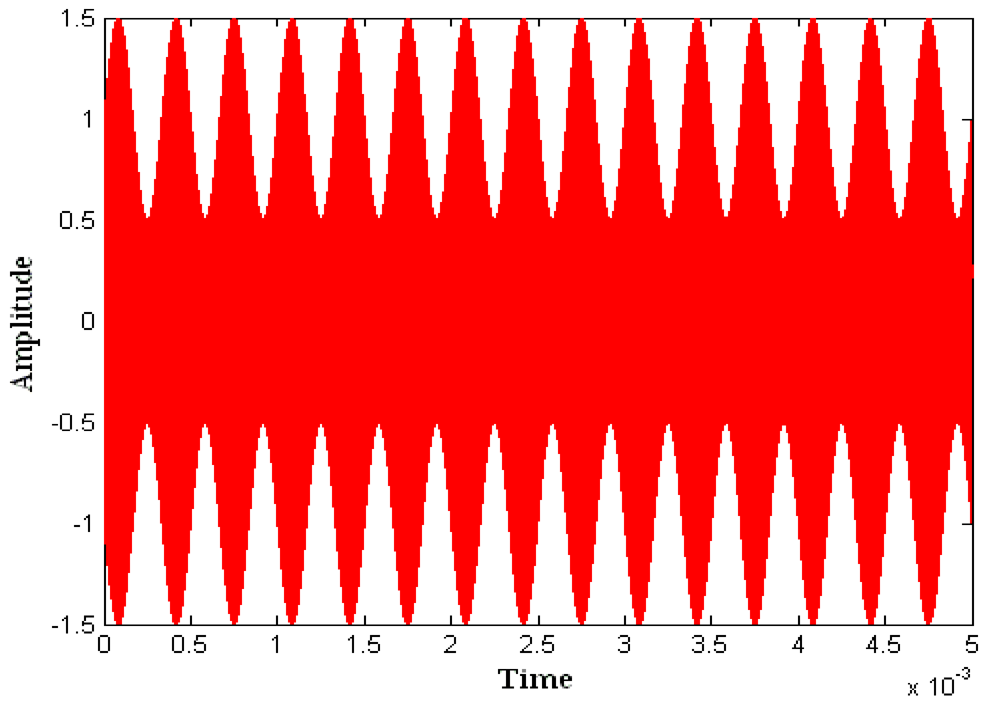


Figure 4.6 AM Signal corresponding to 600 KHz Station

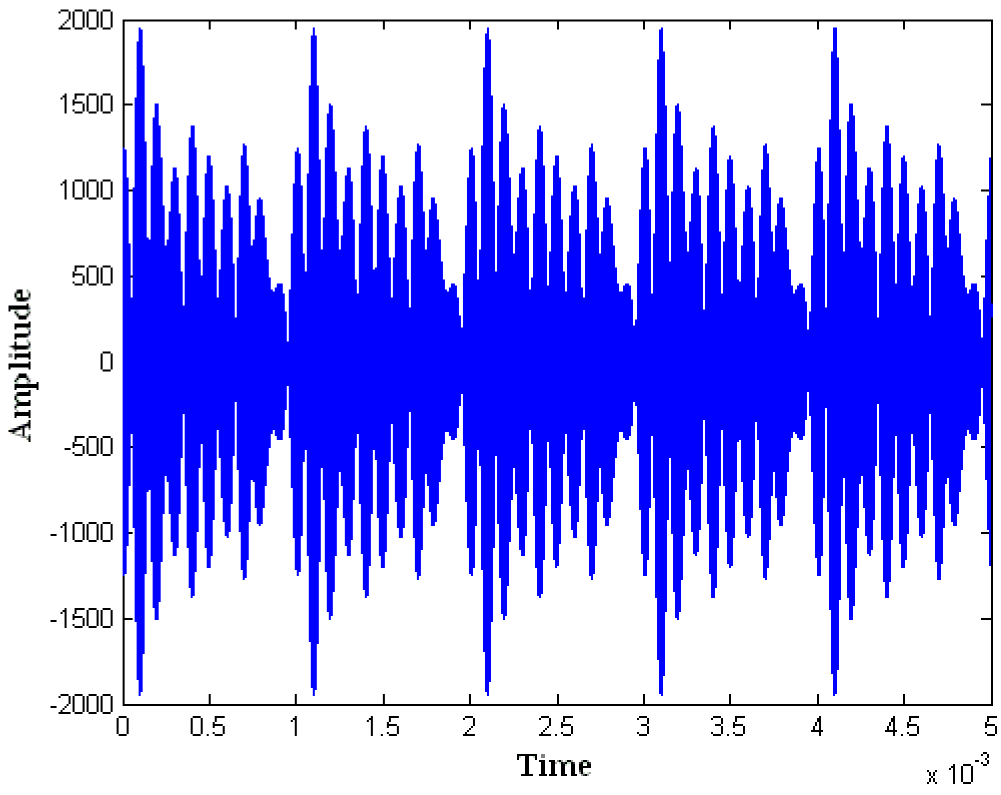


Figure 4.7 Quantized Sum Signal

Two important features of the design that need to be tested are its ability to successfully receive two different stations simultaneously and its ability to avoid adjacent channel interference. Both these features can be tested by making the design tune-in to two stations 10 KHz apart. If the design successfully demodulates two stations 10 KHz apart then it implies that both of the above mentioned features are validated.

The design was tuned in to frequencies 590 KHz and 600 KHz. A Verilog testbench was developed which takes samples from the text file 'input_samples.txt' and supplies them as input to the Design Under Test (DUT) shown in Figure 3.1. The testbench is also responsible for collecting output samples corresponding to the frequencies 590 KHz and 600 KHz from the DUT. The output samples are then stored in two text files 'demod_station1.txt'and 'demod_station2.txt' respectively. The testbench code can be found in appendix B. A section of the top level post-synthesis simulation waveform is shown in Figure 4.8. The signals 'dac_0', 'dac_1' shown in Figure 4.8

correspond to the outputs of the mean remover blocks ‘mean remover 1’ and ‘mean remover 2’ respectively which are shown in Figure 3.1.

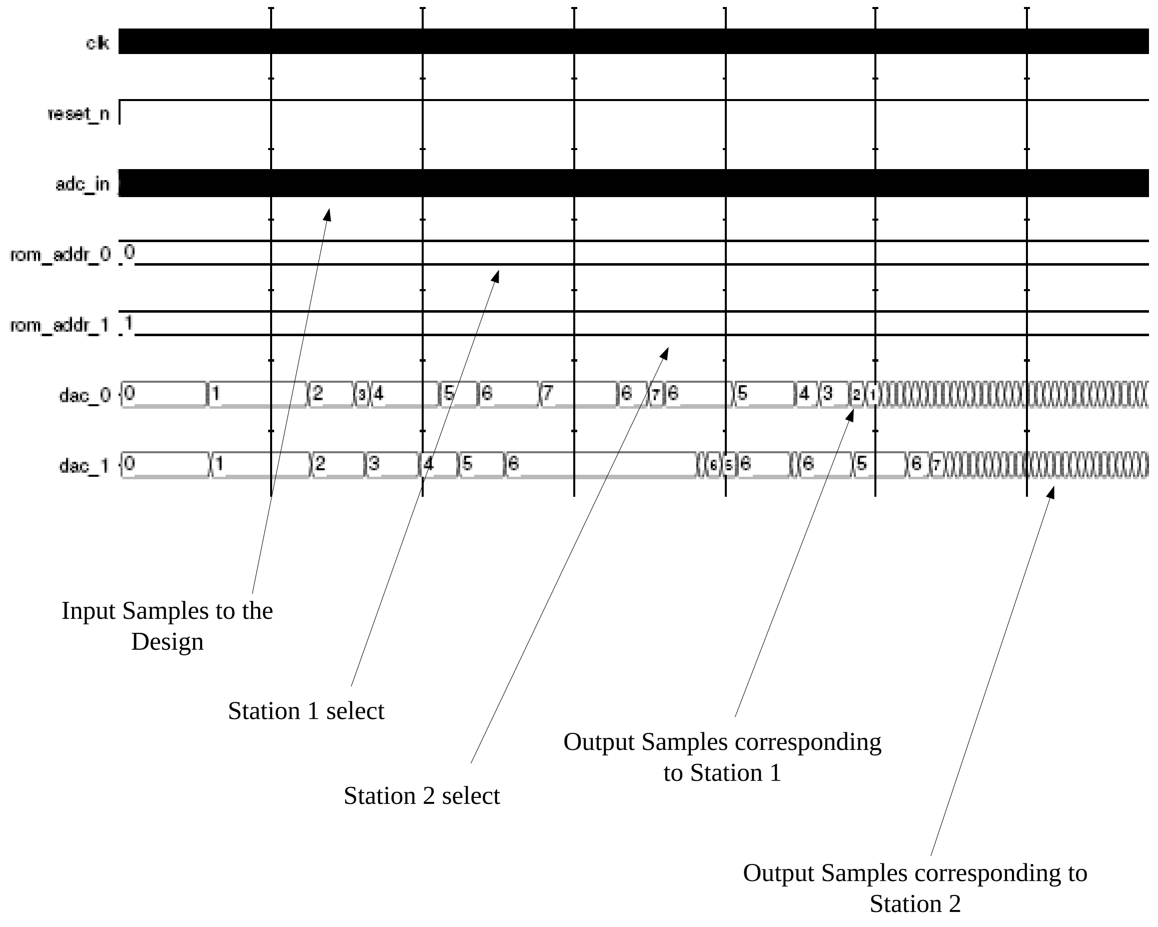


Figure 4.8 Post-Synthesis Simulation Output of Top Level Design

The previously mentioned text files ‘demod_station1.txt’ and ‘demod_station2.txt’ are analyzed in MATLAB to determine if the demodulation was successful. The demodulated signals corresponding to station 1 (Post-Synthesis Demodulated Signal 1) and station 2 (Post-Synthesis Demodulated Signal 2) are shown in Figures 4.9 and 4.10 respectively. The power spectral density plots of Post-Synthesis Demodulated Signal 1 and Post-Synthesis Demodulated Signal 2 are shown in Figures 4.11 and 4.12 respectively.

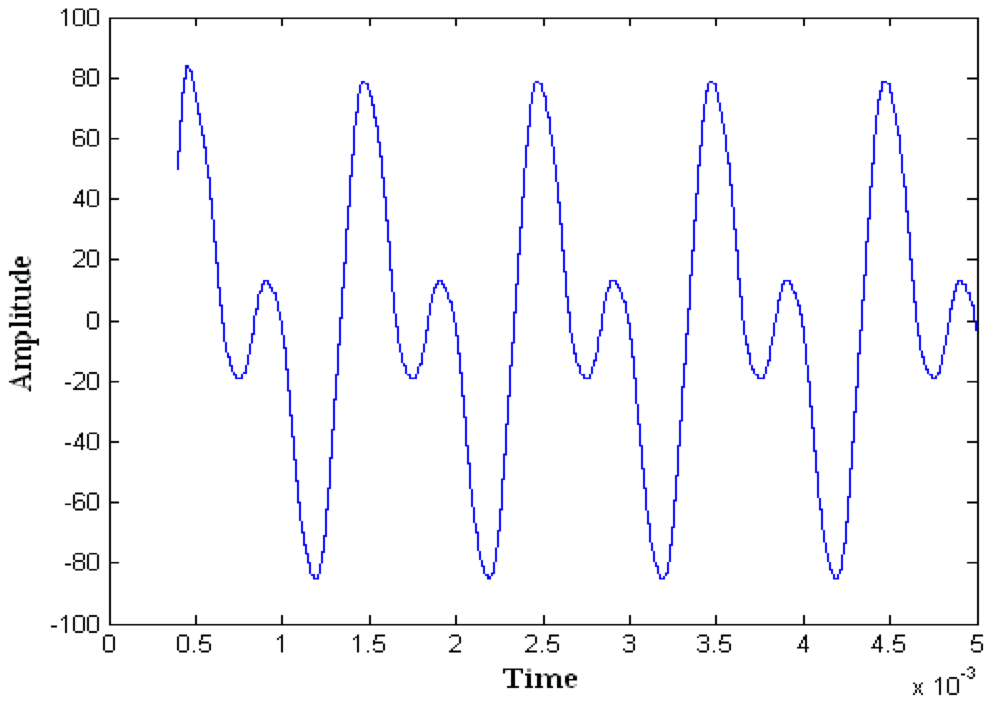


Figure 4.9 Post-Synthesis Demodulated Signal 1

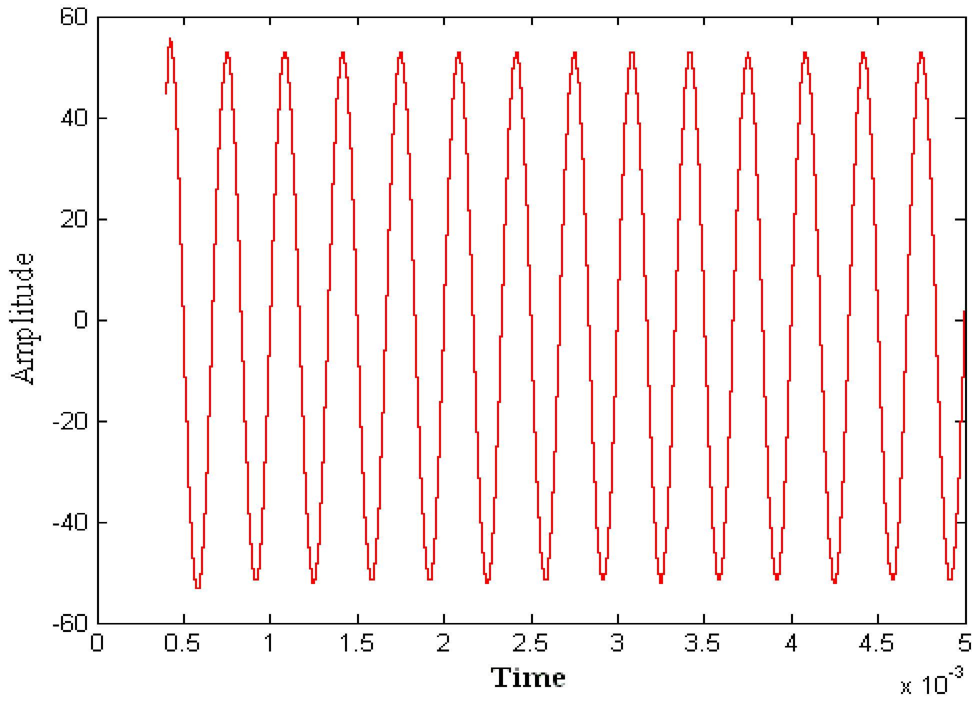


Figure 4.10 Post-Synthesis Demodulated Signal 2

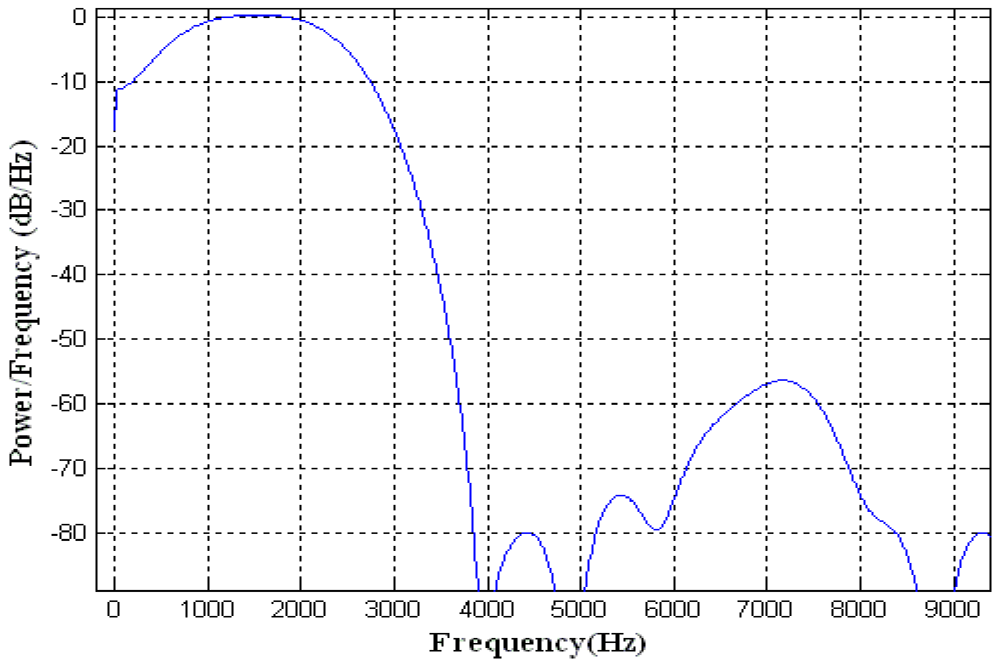


Figure 4.11 Power Spectral Density Plot of Post-Synthesis Demodulated Signal 1

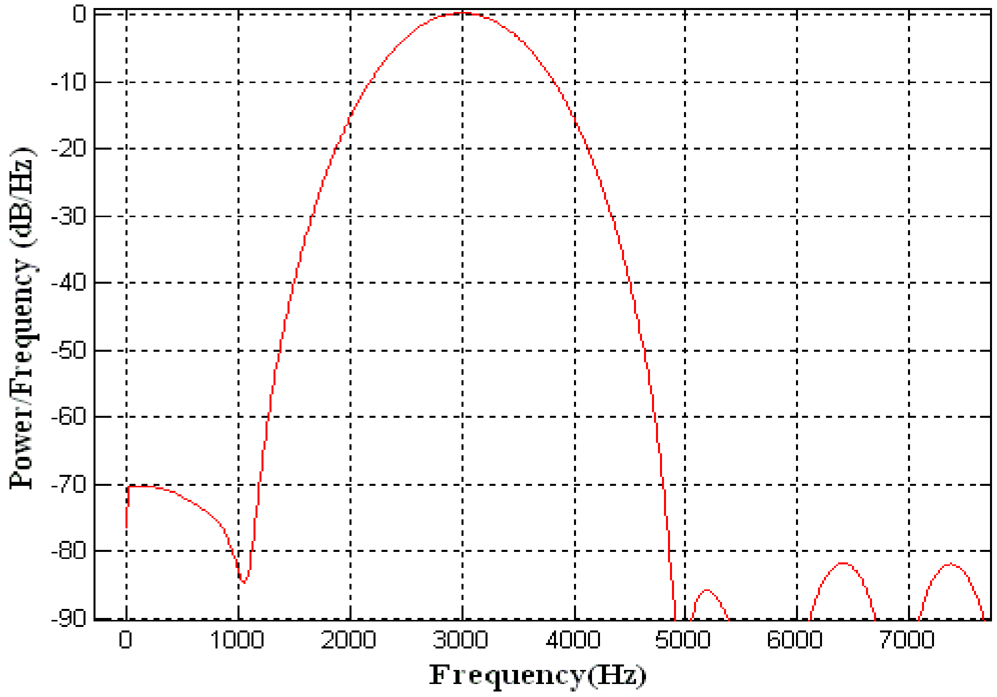


Figure 4.12 Power Spectral Density Plot of Post-Synthesis Demodulated Signal 2

Comparing Figures 4.1, 4.2, 4.3 and 4.4 with Figures 4.9, 4.10, 4.11 and 4.12 respectively it can be seen that both the modulating signals and the demodulated signals are identical. This proves that the receiver was able to successfully demodulate both stations.

4.2 Post-Implementation Simulation Testing and Validation

Post-Synthesis simulation does not take into account the gate propagation delays and propagation delays of other digital logic resources of the circuit. Post-Implementation simulation takes into account gate propagation delays and other logic resource propagation delays and is required to ensure that these delays do not cause the design to malfunction.

The same testbench and input test vectors that were used for Post-Synthesis simulation were used for Post-Implementation simulation. The demodulated signals corresponding to the frequencies of 590 KHz (Post-Implementation Demodulated Signal 1) and 600 KHz (Post-Implementation Demodulated Signal 2) stations are shown in Figures 4.13 and 4.14 respectively. The power spectral density plots of Post-Implementation Demodulated Signal 1 and Post-Implementation Demodulated Signal 2 are shown in Figures 4.15 and 4.16 respectively.

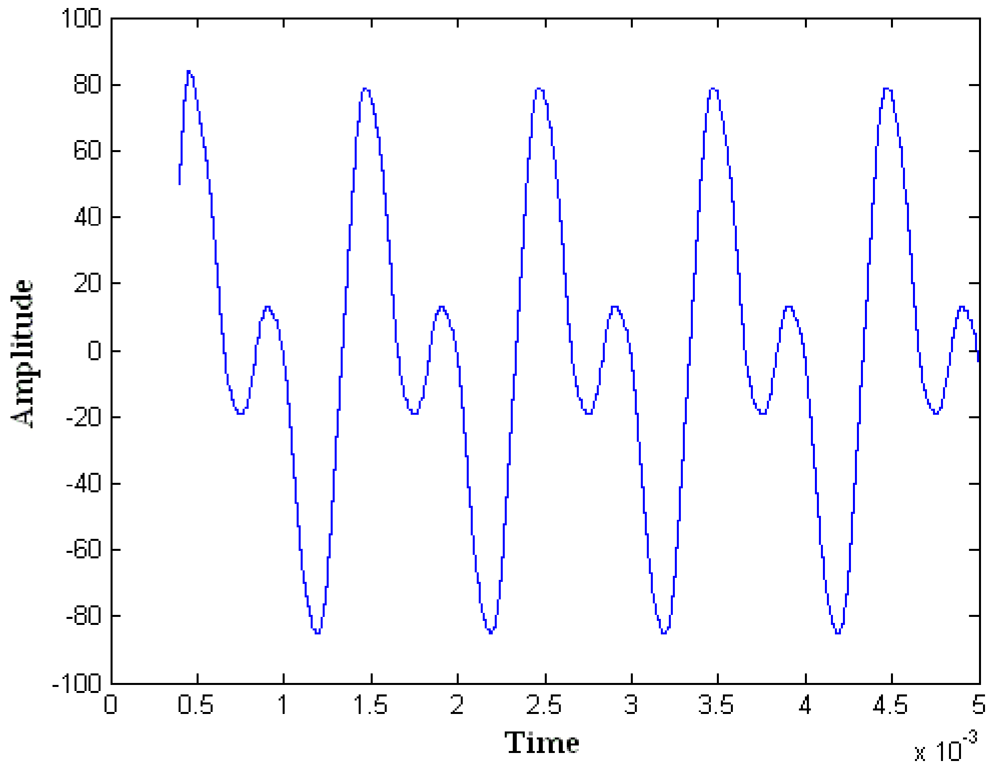


Figure 4.13 Post-Implementation Demodulated Signal 1

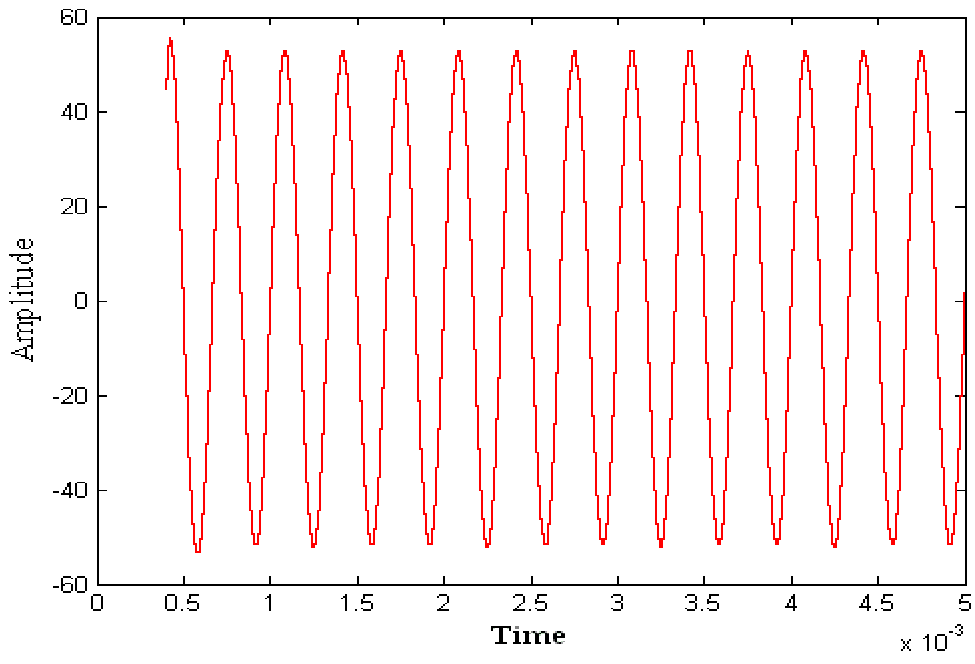


Figure 4.14 Post-Implementation Demodulated Signal 2

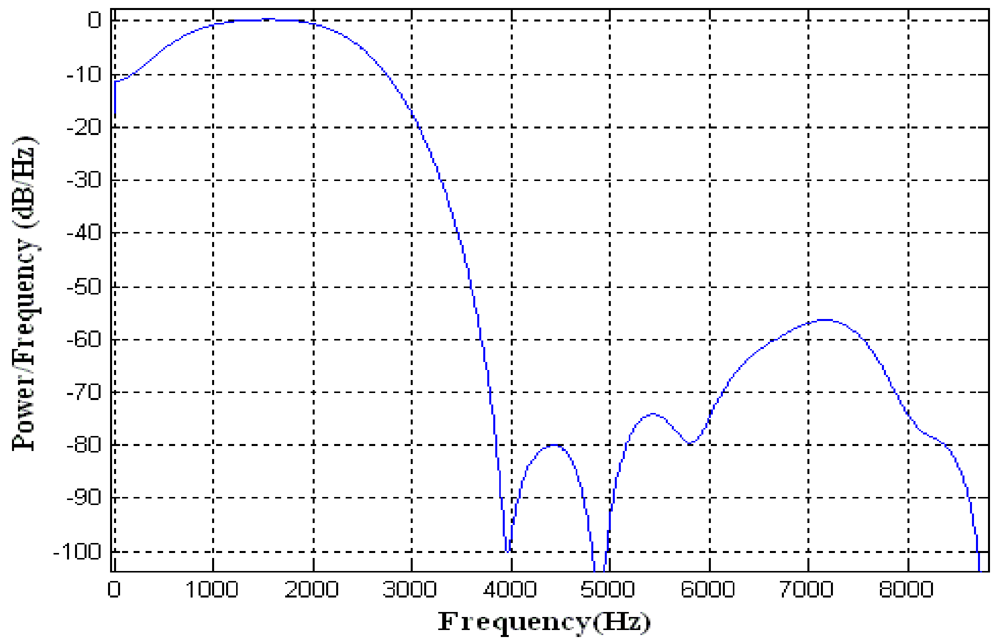


Figure 4.15 Power Spectral Density Plot of Post-Implementation Demodulated Signal 1

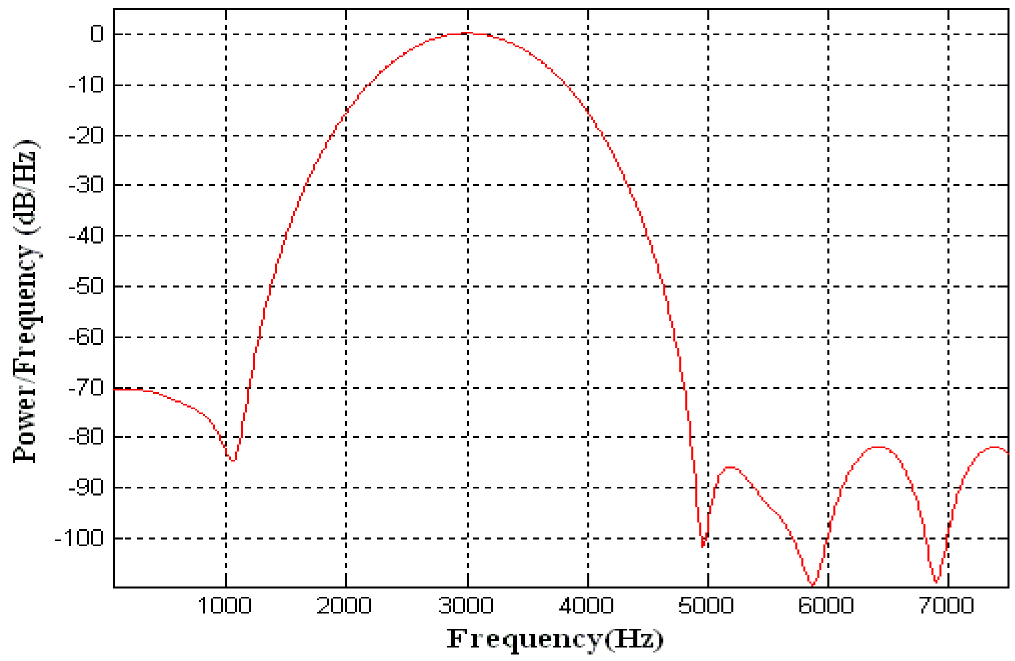


Figure 4.16 Power Spectral Density Plot of Post-Implementation Demodulated Signal 2

From Figures 4.1, 4.2, 4.3, 4.4, 4.13, 4.14, 4.15 and 4.16 it can be seen that both the modulating signal and the demodulating signal are identical. This shows that the demodulation has been successful.

4.3 Maximum Performance of Multiple Channel AM TDM SDR Receiver Architecture

From synthesis reports obtained from the Xilinx ISE 8.2i CAD tools, it was observed that the maximum frequency at which the 4 channel SDR receiver can operate is 84.741 MHz (Min Period : 11.801 ns). Assuming that the 5 MHz input sample rate is enough for AM and also assuming that the maximum frequency at which the design can operate will not decrease significantly as more channels are added to the multi-channel SDR receiver, the maximum number of channels that can be supported on a Virtex-II XC2VP30-51152 FPGA would be 16 channels since 5 MHz/channel * 16 channels = 80 MHz.

Chapter 5

Hardware Resource Requirements Comparison for Multiple Channel Operation

The hardware resource requirements of the proposed TDM SDR architecture for multiple channel operation will be compared to those of conventional architecture structured for multiple channel operation. To do this a single coherent single station conventional architecture module and design was developed. The conventional architecture module and design is simply reciprocated n times to receive n stations and is called the parallel receiver architecture. The parallel architecture is similar to the TDM architecture except that it uses dedicated hardware components to process each channel. The block diagram view of a coherent single station module for a multiple station parallel architecture is shown in Figure 5.1. This single station architecture was synthesized and Place and Route Implemented onto a Virtex-II Pro XC2VP100-5FF1696 FPGA. If it is needed to tune into n stations simultaneously, n single station modules would be used. So as the number of stations to be tuned into simultaneously increases from 1 to n, the hardware resource count would roughly increase from 1 to n.

The TDM architecture is first used to construct 1 station, 2 station through 4 station coherent receivers. The single station module of Figure 5.1 is then used to conventionally configure a 1 station, 2 station through 4 station radio receiver configuration. Corresponding TDM and conventional designs are then individually synthesized and implemented onto a Virtex-II pro XC2VP100-5FF1696 FPGA. The hardware resource utilization of these receivers based on the conventional and TDM architectures are compared. The comparison is based on FPGA chip logic resources of slices, 4-input LUTS and flip-flops. The results are summarized in Figures 5.2, 5.3 and 5.4. In Figures 5.2, 5.3 and 5.4 , “2 Channels” correspond to “1 Station”.

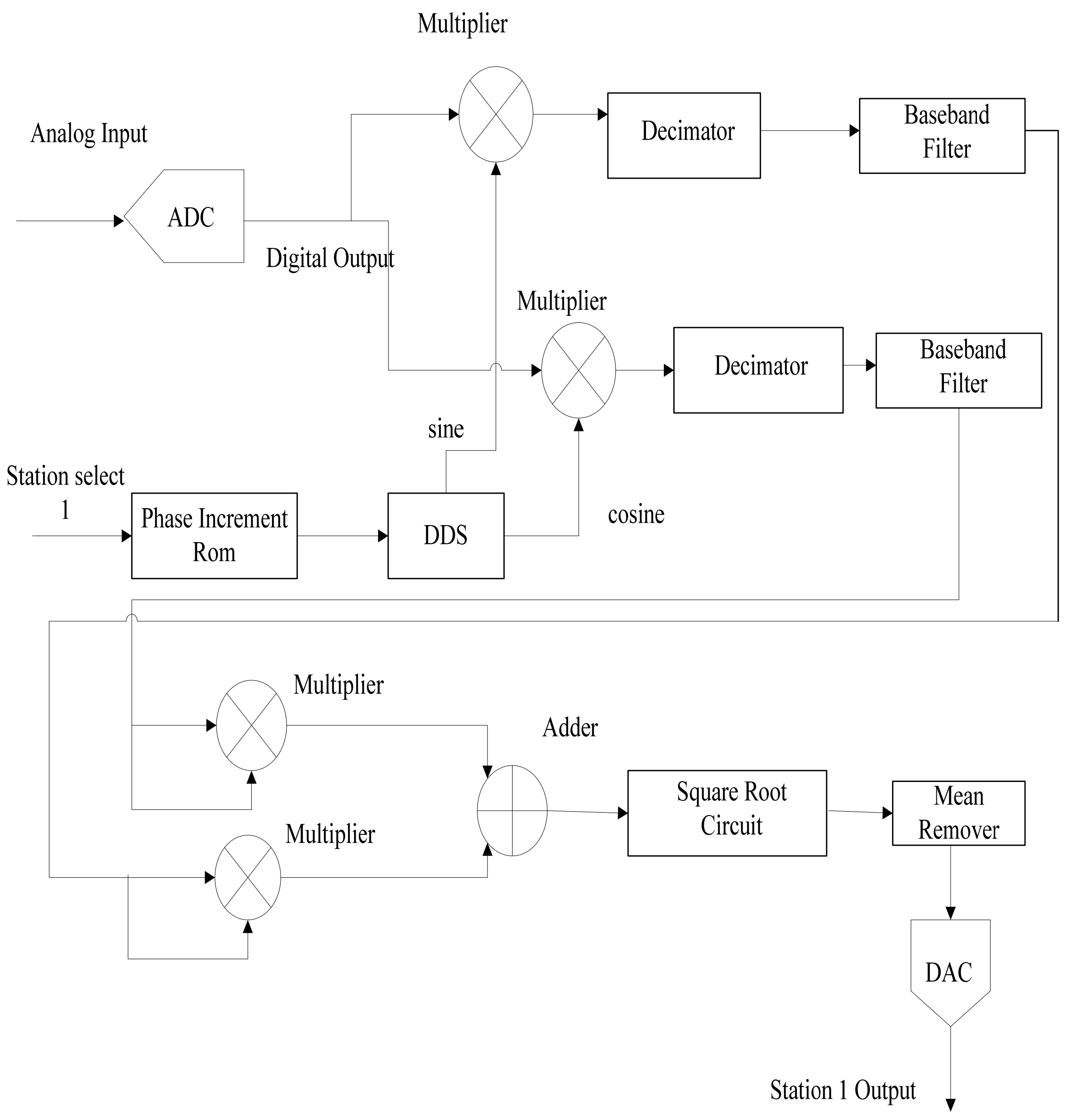


Figure 5.1 Block Diagram of a Coherent Single Station (2 Channels) Module for a Multiple Module Parallel Architecture Implementation of a Multiple Station Receiver.

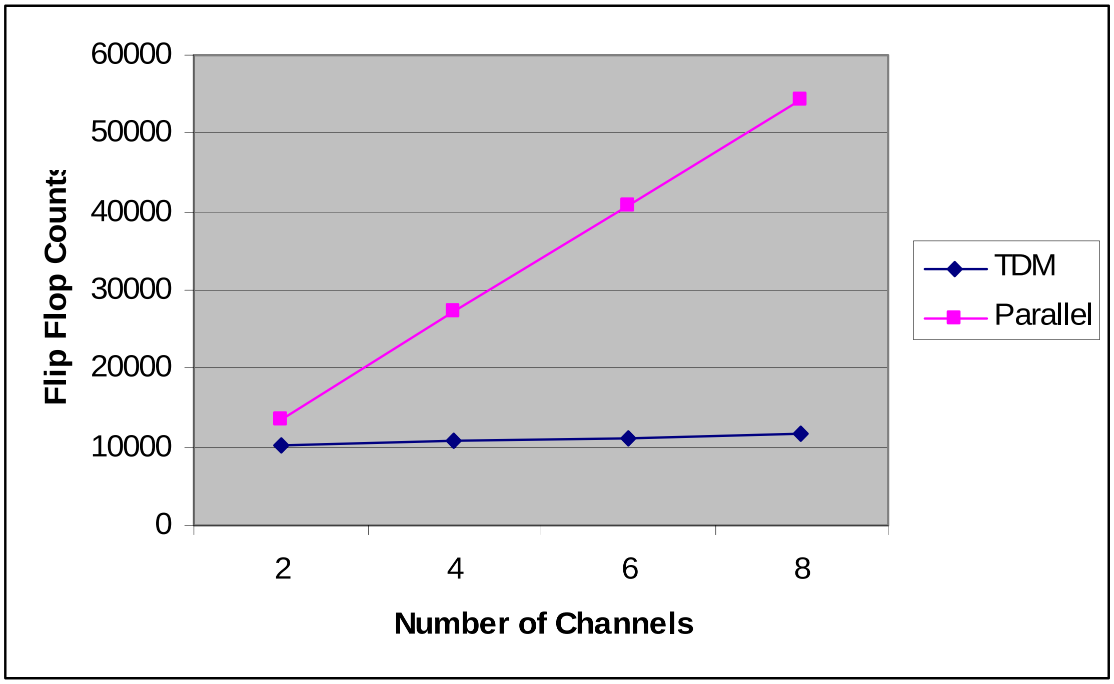


Figure 5.2 Comparison of Flip Flop Utilization

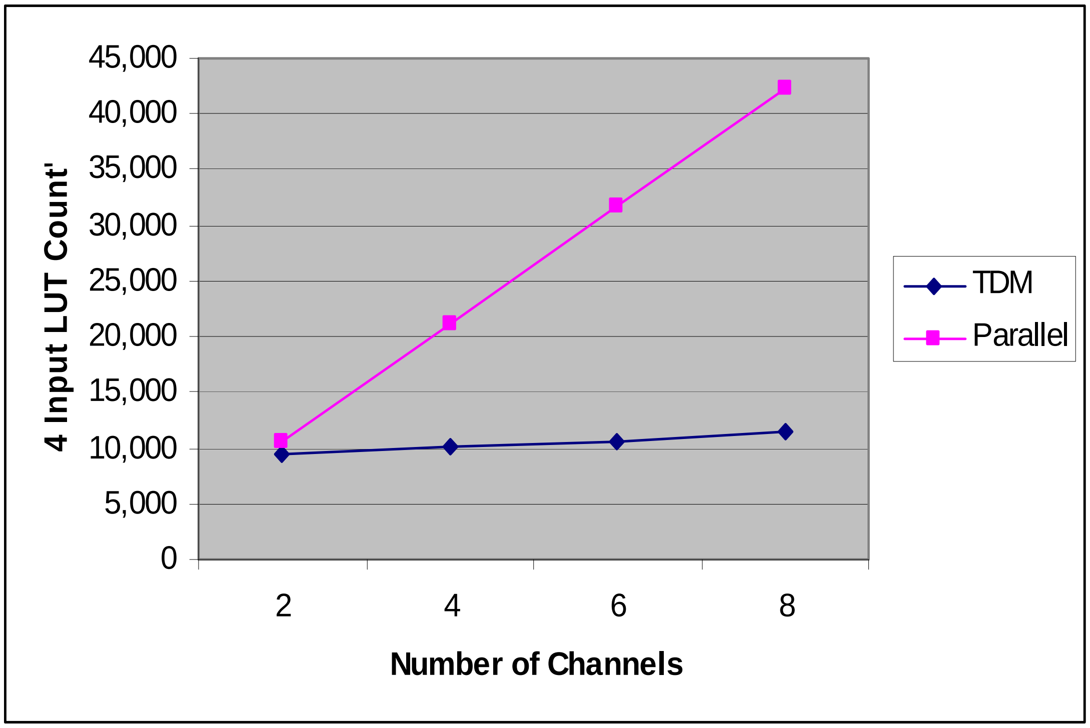


Figure 5.3 Comparison of 4 Input LUT Utilization

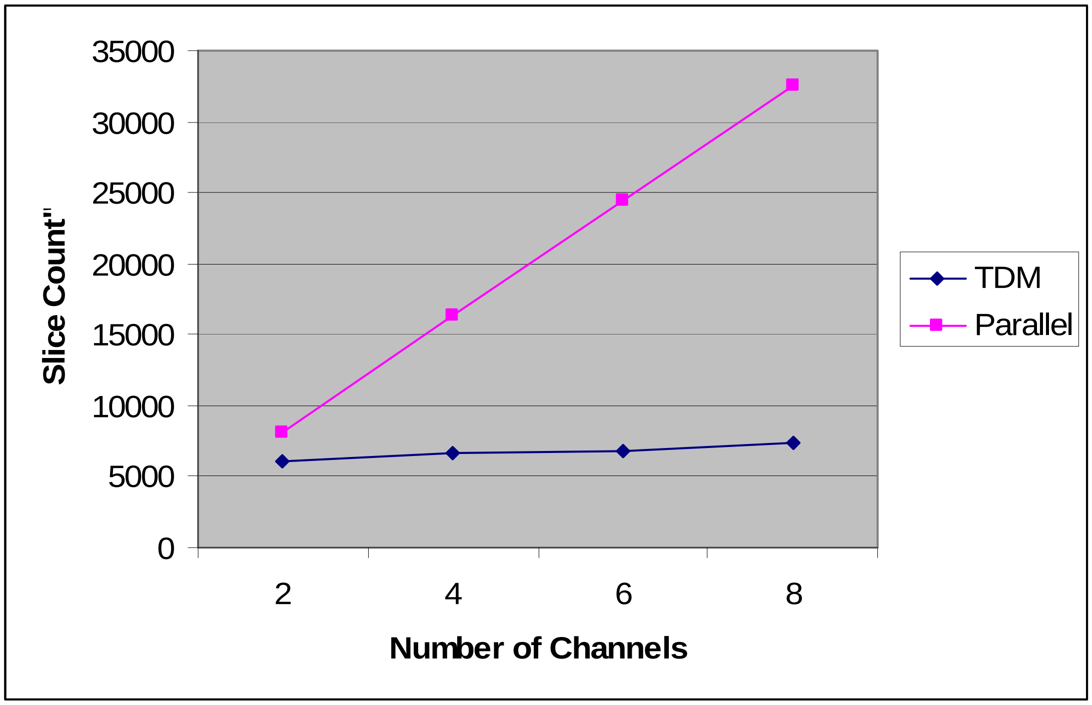


Figure 5.4 Comparison of Post-Place and Route Implementation Slice Count

From Figures 5.2, 5.3 and 5.4 it can be seen that as the number of channels increase the logic resource utilization requirements of the parallel architecture increases linearly in a scalable manner, where as in comparison, for the same number of channels, the logic resource utilization of the TDM architecture increases only by a very small amount. These plots demonstrate the logic resource utilization efficiency of the TDM architecture when implemented to FPGA technology.

Chapter 6

Experimental Hardware Prototyping, Testing and Validation

Hardware Prototyping of the Design is carried out to experimentally validate its functionality. The prototyped design is made to receive two actual AM stations. A hardware prototyping board developed by Nallatech was utilized [22]. The board features two user-programmable Xilinx FPGAs, an XC2VP30-5FF1152 and XC2V80-4CS144. The larger FPGA, the XC2VP30-5FF1152 (Main User FGPA) is used to emulate the radio receiver architecture and design and the smaller FPGA, the XC2V80-4CS144 (Clock FPGA) is used for clock management. The board also contains two ADCs and DACs which can be used to interface the design with the analog world. Only one of the two ADCs is used. A picture of the Nallatech prototyping board is shown in Figure 6.1.

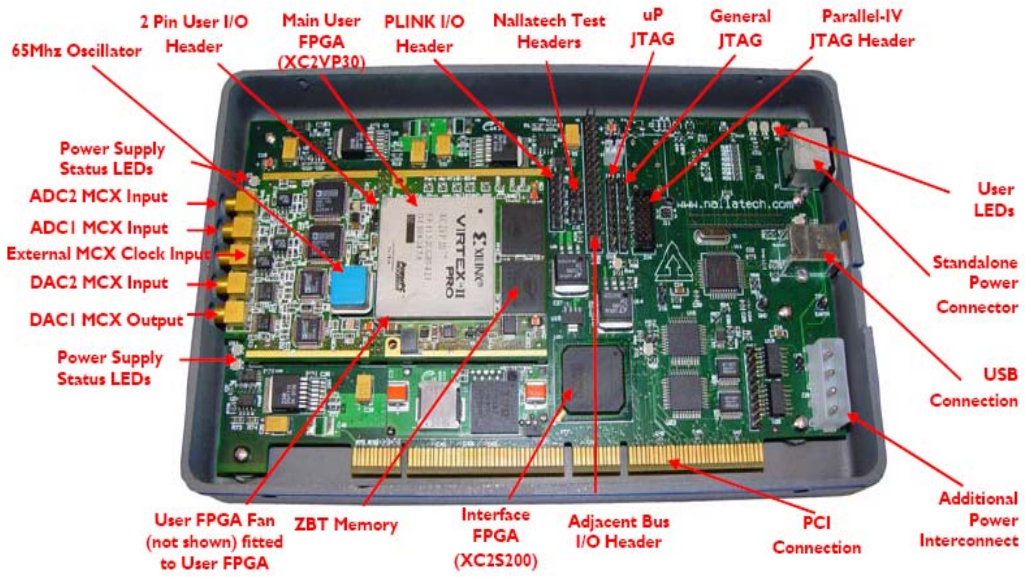


Figure 6.1 Picture of Nallatech Prototyping Board

6.1 Experimental Setup

The board features two software programmable oscillators and a fixed 105 MHz crystal oscillator. One of the two software programmable oscillators is used as the clock source for the setup. The frequency of the software programmable oscillator can be varied from the PC to which the board is connected by using the FUSE software provided by the board manufacturer. The clock output from the software programmable oscillator is connected to one of the global clock pins of the main user FPGA. The clock signal from the software programmable oscillator is not used to clock any logic inside the main user FPGA and is forwarded directly to the clock FPGA. In the clock FPGA, the forwarded clock is connected to the inputs of LVPECL (Low Voltage Positive Emitter Coupled Logic) Output Buffers. The LVPECL Output Buffers are utilized to generate differential clock signals from the single-ended clock signal. These differential clock signals are used to clock the ADCs and DACs. The clock FPGA also feeds back the forwarded clock that it received from the main user FPGA back to the main user FPGA. It is ensured by the board manufacturers that the lengths of the differential clock nets from the clock FPGA to the ADCs, DACs and the length of feedback clock net is the same [22]. This ensures that both these clock nets are matched in skew. This setup which ensures matching clock skews for the ADCs, DACs and the main user FPGA is shown in Figure 6.2.

The feedback clock from the clock FPGA can now be used inside the main user FPGA to clock the design. Before the feedback clock can be used inside the main user FPGA, it is given to an internal clock deskew circuit, which eliminates any internal clock distribution delay. The internal clock deskew circuit uses a Xilinx Digital Clock Manager (DCM) [23] core in deskew configuration. The internal deskew circuit is shown in Figure 6.3. The DCM contains Delay-Locked Loop (DLL) circuit which tries to ensure that the input signal and the feedback signal are locked in phase. The locked signal of DCM goes high when the DCM achieves lock. This signal is used as the reset signal for the design. There are two external reset signals for the main user FPGA both of which can be controlled from the FUSE software. One of these two resets signals can be used to reset the DCM. Before connecting to the reset input of the DCM, the reset signal should be inverted because the reset signal is active low and the DCM expects an active high reset.

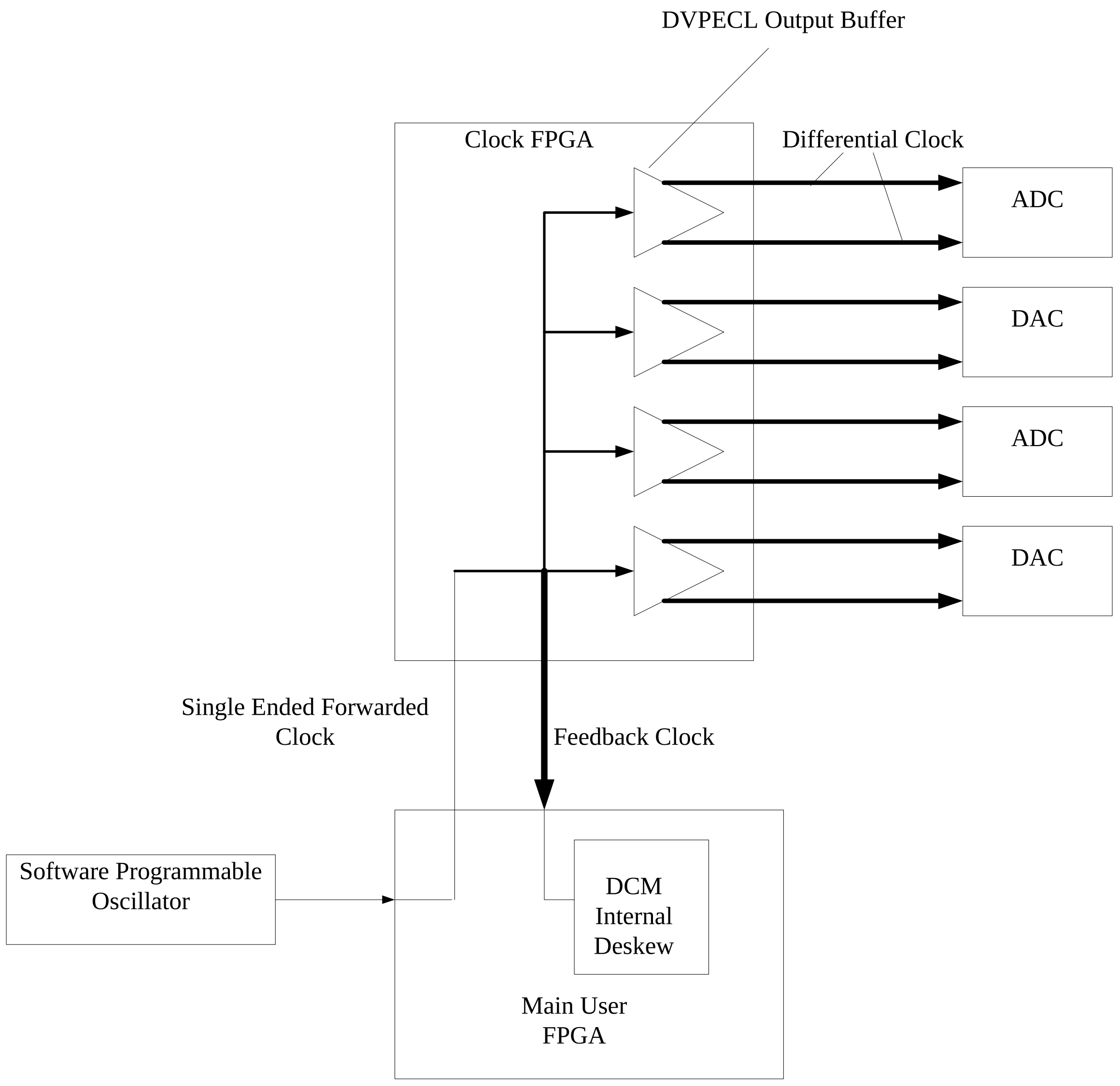


Figure 6.2 Matching Clock Skew Setup.

The suggested maximum input voltage range for the 14 bit ADC is 2V p-p or +/- 1 V. Since only the least significant 12 bits of the ADC are used, the maximum input voltage range now becomes 0.5V p-p or +/- 0.25 V.

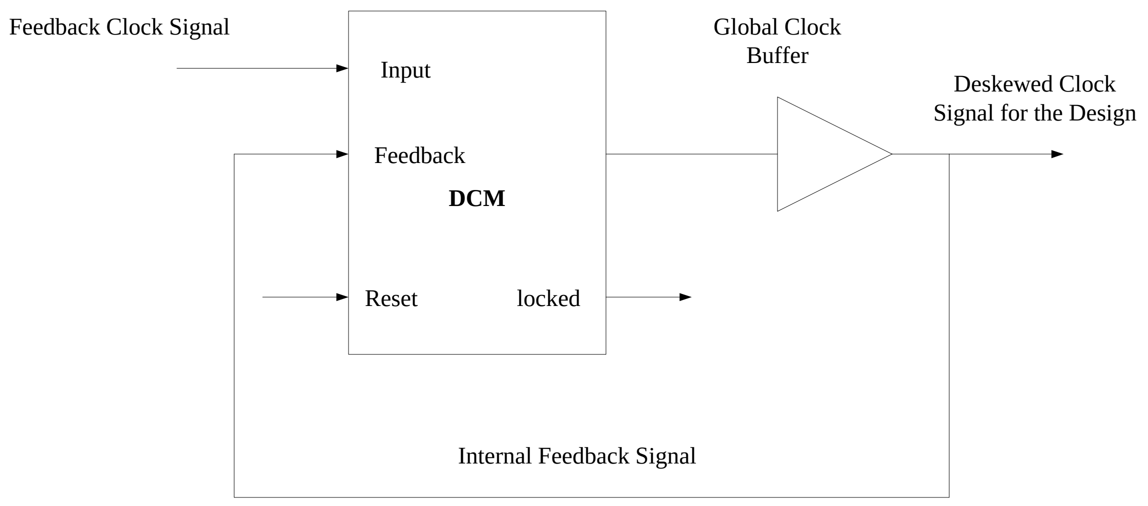
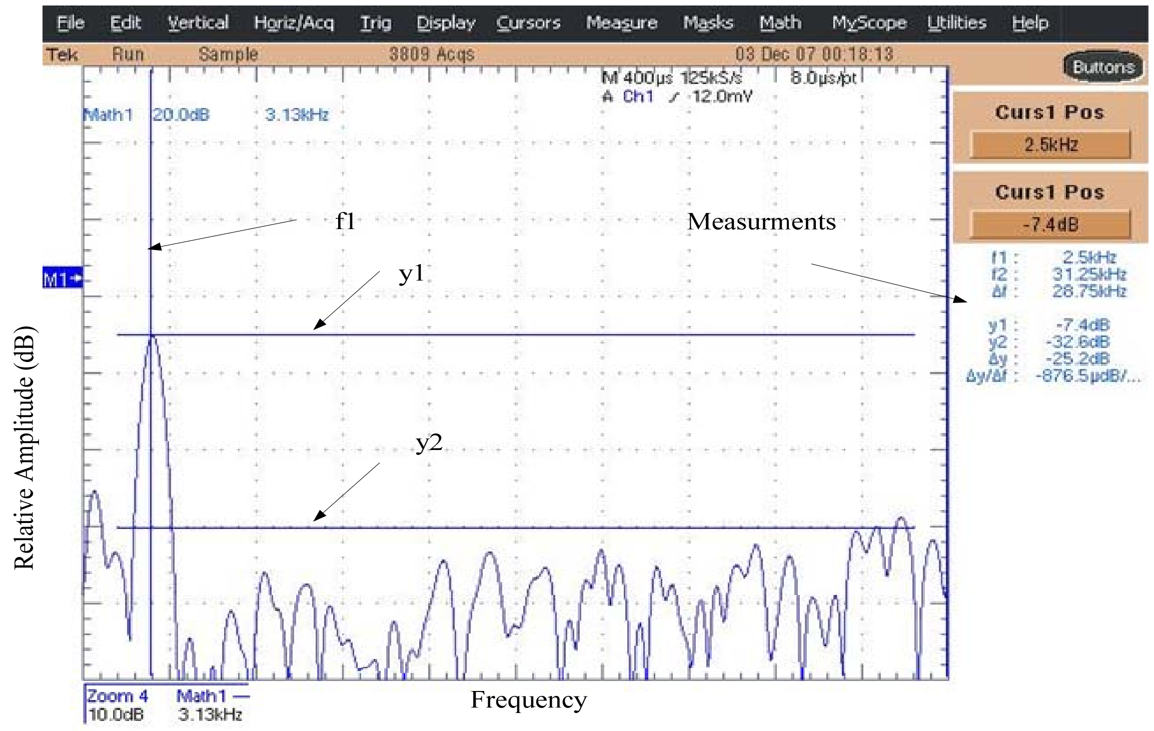


Figure 6.3 Internal Deskew Circuit

6.2 Experimental Testing

Station 1 will be tuned to 590 KHz and station 2 to 1250 KHz (see Figure 3.1 for reference). Two digital function generators (AFG310) are used to locally generate an AM wave which is supplied to the ADC input. First, a 590 KHz AM wave, with a modulation index of 90 % and modulating signal frequency of 2.5 KHz is applied to the ADC input. The spectrum plots of the DAC 1 output which corresponds to station1, and DAC 2 output, which corresponds to station 2, are shown in Figures 6.4 and 6.5 respectively.

From Figure 6.4 it can be seen that the design was successfully able to demodulate the 590 KHz station. Next a 600 KHz AM wave, modulation index of 90 % and a modulating signal frequency of 2.5 KHz is applied to the ADC input. The spectrum plots of the DAC 1 output and DAC 2 output are shown in Figures 6.6 and 6.7 respectively.



Tek Run Sample 3809 Acqs 03 Dec 07 00:18:13

M 400µs 125kS/s 8.0µs/pt
A Ch1 ⁄ 12.0mV

Math1 20.0dB 3.13kHz

Curs1 Pos 2.5kHz

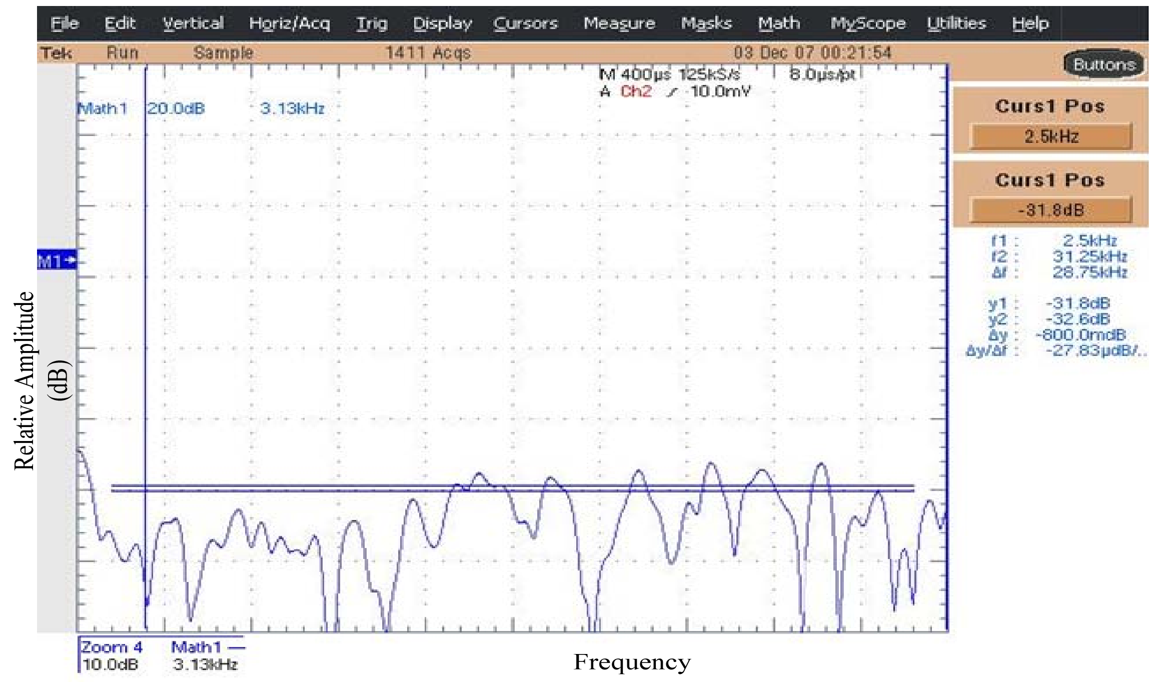
Curs1 Pos -7.4dB

```
f1 :     2.5kHz
f2 :     31.25kHz
Δf :     28.75kHz

y1 :     -7.4dB
y2 :     -32.6dB
Δy :     -25.2dB
Δy/Δf :  -876.5µdB/...
```

Zoom 4 10.0dB Math1 — 3.13kHz

Figure 6.4 Spectrum Plot of DAC 1 Output (590 KHz AM Input)



Tek Run Sample 1411 Acqs 03 Dec 07 00:21:54

M 400µs 125kS/s 8.0µs/pt
A Ch2 ⁄ 10.0mV

Math1 20.0dB 3.13kHz

Curs1 Pos 2.5kHz

Curs1 Pos -31.8dB

```
f1 :     2.5kHz
f2 :     31.25kHz
Δf :     28.75kHz

y1 :     -31.8dB
y2 :     -32.6dB
Δy :     -800.0mdB
Δy/Δf :  -27.83µdB/...
```

Zoom 4 10.0dB Math1 — 3.13kHz

Figure 6.5 Spectrum Plot of DAC 2 Output (590 KHz AM Input)

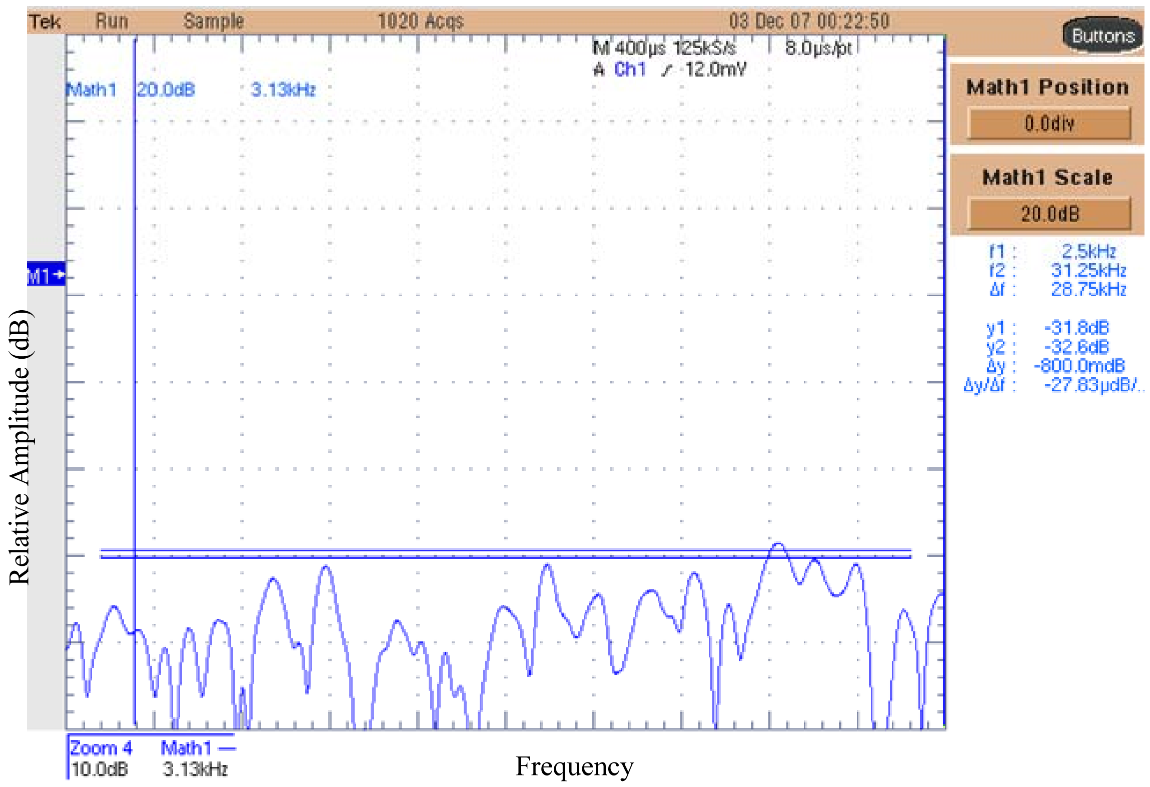


Figure 6.6 Spectrum Plot of DAC 1 Output (600 KHz AM Input)

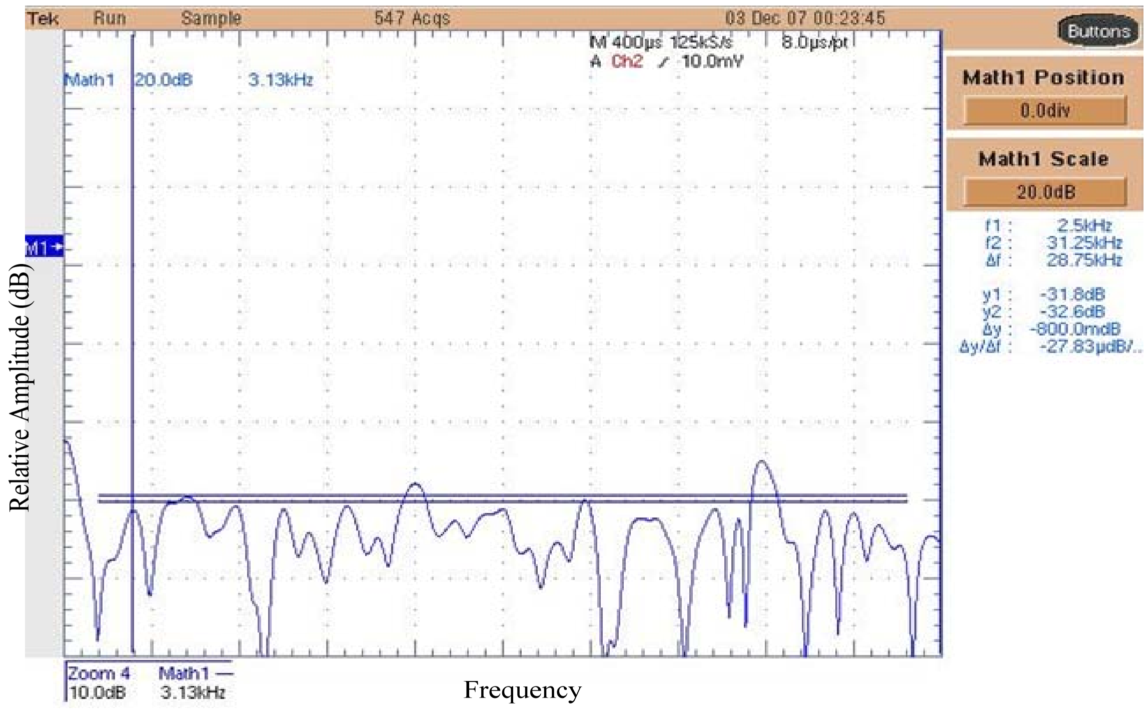


Figure 6.7 Spectrum Plot of DAC 2 Output (600 KHz AM Input)

From Figure 6.6 it can be seen that the architecture and design was successful in eliminating adjacent channel interference for station 1.

Similarly the spectrum plots of DAC 1 and DAC 2 when a 1250 KHz AM wave, with a modulation index 100 % and modulating signal frequency 3 KHz is applied to the input of the ADC are shown in Figures 6.8 and 6.9 respectively. The spectrum plots of DAC 1 and DAC 2, when a 1240 KHz AM wave, with a modulation index of 100 % and modulating signal frequency of 3 KHz is applied to the input of the ADC, are shown in Figures 6.10 and 6.11 respectively.

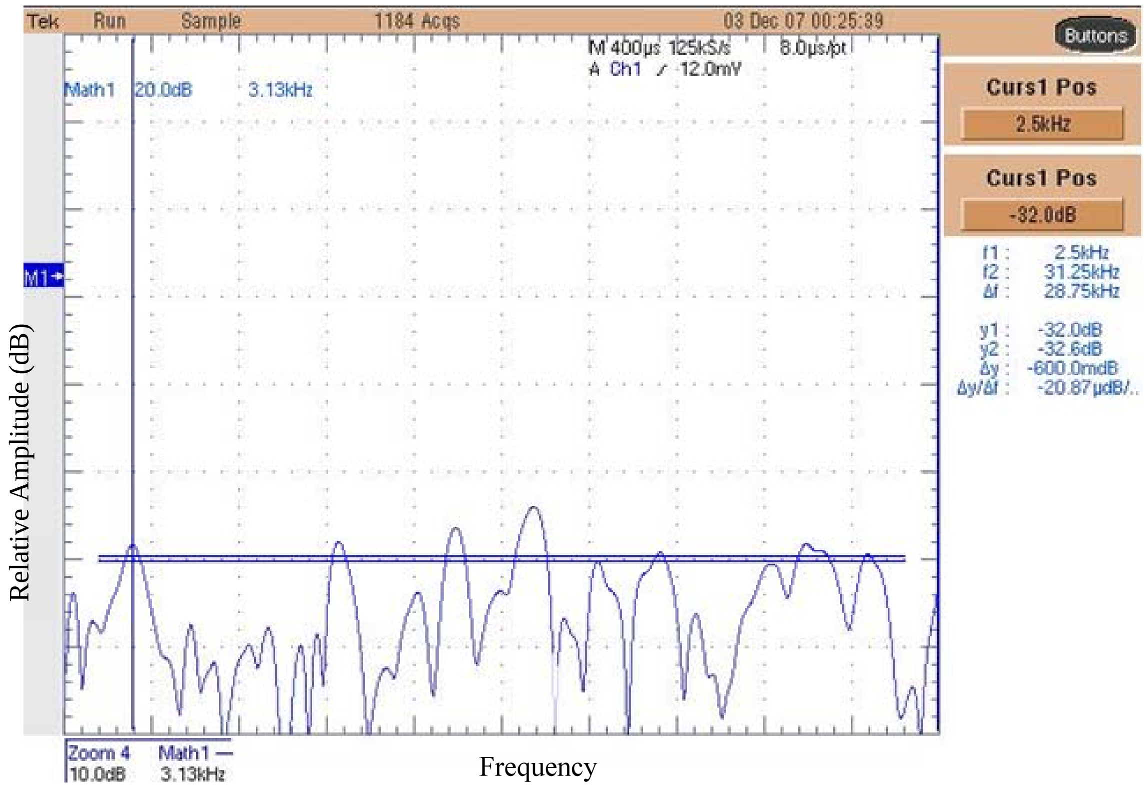


Figure 6.8 Spectrum Plot of DAC 1 Output (1250 KHz AM Input)

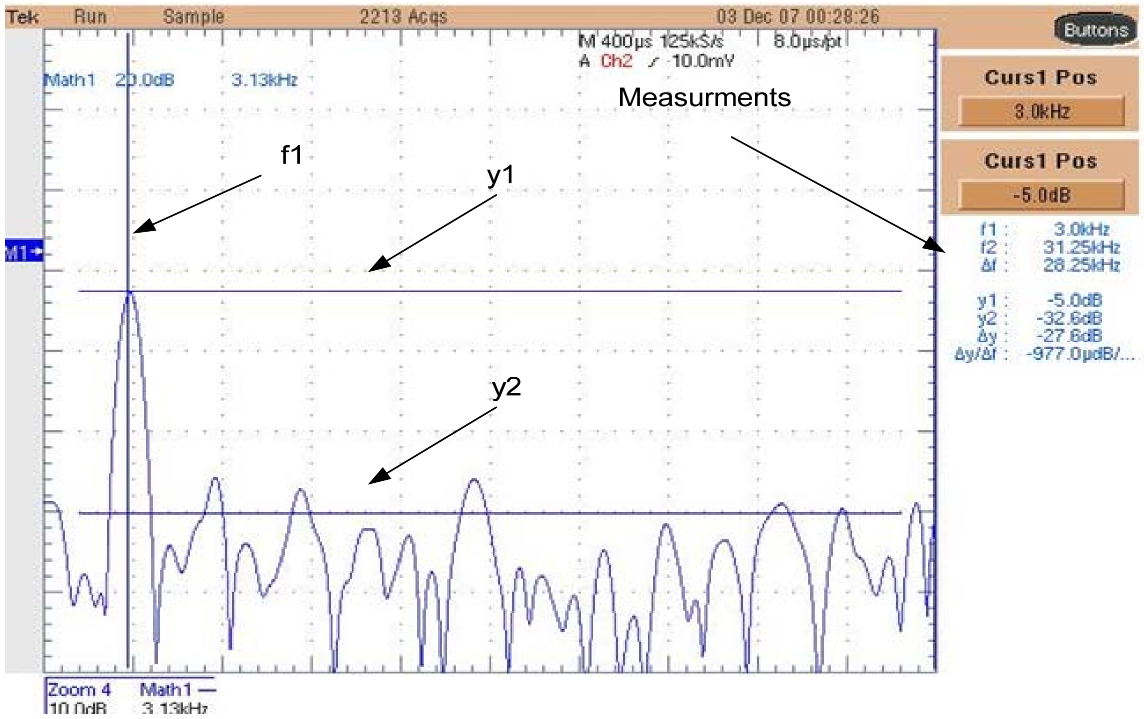


Figure 6.9 Spectrum Plot of DAC 2 Output (1250 KHz AM Input)

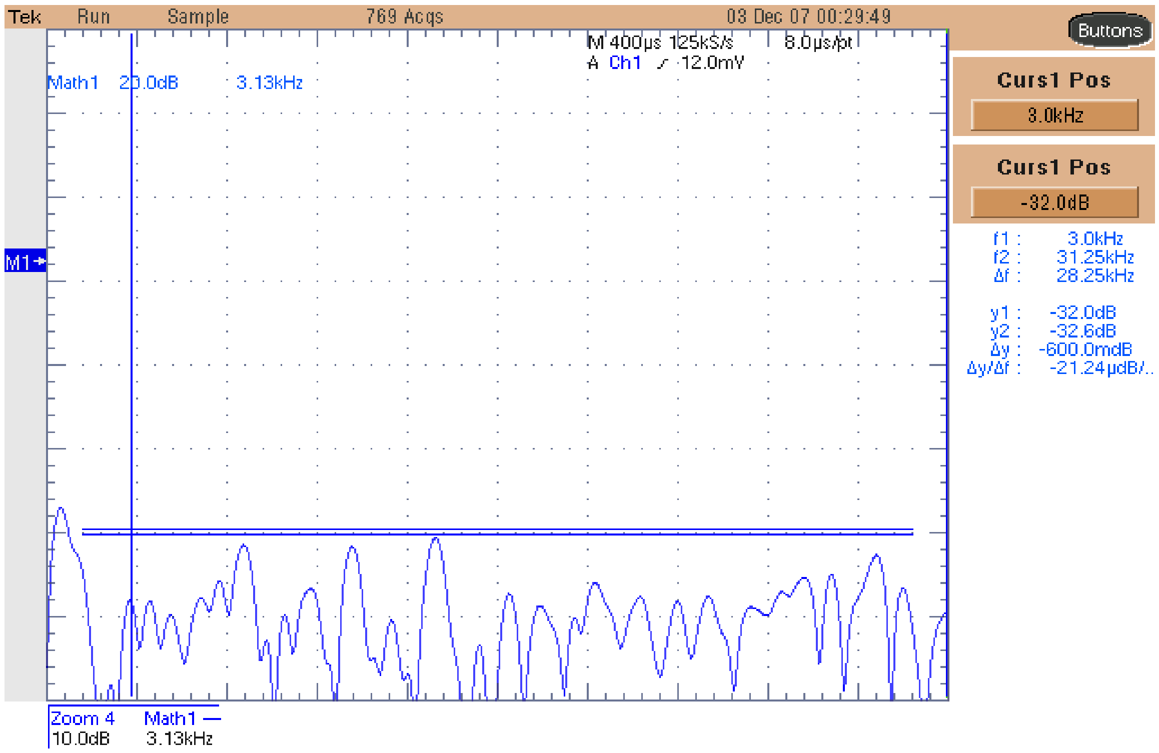


Figure 6.10 Spectrum Plot of DAC 1 Output (1240 KHz AM Input)

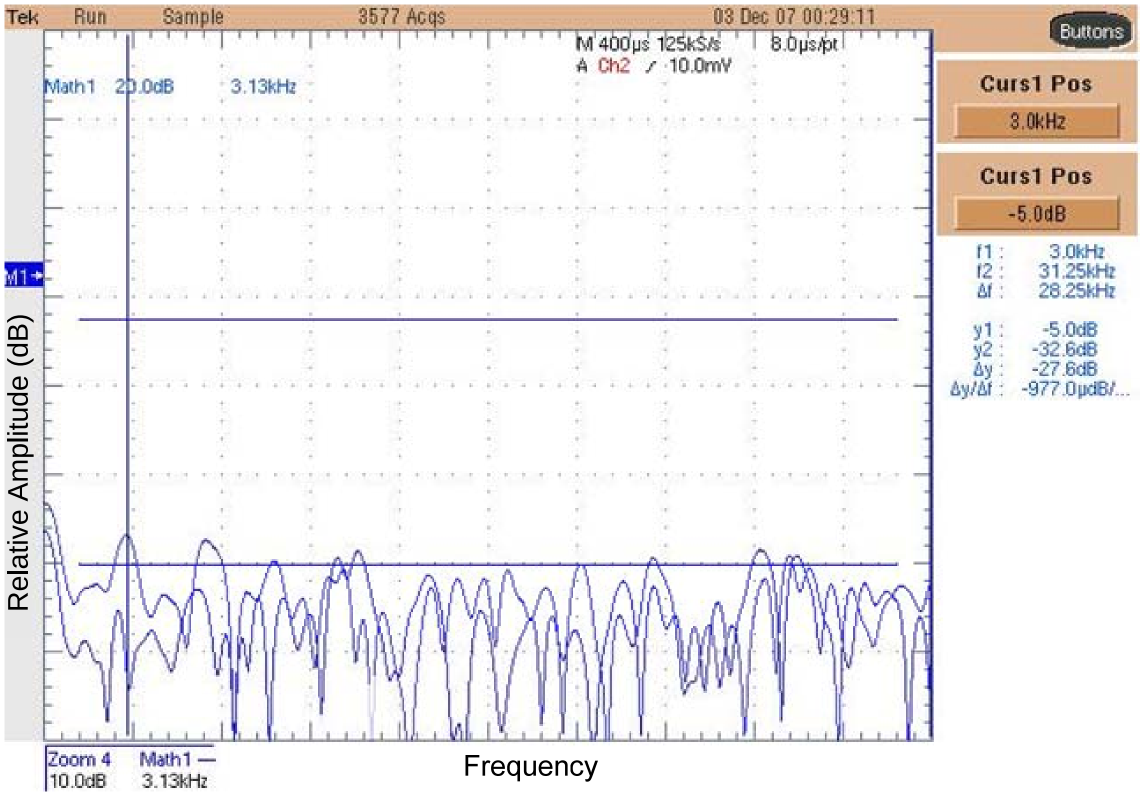


Figure 6.11 Spectrum Plot of DAC 2 Output (1240 KHz AM Input)

From Figures 6.9 and 6.11 it can be seen that the architecture and design was successfully able to demodulate the station 2 signal and also avoid any adjacent channel interference for station 2.

This validates that the architecture and design is able to demodulate both the stations simultaneously.

Chapter 7

Conclusions

The main objective of this thesis is to develop a hardware efficient Coherent AM Multi-channel TDM SDR architecture and design. The developed architecture and design was captured using the Verilog HDL. Synthesis and Implementation of the captured architecture were done using Xilinx ISE 8.2i CAD software packages. The architecture was functionally and performance validated using Post-Synthesis and Post-Implementation HDL simulations. Experimental prototype testing of the architecture was carried out on a Nallatech FPGA board which featured a Xilinx Virtex-II Pro FPGA. All Post-Synthesis and Post-Implementation HDL simulations and experimental hardware testing of the hardware prototype validated that the architecture and design functioned and performed correctly.

The logic resource utilization of the multi-channel TDM architecture was compared to that of a conventional multi-channel architecture. For a two station (4 Channel) implementation it was found that the TDM architecture was able to achieve logic utilization savings of more than 100 %. Also, as the number of channels increase the hardware resource utilization of the TDM architecture minimally increases as compared to the linear scalable increase in hardware resources required by a conventional architecture.

Appendix A

Verilog Code for Multi-Channel Coherent AM TDM SDR Architecture

1. Top Level Module with Internal DeSkew and Reset Logic

```
`default_nettype none
module device_top(
            input           CLKA,                       // input software programmable dime clock
                            CLK1_FB,                    // feedback clock from the clock fpga
            input           RESET1,                     // system reset (active-low)
            input           [13:0]    adc1_d,           // ADC1 OUTPUT
            output          [13:0]    dac1_d,           // DAC1 INPUT
                                      dac2_d,           // DAC2 INPUT
            output                    GEN_CLKA,         // clock for ADC and DAC
            output                    CONFIG_DONE,      // configuration done
            output                    dac1_div0,        // DAC control signals
                                      dac1_div1,
                                      dac1_mod0,
                                      dac1_mod1,
                                      dac1_reset,
                                      dac2_div0,
                                      dac2_div1,
                                      dac2_mod0,
                                      dac2_mod1,
                                      dac2_reset
                                      );

wire        reset_design,
            clk,
            RST_IN,
            LOCKED_OUT,
            NO_CONNECTION;

wire        [2:0]     rom_addr_0,       // Station Select Signals
                      rom_addr_1;

assign CONFIG_DONE = 0;
assign dac1_mod0 = 0;
assign dac1_mod1 = 0;
assign dac1_div0 = 0;
assign dac1_div1 = 1;
assign dac2_mod0 = 0;
assign dac2_mod1 = 0;
assign dac2_div0 = 0;
assign dac2_div1 = 1;
assign dac1_reset = 0;
assign dac2_reset = 0;
assign RST_IN = ~ RESET1;
OBUF CLKA_OUT (.I(CLKA),
            .O(GEN_CLKA));
internal_deskew INTERNAL_DESKEW (  //Internal Deskew DCM
   .CLKIN_IN(CLK1_FB),
   .RST_IN(RST_IN),
   .CLKIN_IBUFG_OUT(NO_CONNECTION),
   .CLK0_OUT(clk),
   .LOCKED_OUT(LOCKED_OUT)
   );
assign reset_design = LOCKED_OUT;
assign rom_addr_0 = 3'd0;
assign rom_addr_1 = 3'd2;
top TOP (
                            .clk(clk),
                            .reset_n(reset_design),
                            .adc_in(adc1_d),
                            .rom_addr_0(rom_addr_0),
                            .rom_addr_1(rom_addr_1),
                            .dac_0(dac1_d),
                            .dac_1(dac2_d)
);
endmodule
```

2. Internal DeSkew Logic

```
////////////////////////////////////////////////////////////////////////////////
// Copyright (c) 1995-2006 Xilinx, Inc.  All rights reserved.
////////////////////////////////////////////////////////////////////////////////
//   ____  ____
//  /   /\/   /
// /___/  \  /    Vendor: Xilinx
// \   \   \/     Version : 8.2.03i
//  \   \         Application : xaw2verilog
//  /   /         Filename : internal_deskew.v
// /___/   /\     Timestamp : 11/20/2007 14:56:12
// \   \  /  \
//  \___\/\___\
//
//Command: xaw2verilog -intstyle F:/thesis1/internal_deskew.xaw -st internal_deskew.v
//Design Name: internal_deskew
//Device: xc2vp30-5ff1152
//
// Module internal_deskew
// Generated by Xilinx Architecture Wizard
// Written for synthesis tool: XST
`timescale 1ns / 1ps

module internal_deskew(CLKIN_IN,
                       RST_IN,
                       CLKIN_IBUFG_OUT,
                       CLK0_OUT,
                       LOCKED_OUT);

    input CLKIN_IN;
    input RST_IN;
   output CLKIN_IBUFG_OUT;
   output CLK0_OUT;
   output LOCKED_OUT;

   wire CLKFB_IN;
   wire CLKIN_IBUFG;
   wire CLK0_BUF;
   wire GND1;

   assign GND1 = 0;
   assign CLKIN_IBUFG_OUT = CLKIN_IBUFG;
   assign CLK0_OUT = CLKFB_IN;
   IBUFG CLKIN_IBUFG_INST (.I(CLKIN_IN),
                          .O(CLKIN_IBUFG));
   BUFG CLK0_BUFG_INST (.I(CLK0_BUF),
                       .O(CLKFB_IN));
   DCM DCM_INST (.CLKFB(CLKFB_IN),
                 .CLKIN(CLKIN_IBUFG),
                 .DSSEN(GND1),
                 .PSCLK(GND1),
                 .PSEN(GND1),
                 .PSINCDEC(GND1),
                 .RST(RST_IN),
                 .CLKDV(),
                 .CLKFX(),
                 .CLKFX180(),
                 .CLK0(CLK0_BUF),
                 .CLK2X(),
                 .CLK2X180(),
                 .CLK90(),
                 .CLK180(),
            .CLK270(),
            .LOCKED(LOCKED_OUT),
            .PSDONE(),
            .STATUS());
   defparam DCM_INST.CLK_FEEDBACK = "1X";
   defparam DCM_INST.CLKDV_DIVIDE = 2.0;
   defparam DCM_INST.CLKFX_DIVIDE = 1;
   defparam DCM_INST.CLKFX_MULTIPLY = 4;
   defparam DCM_INST.CLKIN_DIVIDE_BY_2 = "FALSE";
   defparam DCM_INST.CLKIN_PERIOD = 40.0;
   defparam DCM_INST.CLKOUT_PHASE_SHIFT = "NONE";
   defparam DCM_INST.DESKEW_ADJUST = "SYSTEM_SYNCHRONOUS";
   defparam DCM_INST.DFS_FREQUENCY_MODE = "LOW";
   defparam DCM_INST.DLL_FREQUENCY_MODE = "LOW";
   defparam DCM_INST.DUTY_CYCLE_CORRECTION = "TRUE";
   defparam DCM_INST.FACTORY_JF = 16'hC080;
   defparam DCM_INST.PHASE_SHIFT = 0;
   defparam DCM_INST.STARTUP_WAIT = "FALSE";
endmodule
```

3. Module Connecting Datapath and Controller

```
`default_nettype none
module top(
input                   clk,
                        reset_n,
input                   [13:0]      adc_in,
input                   [2:0]       rom_addr_0,
                                    rom_addr_1,
output      [13:0]                  dac_0,
                                    dac_1
);
wire                    clr,
                        mux_1_sel,
                        we,
                        nd_fir_da,
                        nd_fir_da_inp,
                        rfd_dds,
                        rdy_dds;
wire                    [1:0]       channel;
wire                    [4:0]       dds_a;
datapath DATAPATH (
                                    .adc_in(adc_in),
                                    .clk(clk),
                                    .clr(clr),
                                    .reset_n(reset_n),
                                    .rom_addr_0(rom_addr_0),
                                    .rom_addr_1(rom_addr_1),
                                    .mux_1_sel(mux_1_sel),
                                    .we(we),
                                    .dds_a(dds_a),
                                    .nd_fir_da_inp(nd_fir_da_inp),
                                    .nd_fir_da(nd_fir_da),//
                                    //outputs
                                    .channel(channel),
                                    .rfd_dds(rfd_dds),
                                    .rdy_dds(rdy_dds),
                                    .dac_0(dac_0),
                                    .dac_1(dac_1)
                                    );
controller CONTROLLER (
                                    .clk(clk),
                                    .reset_n(reset_n),
                                    //outputs
                                    .clr(clr),
                                    .mux_1_sel(mux_1_sel),
                                    .we(we),
                                    .dds_a(dds_a),
                                    .nd_fir_da(nd_fir_da),
                                    .nd_fir_da_inp(nd_fir_da_inp),//
                                    //inputs
                                    .channel(channel),
                                    .rfd_dds(rfd_dds),
                                    .rdy_dds(rdy_dds)
                                    );

endmodule
```

4. Controller

```
module controller(
input                   clk,
                        reset_n,
input           [1:0]   channel,
input                   rfd_dds,
                        rdy_dds,
output  reg             mux_1_sel,              // mux 1 select
output  reg             we,                     //dds write enable signal
output  reg     [4:0]   dds_a,                  //dds address
output  reg             clr,
                        nd_fir_da_inp,
                        nd_fir_da
);
reg                     [7:0]   state,
                                next_state;
parameter  [7:0]  s1 = 8'd1,
                  s2 = 8'd2,
                  s3 = 8'd4,
                  s4 = 8'd8,
                  s5 = 8'd16,
                  s6 = 8'd32,
                  s7 = 8'd64,
                  s8 = 8'd128;
always @ (posedge(clk))
begin
                if (!(reset_n))
                        begin
                        state <= s1;
                        end
                else
                        begin
                        state <= next_state;
                        end
end
always @ (state,rdy_dds)
begin
                case (state)
                        s1:begin
                                        next_state<=s2;
                                end
                        s2:begin
                                        next_state<=s3;
                                end
                        s3:begin
                                        next_state<=s4;
                                end
                        s4:begin
                                        next_state<=s5;
                                end
                        s5:begin
                                        next_state<=s6;
                                end
                        s6:begin
                                        if (rdy_dds)
                                                next_state<=s7;
                                        else
                                                next_state<=s6;
                                end
                        s7:begin
```

```
                                        next_state<=s8;
                                end
                        s8:begin
                                        next_state<=s8;
                                end
                endcase
end
always @ (state)
begin
                case (state)
                        s1:begin
                                        clr<=1;
                                        mux_1_sel<=0;
                                        we<=0;
                                        dds_a<=0;
                                        nd_fir_da<=0;
                                        nd_fir_da_inp<=0;
                                end
                        s2:begin
                                        clr <= 1;
                                        mux_1_sel<=0;
                                        we<=1;
                                        dds_a<=0;
                                        nd_fir_da<=0;
                                        nd_fir_da_inp<=1;
                                end
                        s3:begin
                                        clr <= 1;
                                        mux_1_sel<=1;
                                        we<=1;
                                        dds_a<=1;
                                        nd_fir_da<=0;
                                        nd_fir_da_inp<=1;
                                end
                        s4:begin
                                        clr <= 1;
                                        mux_1_sel<=1;
                                        we<=1;
                                        dds_a<=2;
                                        nd_fir_da<=0;
                                        nd_fir_da_inp<=1;
                                end
                        s5:begin
                                        clr<=1;
                                        mux_1_sel<=0;
                                        we<=1;
                                        dds_a<=3;
                                        nd_fir_da<=0;
                                        nd_fir_da_inp<=1;
                                end
                        s6:begin
                                        clr <= 0;
                                        mux_1_sel<=0;
                                        we<=0;
                                        dds_a<=0;
                                        nd_fir_da<=0;
                                        nd_fir_da_inp<=1;
                                end
                        s7:begin
                                        clr <= 0;
                                        mux_1_sel<=0;
                                        we<=0;
                                        dds_a<=0;
                                        nd_fir_da<=0;
                                        nd_fir_da_inp<=1;
                                end
                        s8:begin
                                        clr<=0;
                                        mux_1_sel<=0;
                                        we<=0;
```

```
                                                    dds_a<=0;
                                                    nd_fir_da<=1;
                                                    nd_fir_da_inp<=1;
                                          end
                    endcase
end
endmodule
```

4. Datapath

```
// Top Level Module Connecting Different Subcomponents
`default_nettype none
module datapath(
input               [13:0]      adc_in,
input                           clk,
                                clr,
                                reset_n,
input               [2:0]       rom_addr_0,
                                rom_addr_1,
input                           mux_1_sel,
input                           we,
input               [4:0]       dds_a,
input                           nd_fir_da_inp,
                                nd_fir_da,//
output              [1:0]       channel,
output                          rfd_dds,
                                rdy_dds,
output              [13:0]      dac_0,
                                dac_1
);
wire                            rst_fir_in,
                                rdy_dds_in,
                                rdy_fir_da,
                                nd_fir_da_2,
                                rdy_fir_da_2,
                                rfd_fir_da_2,
                                rdy_sqr_ckt_0,
                                rdy_sqr_ckt_1;
wire                [1:0]       sel_o,
                                sel_o_2;
wire                [2:0]       rom_addr;
wire                [11:0]      adc_sample,
                                adc_in_tr;
wire                [31:0]      dds_data;
wire                [11:0]      cosine_out,//
                                mixer_out,//
                                fir_da_in_2;
wire                [24:0]      fir_da_inp_out,
                                fir_da_out_1,
                                fir_da_out_2;

wire                [23:0]      square_out_0,
                                square_out_1,
                                square_out;
wire                [1:0]       rdy_root;
wire                [23:0]      square_root;
input_reg #(.n1(14),.n2(12)) INPUT_REG  (
          .clk(clk),
          .reset_n(reset_n),
          .adc_in(adc_in),
          .adc_in_tr(adc_in_tr));
assign rst_fir_in = ~reset_n;
input_fir FIR_DA_INP ( // Input Anti-Aliasing Filter
          .ND(nd_fir_da_inp),
          .RDY(),
          .CLK(clk),
          .RST(rst_fir_in),
          .RFD(),
          .DIN(adc_in_tr),
          .DOUT(fir_da_inp_out));
```

```
front_decimator #(.n(12))            INPUT_DEC (// Input Decimator
                .clk(clk),
                .clr(clr),
                .rdy_dds(rdy_dds_in),
                .in(fir_da_inp_out[22:11]),
                .out(adc_sample)
                );
mux #(.n(3)) MUX_1  (
                .in0(rom_addr_0),
                .in1(rom_addr_1),
                .sel(mux_1_sel),
                .out(rom_addr)
                );
phaseinc_rom ROM ( // Phase Increment ROM
                .clk(clk),
                .clr(clr),
                .station_addr(rom_addr),
                .phase_inc(dds_data)
                );

dds  DDS (
                .DATA(dds_data),
                .WE(we),
                .A(dds_a),
                 .CLK(clk),
                .SCLR(clr),
                .CHANNEL(channel),
                 .RFD(rfd_dds),
                 .RDY(rdy_dds_in),
                 .COSINE(cosine_out)
   );
assign rdy_dds = rdy_dds_in;
mixer MIXER_1 (
                .clk(clk),
                .a(adc_sample),
                .b(cosine_out),
                .sclr(clr),
                .p(mixer_out));
fir_da FIR_DA_1 ( //Anti-Aliasing Filter
                .ND(nd_fir_da),
                .RDY(rdy_fir_da),
                .CLK(clk),
                .RST(clr),
                .RFD(),
                .DIN(mixer_out),
                .SEL_I(),
                .SEL_O(sel_o),
                .DOUT(fir_da_out_1));
sync_fir FIR1_FIR2_BRIDGE (// Downsampler
                .clk(clk),
                .clr(clr),
                .fir_da_out_1(fir_da_out_1[20:9]),//20:9
                .sel_o_1(sel_o),
                .rdy_fir_da_1(rdy_fir_da),
                .rfd_fir_da_2(rfd_fir_da_2),
                .nd_fir_da_2(nd_fir_da_2),
                .fir_da_in_2(fir_da_in_2));
fir_da_sec FIR_DA_2 ( //Baseband Filter
                .ND(nd_fir_da_2),
                .RDY(rdy_fir_da_2),
                .CLK(clk),
                .RST(clr),
                .RFD(rfd_fir_da_2),
                .DIN(fir_da_in_2),
                .SEL_I(),
                .SEL_O(sel_o_2),
                .DOUT(fir_da_out_2));
squaring_ckt SQR_CKT_0 (
                .clk(clk),
                .clr(clr),
		.fir2_out(fir_da_out_2[21:10]),
		.sel_o_2(sel_o_2),
		.rdy_fir_da_2(rdy_fir_da_2),
		.rdy_sqr_ckt(rdy_sqr_ckt_0),
		.square_out(square_out_0));
squaring_ckt_1 SQR_CKT_1 (
		.clk(clk),
		.clr(clr),
		.fir2_out(fir_da_out_2[21:10]),
		.sel_o_2(sel_o_2),
		.rdy_fir_da_2(rdy_fir_da_2),
		.rdy_sqr_ckt(rdy_sqr_ckt_1),
		.square_out(square_out_1));
clk2_clk3_bridge #(.n(24)) CK2_CK3_BRDG (
		.rdy_sqr_ckt_0(rdy_sqr_ckt_0),
		.rdy_sqr_ckt_1(rdy_sqr_ckt_1),
		.square_0(square_out_0),
		.square_1(square_out_1),
		.square_out(square_out));
square_root #(.n(24),.n1(2)) SQ_ROOT (
		.clk(clk),
		.clr(clr),
		.remainder_in(square_out),
		.sum_in({rdy_sqr_ckt_1,rdy_sqr_ckt_0}),
		.sum_out(rdy_root),
		.root_out(square_root));
mean_remover1 #(.n(12)) MEAN_REMOVER_0 (
		.clk(clk),
		.clr(clr),
		.rdy_root(rdy_root[0]),
		.in(square_root[11:0]),
		.out(dac_0));
mean_remover1 #(.n(12)) MEAN_REMOVER_1 (
		.clk(clk),
		.clr(clr),
		.rdy_root(rdy_root[1]),
		.in(square_root[11:0]),
		.out(dac_1));
endmodule


// Input Register
// This module is used to latch the digital samples from the ADC
module input_reg(clk,reset_n,adc_in,adc_in_tr);
parameter		n1 = 14,
			n2	= 12;
input			clk,
			reset_n;
input			[n1-1:0]	adc_in;			// Input from ADC (14 bit Wide)
reg			[n1-1:0]	adc_temp;
output	reg		[n2-1:0]	adc_in_tr;		// Truncated Ouput (12 bit wide)

always @ (posedge(clk))
begin
		if (!reset_n)
		begin
					adc_temp <= 0;
					adc_in_tr <= 0;
		end
		else
		begin
					adc_temp <= adc_in;
					adc_in_tr <= adc_temp[n1-1:2];
		end
end
endmodule


// Input Anti-Aliasing Filter (FIR Filter, Xilinx Core)
/*******************************************************************************
*    This file is owned and controlled by Xilinx and must be used             *
```

```
// The synopsys directives "translate_off/translate_on" specified below are
// supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity synthesis
// tools. Ensure they are correct for your synthesis tool(s).

// You must compile the wrapper file input_fir.v when simulating
// the core, input_fir. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Help".

`timescale 1ns/1ps

module input_fir(
        ND,
        RDY,
        CLK,
        RST,
        RFD,
        DIN,
        DOUT);


input ND;
output RDY;
input CLK;
input RST;
output RFD;
input [11 : 0] DIN;
output [24 : 0] DOUT;

// synopsys translate_off

   C_DA_FIR_V9_0 #(
                12,     // c_baat
                1,      // c_channels
                0,      // c_coeff_type
                12,     // c_coeff_width
                0,      // c_data_type
                12,     // c_data_width
                0,      // c_enable_rlocs
                0,      // c_filter_type
                1,      // c_has_reset
                0,      // c_has_sel_i
                0,      // c_has_sel_o
                0,      // c_has_sin_f
```

```
                    0,          // c_has_sin_r
                    0,          // c_has_sout_f
                    0,          // c_has_sout_r
                    11,         // c_latency
                    "input_fir.mif",        // c_mem_init_file
                    1,          // c_optimize
                    1,          // c_polyphase_factor
                    1,          // c_reg_output
                    0,          // c_reload
                    402,        // c_reload_delay
                    1,          // c_reload_mem_type
                    0,          // c_response
                    25,         // c_result_width
                    0,          // c_saturate
                    0,          // c_shape
                    46,         // c_taps
                    0,          // c_use_model_func
                    1)          // c_zpf
          inst (
                    .ND(ND),
                    .RDY(RDY),
                    .CLK(CLK),
                    .RST(RST),
                    .RFD(RFD),
                    .DIN(DIN),
                    .DOUT(DOUT),
                    .LD_DIN(),
                    .COEF_LD(),
                    .LD_WE(),
                    .DOUT_I(),
                    .DOUT_Q(),
                    .SEL_I(),
                    .SEL_O());


// synopsys translate_on

// FPGA Express black box declaration
// synopsys attribute fpga_dont_touch "true"
// synthesis attribute fpga_dont_touch of input_fir is "true"

// XST black box declaration
// box_type "black_box"
// synthesis attribute box_type of input_fir is "black_box"

endmodule


//Input Downsampler
module front_decimator(clk,clr,rdy_dds,in,out);
parameter  n=12;
input               clk,
                    clr;
input               rdy_dds;
input               [n-1:0]     in;
output    reg       [n-1:0]     out;
reg                 [1:0]       count;
always @ (posedge(clk))
begin
          if (clr)
                    count <= 0;
          else if (rdy_dds)
                    count <= count + 1;
end
always @ (posedge(clk))
begin
          if (clr)
                    out <= 0;
          else if ((rdy_dds==0) || ((rdy_dds==1)&&(count==2'd3)))
                    out <= in;
```

```
end
endmodule

//Mux 1
module mux(in0,in1,sel,out);
parameter n=3;
input        [n-1:0]      in0,in1;
input        sel;
output       reg          [n-1:0]      out;
always @ (*)
             case (sel)
                          1'b0:        out <= in0;
                          1'b1: out <= in1;
                          default: out <= "x";
             endcase
endmodule

//Phase Increment ROM
// This Phase Increment ROM Contains Phase Increment values corresponding to only 8 stations. More Stations can be added as
//needed.
module phaseinc_rom(
input                     clk,
                          clr,
input                     [2:0]        station_addr,
output       reg          [31:0]       phase_inc
);
// dont use double quotes in verilog
// verilog interprets integers without base specifier as signed.
always @ (posedge(clk))
begin
             case(station_addr) //Frequencies in KHZ
                          3'd0: phase_inc <= 32'd405444913;//590
                          3'd1: phase_inc <= 32'd412316860;//600
                          3'd2: phase_inc <= 32'd858993459;//1250
                          3'd3: phase_inc <= 32'd852121512;//1240
                          3'd4: phase_inc <= 32'd687194767;//1000
                          3'd5: phase_inc <= 32'd694066715;//1010
                          3'd6: phase_inc <= 32'd962072674;//1400
                          3'd7: phase_inc <= 32'd968944622;//1410
             endcase
end
endmodule

// Direct Digital Synthesizer (Xilinx Core)
///*******************************************************************************
*    This file is owned and controlled by Xilinx and must be used            *
*    solely for design, simulation, implementation and creation of           *
*    design files limited to Xilinx devices or technologies. Use             *
*    with non-Xilinx devices or technologies is expressly prohibited         *
*    and immediately terminates your license.                                *
*                                                                            *
*    XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"           *
*    SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR                 *
*    XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION         *
*    AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION             *
*    OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS               *
*    IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,                 *
*    AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE        *
*    FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY                *
*    WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE                 *
*    IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR          *
*    REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF         *
*    INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS         *
*    FOR A PARTICULAR PURPOSE.                                               *
*                                                                            *
*    Xilinx products are not intended for use in life support                *
*    appliances, devices, or systems. Use in such applications are           *
*    expressly prohibited.                                                   *
*                                                                            *
*    (c) Copyright 1995-2006 Xilinx, Inc.                                    *
*     All rights reserved.                                                    *
*******************************************************************************/

/* Behavioural components instantiated:
C_SHIFT_RAM_V7_0
C_REG_FD_V7_0
BLKMEMSP_V6_0
C_GATE_BIT_V7_0
C_SHIFT_FD_V7_0
C_ADDSUB_V7_0
C_DIST_MEM_V7_0
C_COUNTER_BINARY_V7_0
*/

`timescale 1ns/1ps

module dds(
   DATA,
   WE,
   A,
   CLK,
   SCLR,
   CHANNEL,
   RFD,
   RDY,
   COSINE
   ); // synthesis black_box

   input [31 : 0] DATA;
   input WE;
   input [4 : 0] A;
   input CLK;
   input SCLR;
   output [1 : 0] CHANNEL;
   output RFD;
   output RDY;
   output [11 : 0] COSINE;
//synopsys translate_off
   wire n0 = 1'b0;
   wire n1 = 1'b1;
   wire n2;
   wire n3;
   wire n4;
   wire n5;
   wire n6;
   wire n13;
   wire n14;
   wire n15;
   wire n16;
   wire n17;
   wire n18;
   wire n19;
   wire n20;
   wire n21;
   wire n22;
   wire n23;
   wire n24;
   wire n25;
   wire n26;
   wire n27;
   wire n28;
   wire n29;
   wire n30;
   wire n31;
   wire n32;
   wire n33;
   wire n34;
   wire n35;
   wire n36;
   wire n37;
   wire n38;
   wire n39;
   wire n40;
   wire n41;
   wire n42;
   wire n43;
   wire n44;
   wire n58;
   wire n59;
   wire n60;
   wire n61;
   wire n62;
   wire n63;
   wire n64;
   wire n65;
   wire n66;
   wire n67;
   wire n68;
   wire n69;
   wire n70;
   wire n71;
   wire n72;
   wire n73;
   wire n74;
   wire n75;
   wire n76;
   wire n79;
   wire n80;
   wire n81;
   wire n82;
   wire n83;
   wire n84;
   wire n85;
   wire n86;
   wire n87;
   wire n88;
   wire n89;
   wire n90;
   wire n91;
   wire n92;
   wire n93;
   wire n94;
   wire n95;
   wire n96;
   wire n97;
   wire n98;
   wire n99;
   wire n129;
   wire n130;
   wire n131;
   wire n132;
   wire n133;
   wire n134;
   wire n135;
   wire n136;
   wire n137;
   wire n138;
   wire n139;
   wire n140;
   wire n141;
   wire n142;
   wire n143;
   wire n144;
   wire n145;
   wire n146;
   wire n147;
   wire n148;
   wire n149;
   wire n150;
   wire n151;
   wire n152;
   wire n153;
   wire n154;
   wire n155;
   wire n156;
   wire n157;
   wire n158;
   wire n159;
   wire n160;
   wire n162;
   wire n163;
   wire n167;
   wire n168;
   wire n169;
   wire n170;
   wire n171;
   wire n172;
   wire n173;
   wire n174;
   wire n175;
   wire n176;
   wire n177;
   wire n178;
   wire n179;
   wire n180;
   wire n181;
   wire n182;
   wire n183;
   wire n184;
   wire n205;
   wire n206;
   wire n207;
   wire n208;
   wire n209;
   wire n210;
   wire n211;
   wire n212;
   wire n213;
   wire n214;
   wire n215;
   wire n216;
   wire n217;
   wire n218;
   wire n219;
   wire n220;
   wire n221;
   wire n222;
   wire n223;
   wire n224;
   wire n225;
   wire n226;
   wire n227;
   wire n228;
   wire n229;
   wire n230;
   wire n231;
   wire n232;
   wire n233;
   wire n234;
   wire n235;
   wire n236;
   wire n269;
   wire n270;
   wire n271;
   wire n272;
   wire n273;
   wire n274;
   wire n275;
   wire n276;
   wire n277;
   wire n278;
   wire n279;
   wire n280;
   wire n281;
   wire n282;
   wire n283;
   wire n284;
   wire n285;
   wire n286;
   wire n287;
   wire n288;
   wire n289;
   wire n290;
   wire n291;
   wire n292;
   wire n293;
   wire n294;
   wire n295;
   wire n296;
   wire n297;
   wire n298;
   wire n299;
   wire n300;
   wire n301;
   wire n302;
   wire n335;
   wire n336;
   wire n337;
   wire n338;
   wire n339;
   wire n340;
   wire n341;
   wire n342;
   wire n343;
   wire n344;
   wire n345;
   wire n346;
   wire n347;
   wire n348;
   wire n349;
   wire n350;
   wire n351;
   wire n352;
   wire n353;
   wire n354;
   wire n355;
   wire n356;
   wire n357;
   wire n358;
   wire n359;
   wire n360;
   wire n361;
   wire n362;
   wire n363;
   wire n364;
   wire n365;
   wire n366;
   wire n367;
   wire n1670;
   wire n1671;
   wire n1672;
   wire n1673;
   wire n1674;
   wire n1675;
   wire n1676;
   wire n1677;
   wire n1678;
   wire n1679;
   wire n1680;
   wire n1681;
   wire n1682;
   wire n1683;
   wire n1684;
   wire n1685;
   wire n1686;
   wire n1687;
   wire n1688;
   wire n1689;
   wire n1690;
   wire n1691;
   wire n1692;
   wire n1693;
   wire n1694;
   wire n1695;
   wire n1696;
   wire n1697;
   wire n1698;
   wire n1699;
   wire n1700;
   wire n1701;
   wire n1702;
   wire n1703;
   wire n2507;
   wire n2508;
   wire n2509;
   wire n2510;
   wire n2511;
   wire n2512;
   wire n2513;
   wire n2514;
   wire n2515;
   wire n2516;
   wire n2517;
   wire n2518;
   wire n2519;
   wire n2520;
   wire n2559;
   wire n2560;
   wire n2561;
   wire n2562;
   wire n2563;
   wire n2564;
   wire n2565;
   wire n2566;
   wire n2567;
   wire n2568;
   wire n2569;
   wire n2610;
   wire n2611;
   wire n2612;
   wire n2613;
   wire n2614;
   wire n2615;
   wire n2616;
   wire n2617;
   wire n2618;
   wire n2619;
   wire n2620;
   wire n2621;
   wire n2622;
   wire n2623;
   wire n2624;
   wire n2625;
   wire n2626;
   wire n2627;
   wire n2628;
   wire n2665;
   wire n2666;
   wire n2667;
   wire n2668;
   wire n2669;
   wire n2670;
   wire n2671;
   wire n2672;
   wire n2673;
   wire n2674;
   wire n2675;
   wire n2718;
   wire n2719;
   wire n2720;
   wire n2721;
   wire n2722;
   wire n2723;
   wire n2724;
   wire n2725;
   wire n2726;
   wire n3397;
   wire n3398;
   wire n3399;
   wire n3400;
   wire n3401;
   wire n3402;
   wire n3403;
   wire n3404;
   wire n3405;
   wire n3406;
   wire n3407;
   wire n3408;
   wire n3409;
   wire n3410;
   wire n3411;
   wire n3412;
   wire n3413;
   wire n3414;
   wire n3415;
   wire n3416;
   wire n3417;
   wire n3418;
   wire n3419;
   assign n129 = DATA[0];
   assign n130 = DATA[1];
   assign n131 = DATA[2];
   assign n132 = DATA[3];
   assign n133 = DATA[4];
   assign n134 = DATA[5];
   assign n135 = DATA[6];
   assign n136 = DATA[7];
   assign n137 = DATA[8];
   assign n138 = DATA[9];
   assign n139 = DATA[10];
   assign n140 = DATA[11];
   assign n141 = DATA[12];
   assign n142 = DATA[13];
   assign n143 = DATA[14];
   assign n144 = DATA[15];
   assign n145 = DATA[16];
   assign n146 = DATA[17];
   assign n147 = DATA[18];
   assign n148 = DATA[19];
   assign n149 = DATA[20];
   assign n150 = DATA[21];
   assign n151 = DATA[22];
   assign n152 = DATA[23];
   assign n153 = DATA[24];
   assign n154 = DATA[25];
   assign n155 = DATA[26];
   assign n156 = DATA[27];
   assign n157 = DATA[28];
   assign n158 = DATA[29];
   assign n159 = DATA[30];
   assign n160 = DATA[31];
   assign n6 = WE;
   assign n162 = A[0];
   assign n163 = A[1];
   assign n3 = A[2];
   assign n4 = A[3];
   assign n5 = A[4];
   assign n169 = CLK;
   assign n170 = SCLR;
   assign CHANNEL[0] = n167;
   assign CHANNEL[1] = n168;
   assign RFD = n171;
   assign RDY = n172;
   assign COSINE[0] = n173;
   assign COSINE[1] = n174;
   assign COSINE[2] = n175;
   assign COSINE[3] = n176;
   assign COSINE[4] = n177;
   assign COSINE[5] = n178;
   assign COSINE[6] = n179;
   assign COSINE[7] = n180;
   assign COSINE[8] = n181;
   assign COSINE[9] = n182;
   assign COSINE[10] = n183;
   assign COSINE[11] = n184;

     wire [3 : 0] BU2_I;
       assign BU2_I[0] = n3;
       assign BU2_I[1] = n4;
       assign BU2_I[2] = n5;
       assign BU2_I[3] = n6;
     wire BU2_T;
       assign BU2_T = 1'b0;
     wire BU2_EN;
       assign BU2_EN = 1'b0;
     wire BU2_Q;
     wire BU2_CLK;
       assign BU2_CLK = 1'b0;
     wire BU2_CE;
       assign BU2_CE = 1'b0;
     wire BU2_ACLR;
       assign BU2_ACLR = 1'b0;
     wire BU2_ASET;
       assign BU2_ASET = 1'b0;
     wire BU2_AINIT;
       assign BU2_AINIT = 1'b0;
     wire BU2_SCLR;
       assign BU2_SCLR = 1'b0;
     wire BU2_SSET;
       assign BU2_SSET = 1'b0;
     wire BU2_SINIT;
       assign BU2_SINIT = 1'b0;
     wire BU2_O;
       assign n2 = BU2_O;
     C_GATE_BIT_V7_0 #(
       "0"    /* c_ainit_val*/,
       0    /* c_enable_rlocs*/,
       0    /* c_gate_type*/,
       0    /* c_has_aclr*/,
       0    /* c_has_ainit*/,
       0    /* c_has_aset*/,
       0    /* c_has_ce*/,
       1    /* c_has_o*/,
       1    /* c_has_q*/,
       0    /* c_has_sclr*/,
       0    /* c_has_sinit*/,
       0    /* c_has_sset*/,
       4    /* c_inputs*/,
       "0111"    /* c_input_inv_mask*/,
       0    /* c_pipe_stages*/,
      "0"    /* c_sinit_val*/,
      0    /* c_sync_enable*/,
      1    /* c_sync_priority*/
   )
   BU2(
      .I(BU2_I),
      .T(BU2_T),
      .EN(BU2_EN),
      .Q(BU2_Q),
      .CLK(BU2_CLK),
      .CE(BU2_CE),
      .ACLR(BU2_ACLR),
      .ASET(BU2_ASET),
      .AINIT(BU2_AINIT),
      .SCLR(BU2_SCLR),
      .SSET(BU2_SSET),
      .SINIT(BU2_SINIT),
      .O(BU2_O)
   );

   wire [1 : 0] BU9_Q;
      assign n301 = BU9_Q[0];
      assign n302 = BU9_Q[1];
   wire BU9_CLK;
      assign BU9_CLK = n169;
   wire BU9_SCLR;
      assign BU9_SCLR = n170;
   C_COUNTER_BINARY_V7_0 #(
      "00"    /* c_ainit_val*/,
      "01"    /* c_count_by*/,
      0    /* c_count_mode*/,
      "11"    /* c_count_to*/,
      0    /* c_enable_rlocs*/,
      0    /* c_has_aclr*/,
      0    /* c_has_ainit*/,
      0    /* c_has_aset*/,
      0    /* c_has_ce*/,
      0    /* c_has_iv*/,
      0    /* c_has_l*/,
      0    /* c_has_load*/,
      0    /* c_has_q_thresh0*/,
      0    /* c_has_q_thresh1*/,
      1    /* c_has_sclr*/,
      0    /* c_has_sinit*/,
      0    /* c_has_sset*/,
      0    /* c_has_thresh0*/,
      0    /* c_has_thresh1*/,
      0    /* c_has_up*/,
      0    /* c_load_enable*/,
      0    /* c_load_low*/,
      0    /* c_pipe_stages*/,
      1    /* c_restrict_count*/,
      "00"    /* c_sinit_val*/,
      0    /* c_sync_enable*/,
      0    /* c_sync_priority*/,
      "00"    /* c_thresh0_value*/,
      "00"    /* c_thresh1_value*/,
      0    /* c_thresh_early*/,
      2    /* c_width*/
   )
   BU9(
      .Q(BU9_Q),
      .CLK(BU9_CLK),
      .SCLR(BU9_SCLR)
   );

   wire [3 : 0] BU33_A;
      assign BU33_A[0] = n162;
      assign BU33_A[1] = n163;
      assign BU33_A[2] = 1'b0;
      assign BU33_A[3] = 1'b0;
   wire BU33_CLK;
      assign BU33_CLK = n169;
   wire [31 : 0] BU33_D;
      assign BU33_D[0] = n129;
      assign BU33_D[1] = n130;
      assign BU33_D[2] = n131;
      assign BU33_D[3] = n132;
      assign BU33_D[4] = n133;
      assign BU33_D[5] = n134;
      assign BU33_D[6] = n135;
      assign BU33_D[7] = n136;
      assign BU33_D[8] = n137;
      assign BU33_D[9] = n138;
      assign BU33_D[10] = n139;
      assign BU33_D[11] = n140;
      assign BU33_D[12] = n141;
      assign BU33_D[13] = n142;
      assign BU33_D[14] = n143;
      assign BU33_D[15] = n144;
      assign BU33_D[16] = n145;
      assign BU33_D[17] = n146;
      assign BU33_D[18] = n147;
      assign BU33_D[19] = n148;
      assign BU33_D[20] = n149;
      assign BU33_D[21] = n150;
      assign BU33_D[22] = n151;
      assign BU33_D[23] = n152;
      assign BU33_D[24] = n153;
      assign BU33_D[25] = n154;
      assign BU33_D[26] = n155;
      assign BU33_D[27] = n156;
      assign BU33_D[28] = n157;
      assign BU33_D[29] = n158;
      assign BU33_D[30] = n159;
      assign BU33_D[31] = n160;
   wire BU33_WE;
      assign BU33_WE = n2;
   wire [3 : 0] BU33_DPRA;
      assign BU33_DPRA[0] = n301;
      assign BU33_DPRA[1] = n302;
      assign BU33_DPRA[2] = 1'b0;
      assign BU33_DPRA[3] = 1'b0;
   wire BU33_QDPO_CLK;
      assign BU33_QDPO_CLK = n169;
   wire [31 : 0] BU33_QDPO;
      assign n335 = BU33_QDPO[0];
      assign n336 = BU33_QDPO[1];
      assign n337 = BU33_QDPO[2];
      assign n338 = BU33_QDPO[3];
      assign n339 = BU33_QDPO[4];
      assign n340 = BU33_QDPO[5];
      assign n341 = BU33_QDPO[6];
      assign n342 = BU33_QDPO[7];
      assign n343 = BU33_QDPO[8];
      assign n344 = BU33_QDPO[9];
      assign n345 = BU33_QDPO[10];
      assign n346 = BU33_QDPO[11];
      assign n347 = BU33_QDPO[12];
      assign n348 = BU33_QDPO[13];
      assign n349 = BU33_QDPO[14];
      assign n350 = BU33_QDPO[15];
      assign n351 = BU33_QDPO[16];
      assign n352 = BU33_QDPO[17];
      assign n353 = BU33_QDPO[18];
      assign n354 = BU33_QDPO[19];
      assign n355 = BU33_QDPO[20];
      assign n356 = BU33_QDPO[21];
      assign n357 = BU33_QDPO[22];
      assign n358 = BU33_QDPO[23];
      assign n359 = BU33_QDPO[24];
      assign n360 = BU33_QDPO[25];
      assign n361 = BU33_QDPO[26];
      assign n362 = BU33_QDPO[27];
      assign n363 = BU33_QDPO[28];
      assign n364 = BU33_QDPO[29];
      assign n365 = BU33_QDPO[30];
      assign n366 = BU33_QDPO[31];
   wire [31 : 0] BU33_QSPO;
   wire BU33_QSPO_SRST;
      assign BU33_QSPO_SRST = n170;
   wire BU33_QDPO_SRST;
      assign BU33_QDPO_SRST = n170;
   C_DIST_MEM_V7_0 #(
      4    /* c_addr_width*/,
      "0"    /* c_default_data*/,
      2    /* c_default_data_radix*/,
      16    /* c_depth*/,
      0    /* c_enable_rlocs*/,
      0    /* c_generate_mif*/,
      1    /* c_has_clk*/,
      1    /* c_has_d*/,
      0    /* c_has_dpo*/,
      1    /* c_has_dpra*/,
      0    /* c_has_i_ce*/,
      1    /* c_has_qdpo*/,
      0    /* c_has_qdpo_ce*/,
      1    /* c_has_qdpo_clk*/,
      0    /* c_has_qdpo_rst*/,
      1    /* c_has_qdpo_srst*/,
      1    /* c_has_qspo*/,
      0    /* c_has_qspo_ce*/,
      0    /* c_has_qspo_rst*/,
      1    /* c_has_qspo_srst*/,
      0    /* c_has_rd_en*/,
      0    /* c_has_spo*/,
      0    /* c_has_spra*/,
      1    /* c_has_we*/,
      1    /* c_latency*/,
      "dds_PHASE_ACCUM_MEM.mif"    /* c_mem_init_file*/,
      2    /* c_mem_type*/,
      0    /* c_mux_type*/,
      0    /* c_qce_joined*/,
      0    /* c_qualify_we*/,
      1    /* c_read_mif*/,
      0    /* c_reg_a_d_inputs*/,
      0    /* c_reg_dpra_input*/,
      0    /* c_sync_enable*/,
      32    /* c_width*/
   )
   BU33(
      .A(BU33_A),
      .CLK(BU33_CLK),
      .D(BU33_D),
      .WE(BU33_WE),
      .DPRA(BU33_DPRA),
      .QDPO_CLK(BU33_QDPO_CLK),
      .QDPO(BU33_QDPO),
      .QSPO(BU33_QSPO),
      .QSPO_SRST(BU33_QSPO_SRST),
      .QDPO_SRST(BU33_QDPO_SRST)
   );

   wire [31 : 0] BU199_A;
      assign BU199_A[0] = n13;
      assign BU199_A[1] = n14;
      assign BU199_A[2] = n15;
      assign BU199_A[3] = n16;
      assign BU199_A[4] = n17;
      assign BU199_A[5] = n18;
      assign BU199_A[6] = n19;
      assign BU199_A[7] = n20;
      assign BU199_A[8] = n21;
      assign BU199_A[9] = n22;
      assign BU199_A[10] = n23;
      assign BU199_A[11] = n24;
      assign BU199_A[12] = n25;
      assign BU199_A[13] = n26;
      assign BU199_A[14] = n27;
      assign BU199_A[15] = n28;
      assign BU199_A[16] = n29;
      assign BU199_A[17] = n30;
      assign BU199_A[18] = n31;
      assign BU199_A[19] = n32;
      assign BU199_A[20] = n33;
      assign BU199_A[21] = n34;
      assign BU199_A[22] = n35;
      assign BU199_A[23] = n36;
      assign BU199_A[24] = n37;
      assign BU199_A[25] = n38;
      assign BU199_A[26] = n39;
      assign BU199_A[27] = n40;
      assign BU199_A[28] = n41;
      assign BU199_A[29] = n42;
      assign BU199_A[30] = n43;
      assign BU199_A[31] = n44;
   wire [31 : 0] BU199_B;
      assign BU199_B[0] = n335;
      assign BU199_B[1] = n336;
      assign BU199_B[2] = n337;
      assign BU199_B[3] = n338;
      assign BU199_B[4] = n339;
      assign BU199_B[5] = n340;
      assign BU199_B[6] = n341;
      assign BU199_B[7] = n342;
      assign BU199_B[8] = n343;
      assign BU199_B[9] = n344;
      assign BU199_B[10] = n345;
      assign BU199_B[11] = n346;
      assign BU199_B[12] = n347;
      assign BU199_B[13] = n348;
      assign BU199_B[14] = n349;
      assign BU199_B[15] = n350;
      assign BU199_B[16] = n351;
      assign BU199_B[17] = n352;
      assign BU199_B[18] = n353;
      assign BU199_B[19] = n354;
      assign BU199_B[20] = n355;
      assign BU199_B[21] = n356;
      assign BU199_B[22] = n357;
      assign BU199_B[23] = n358;
      assign BU199_B[24] = n359;
      assign BU199_B[25] = n360;
      assign BU199_B[26] = n361;
      assign BU199_B[27] = n362;
      assign BU199_B[28] = n363;
      assign BU199_B[29] = n364;
      assign BU199_B[30] = n365;
      assign BU199_B[31] = n366;
   wire [31 : 0] BU199_Q;
      assign n205 = BU199_Q[0];
      assign n206 = BU199_Q[1];
      assign n207 = BU199_Q[2];
      assign n208 = BU199_Q[3];
      assign n209 = BU199_Q[4];
      assign n210 = BU199_Q[5];
      assign n211 = BU199_Q[6];
      assign n212 = BU199_Q[7];
      assign n213 = BU199_Q[8];
      assign n214 = BU199_Q[9];
      assign n215 = BU199_Q[10];
      assign n216 = BU199_Q[11];
      assign n217 = BU199_Q[12];
      assign n218 = BU199_Q[13];
      assign n219 = BU199_Q[14];
      assign n220 = BU199_Q[15];
      assign n221 = BU199_Q[16];
      assign n222 = BU199_Q[17];
      assign n223 = BU199_Q[18];
      assign n224 = BU199_Q[19];
      assign n225 = BU199_Q[20];
      assign n226 = BU199_Q[21];
      assign n227 = BU199_Q[22];
      assign n228 = BU199_Q[23];
      assign n229 = BU199_Q[24];
      assign n230 = BU199_Q[25];
      assign n231 = BU199_Q[26];
      assign n232 = BU199_Q[27];
      assign n233 = BU199_Q[28];
      assign n234 = BU199_Q[29];
      assign n235 = BU199_Q[30];
      assign n236 = BU199_Q[31];
   wire BU199_CLK;
      assign BU199_CLK = n169;
   C_ADDSUB_V7_0 #(
      0    /* c_add_mode*/,
      "00000000000000000000000000000000"    /* c_ainit_val*/,
      1    /* c_a_type*/,
      32    /* c_a_width*/,
      1    /* c_bypass_enable*/,
      0    /* c_bypass_low*/,
      0    /* c_b_constant*/,
      1    /* c_b_type*/,
      "00000000000000000000000000000000"    /* c_b_value*/,
      32    /* c_b_width*/,
      0    /* c_enable_rlocs*/,
      0    /* c_has_aclr*/,
      0    /* c_has_add*/,
      0    /* c_has_ainit*/,
      0    /* c_has_aset*/,
      0    /* c_has_a_signed*/,
      0    /* c_has_bypass*/,
      0    /* c_has_bypass_with_cin*/,
      0    /* c_has_b_in*/,
      0    /* c_has_b_out*/,
      0    /* c_has_b_signed*/,
      0    /* c_has_ce*/,
      0    /* c_has_c_in*/,
      0    /* c_has_c_out*/,
      0    /* c_has_ovfl*/,
      1    /* c_has_q*/,
      0    /* c_has_q_b_out*/,
      0    /* c_has_q_c_out*/,
      0    /* c_has_q_ovfl*/,
      1    /* c_has_s*/,
      0    /* c_has_sclr*/,
      0    /* c_has_sinit*/,
      0    /* c_has_sset*/,
      31    /* c_high_bit*/,
      1    /* c_latency*/,
      0    /* c_low_bit*/,
      32    /* c_out_width*/,
      0    /* c_pipe_stages*/,
      "00000000000000000000000000000000"    /* c_sinit_val*/,
      0    /* c_sync_enable*/,
      0    /* c_sync_priority*/
   )
   BU199(
      .A(BU199_A),
      .B(BU199_B),
      .Q(BU199_Q),
      .CLK(BU199_CLK)
   );

   wire BU392_CLK;
      assign BU392_CLK = n169;
   wire [31 : 0] BU392_D;
      assign BU392_D[0] = n205;
      assign BU392_D[1] = n206;
      assign BU392_D[2] = n207;
      assign BU392_D[3] = n208;
      assign BU392_D[4] = n209;
      assign BU392_D[5] = n210;
      assign BU392_D[6] = n211;
      assign BU392_D[7] = n212;
      assign BU392_D[8] = n213;
      assign BU392_D[9] = n214;
      assign BU392_D[10] = n215;
      assign BU392_D[11] = n216;
      assign BU392_D[12] = n217;
      assign BU392_D[13] = n218;
      assign BU392_D[14] = n219;
      assign BU392_D[15] = n220;
      assign BU392_D[16] = n221;
      assign BU392_D[17] = n222;
      assign BU392_D[18] = n223;
      assign BU392_D[19] = n224;
      assign BU392_D[20] = n225;
      assign BU392_D[21] = n226;
      assign BU392_D[22] = n227;
      assign BU392_D[23] = n228;
      assign BU392_D[24] = n229;
      assign BU392_D[25] = n230;
      assign BU392_D[26] = n231;
      assign BU392_D[27] = n232;
      assign BU392_D[28] = n233;
      assign BU392_D[29] = n234;
      assign BU392_D[30] = n235;
      assign BU392_D[31] = n236;
   wire [31 : 0] BU392_Q;
      assign n269 = BU392_Q[0];
      assign n270 = BU392_Q[1];
      assign n271 = BU392_Q[2];
      assign n272 = BU392_Q[3];
      assign n273 = BU392_Q[4];
      assign n274 = BU392_Q[5];
      assign n275 = BU392_Q[6];
      assign n276 = BU392_Q[7];
      assign n277 = BU392_Q[8];
      assign n278 = BU392_Q[9];
      assign n279 = BU392_Q[10];
      assign n280 = BU392_Q[11];
      assign n281 = BU392_Q[12];
      assign n282 = BU392_Q[13];
      assign n283 = BU392_Q[14];
      assign n284 = BU392_Q[15];
      assign n285 = BU392_Q[16];
      assign n286 = BU392_Q[17];
      assign n287 = BU392_Q[18];
      assign n288 = BU392_Q[19];
      assign n289 = BU392_Q[20];
      assign n290 = BU392_Q[21];
      assign n291 = BU392_Q[22];
      assign n292 = BU392_Q[23];
      assign n293 = BU392_Q[24];
      assign n294 = BU392_Q[25];
      assign n295 = BU392_Q[26];
      assign n296 = BU392_Q[27];
      assign n297 = BU392_Q[28];
      assign n298 = BU392_Q[29];
      assign n299 = BU392_Q[30];
      assign n300 = BU392_Q[31];
   C_SHIFT_RAM_V7_0 #(
      1    /* c_addr_width*/,
      "00000000000000000000000000000000"    /* c_ainit_val*/,
      "00000000000000000000000000000000"    /* c_default_data*/,
      2    /* c_default_data_radix*/,
      2    /* c_depth*/,
      0    /* c_enable_rlocs*/,
      0    /* c_generate_mif*/,
      0    /* c_has_a*/,
      0    /* c_has_aclr*/,
      0    /* c_has_ainit*/,
      0    /* c_has_aset*/,
      0    /* c_has_ce*/,
      0    /* c_has_sclr*/,
      0    /* c_has_sinit*/,
      0    /* c_has_sset*/,
      "null"    /* c_mem_init_file*/,
      2    /* c_mem_init_radix*/,
      0    /* c_read_mif*/,
      1    /* c_reg_last_bit*/,
      0    /* c_shift_type*/,
      "00000000000000000000000000000000"    /* c_sinit_val*/,
      0    /* c_sync_enable*/,
      0    /* c_sync_priority*/,
      32    /* c_width*/
   )
   BU392(
      .CLK(BU392_CLK),
      .D(BU392_D),
      .Q(BU392_Q)
   );

   wire BU492_CLK;
      assign BU492_CLK = n169;
   wire BU492_SDOUT;
      assign n367 = BU492_SDOUT;
   wire BU492_SCLR;
      assign BU492_SCLR = n170;
   C_SHIFT_FD_V7_0 #(
      "0000"    /* c_ainit_val*/,
      0    /* c_enable_rlocs*/,
      1    /* c_fill_data*/,
      0    /* c_has_aclr*/,
      0    /* c_has_ainit*/,
      0    /* c_has_aset*/,
      0    /* c_has_ce*/,
      0    /* c_has_d*/,
      0    /* c_has_lsb_2_msb*/,
      0    /* c_has_q*/,
      1    /* c_has_sclr*/,
      0    /* c_has_sdin*/,
      1    /* c_has_sdout*/,
      0    /* c_has_sinit*/,
      0    /* c_has_sset*/,
      0    /* c_shift_type*/,
      "0000"    /* c_sinit_val*/,
      0    /* c_sync_enable*/,
      0    /* c_sync_priority*/,
      4    /* c_width*/
   )
   BU492(
      .CLK(BU492_CLK),
      .SDOUT(BU492_SDOUT),
      .SCLR(BU492_SCLR)
   );

   wire [31 : 0] BU504_D;
      assign BU504_D[0] = n269;
```

```
  assign BU504_D[1] = n270;
  assign BU504_D[2] = n271;
  assign BU504_D[3] = n272;
  assign BU504_D[4] = n273;
  assign BU504_D[5] = n274;
  assign BU504_D[6] = n275;
  assign BU504_D[7] = n276;
  assign BU504_D[8] = n277;
  assign BU504_D[9] = n278;
  assign BU504_D[10] = n279;
  assign BU504_D[11] = n280;
  assign BU504_D[12] = n281;
  assign BU504_D[13] = n282;
  assign BU504_D[14] = n283;
  assign BU504_D[15] = n284;
  assign BU504_D[16] = n285;
  assign BU504_D[17] = n286;
  assign BU504_D[18] = n287;
  assign BU504_D[19] = n288;
  assign BU504_D[20] = n289;
  assign BU504_D[21] = n290;
  assign BU504_D[22] = n291;
  assign BU504_D[23] = n292;
  assign BU504_D[24] = n293;
  assign BU504_D[25] = n294;
  assign BU504_D[26] = n295;
  assign BU504_D[27] = n296;
  assign BU504_D[28] = n297;
  assign BU504_D[29] = n298;
  assign BU504_D[30] = n299;
  assign BU504_D[31] = n300;
wire [31 : 0] BU504_Q;
  assign n13 = BU504_Q[0];
  assign n14 = BU504_Q[1];
  assign n15 = BU504_Q[2];
  assign n16 = BU504_Q[3];
  assign n17 = BU504_Q[4];
  assign n18 = BU504_Q[5];
  assign n19 = BU504_Q[6];
  assign n20 = BU504_Q[7];
  assign n21 = BU504_Q[8];
  assign n22 = BU504_Q[9];
  assign n23 = BU504_Q[10];
  assign n24 = BU504_Q[11];
  assign n25 = BU504_Q[12];
  assign n26 = BU504_Q[13];
  assign n27 = BU504_Q[14];
  assign n28 = BU504_Q[15];
  assign n29 = BU504_Q[16];
  assign n30 = BU504_Q[17];
  assign n31 = BU504_Q[18];
  assign n32 = BU504_Q[19];
  assign n33 = BU504_Q[20];
  assign n34 = BU504_Q[21];
  assign n35 = BU504_Q[22];
  assign n36 = BU504_Q[23];
  assign n37 = BU504_Q[24];
  assign n38 = BU504_Q[25];
  assign n39 = BU504_Q[26];
  assign n40 = BU504_Q[27];
  assign n41 = BU504_Q[28];
  assign n42 = BU504_Q[29];
  assign n43 = BU504_Q[30];
  assign n44 = BU504_Q[31];
wire BU504_CLK;
  assign BU504_CLK = n169;
wire BU504_CE;
  assign BU504_CE = n367;
wire BU504_SCLR;
  assign BU504_SCLR = n170;
```

```
C_REG_FD_V7_0 #(
  "00000000000000000000000000000000"    /* c_ainit_val*/,
  0    /* c_enable_rlocs*/,
  0    /* c_has_aclr*/,
  0    /* c_has_ainit*/,
  0    /* c_has_aset*/,
  1    /* c_has_ce*/,
  1    /* c_has_sclr*/,
  0    /* c_has_sinit*/,
  0    /* c_has_sset*/,
  "00000000000000000000000000000000"    /* c_sinit_val*/,
  0    /* c_sync_enable*/,
  0    /* c_sync_priority*/,
  32    /* c_width*/
)
BU504(
  .D(BU504_D),
  .Q(BU504_Q),
  .CLK(BU504_CLK),
  .CE(BU504_CE),
  .SCLR(BU504_SCLR)
);

wire BU569_CLK;
  assign BU569_CLK = n169;
wire BU569_SDOUT;
  assign n79 = BU569_SDOUT;
wire BU569_SCLR;
  assign BU569_SCLR = n170;
C_SHIFT_FD_V7_0 #(
  "000"    /* c_ainit_val*/,
  0    /* c_enable_rlocs*/,
  1    /* c_fill_data*/,
  0    /* c_has_aclr*/,
  0    /* c_has_ainit*/,
  0    /* c_has_aset*/,
  0    /* c_has_ce*/,
  0    /* c_has_d*/,
  0    /* c_has_lsb_2_msb*/,
  0    /* c_has_q*/,
  1    /* c_has_sclr*/,
  0    /* c_has_sdin*/,
  1    /* c_has_sdout*/,
  0    /* c_has_sinit*/,
  0    /* c_has_sset*/,
  1    /* c_shift_type*/,
  "000"    /* c_sinit_val*/,
  0    /* c_sync_enable*/,
  1    /* c_sync_priority*/,
  3    /* c_width*/
)
BU569(
  .CLK(BU569_CLK),
  .SDOUT(BU569_SDOUT),
  .SCLR(BU569_SCLR)
);

wire [1 : 0] BU580_Q;
  assign n1670 = BU580_Q[0];
  assign n1671 = BU580_Q[1];
wire BU580_CLK;
  assign BU580_CLK = n169;
wire BU580_SCLR;
  assign BU580_SCLR = n170;
C_COUNTER_BINARY_V7_0 #(
  "00"    /* c_ainit_val*/,
  "01"    /* c_count_by*/,
  0    /* c_count_mode*/,
  "11"    /* c_count_to*/,
  0    /* c_enable_rlocs*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_ce*/,
   0    /* c_has_iv*/,
   0    /* c_has_l*/,
   0    /* c_has_load*/,
   0    /* c_has_q_thresh0*/,
   0    /* c_has_q_thresh1*/,
   1    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   0    /* c_has_thresh0*/,
   0    /* c_has_thresh1*/,
   0    /* c_has_up*/,
   0    /* c_load_enable*/,
   0    /* c_load_low*/,
   0    /* c_pipe_stages*/,
   1    /* c_restrict_count*/,
   "00"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/,
   "00"    /* c_thresh0_value*/,
   "00"    /* c_thresh1_value*/,
   0    /* c_thresh_early*/,
   2    /* c_width*/
)
BU580(
   .Q(BU580_Q),
   .CLK(BU580_CLK),
   .SCLR(BU580_SCLR)
);

wire [3 : 0] BU604_A;
   assign BU604_A[0] = n1670;
   assign BU604_A[1] = n1671;
   assign BU604_A[2] = 1'b0;
   assign BU604_A[3] = 1'b0;
wire BU604_CLK;
   assign BU604_CLK = n169;
wire [31 : 0] BU604_QSPO;
   assign n1672 = BU604_QSPO[0];
   assign n1673 = BU604_QSPO[1];
   assign n1674 = BU604_QSPO[2];
   assign n1675 = BU604_QSPO[3];
   assign n1676 = BU604_QSPO[4];
   assign n1677 = BU604_QSPO[5];
   assign n1678 = BU604_QSPO[6];
   assign n1679 = BU604_QSPO[7];
   assign n1680 = BU604_QSPO[8];
   assign n1681 = BU604_QSPO[9];
   assign n1682 = BU604_QSPO[10];
   assign n1683 = BU604_QSPO[11];
   assign n1684 = BU604_QSPO[12];
   assign n1685 = BU604_QSPO[13];
   assign n1686 = BU604_QSPO[14];
   assign n1687 = BU604_QSPO[15];
   assign n1688 = BU604_QSPO[16];
   assign n1689 = BU604_QSPO[17];
   assign n1690 = BU604_QSPO[18];
   assign n1691 = BU604_QSPO[19];
   assign n1692 = BU604_QSPO[20];
   assign n1693 = BU604_QSPO[21];
   assign n1694 = BU604_QSPO[22];
   assign n1695 = BU604_QSPO[23];
   assign n1696 = BU604_QSPO[24];
   assign n1697 = BU604_QSPO[25];
   assign n1698 = BU604_QSPO[26];
   assign n1699 = BU604_QSPO[27];
   assign n1700 = BU604_QSPO[28];
   assign n1701 = BU604_QSPO[29];
   assign n1702 = BU604_QSPO[30];
   assign n1703 = BU604_QSPO[31];
wire BU604_QSPO_SRST;
   assign BU604_QSPO_SRST = n170;
C_DIST_MEM_V7_0 #(
   4    /* c_addr_width*/,
   "0"    /* c_default_data*/,
   2    /* c_default_data_radix*/,
   16    /* c_depth*/,
   0    /* c_enable_rlocs*/,
   0    /* c_generate_mif*/,
   1    /* c_has_clk*/,
   0    /* c_has_d*/,
   0    /* c_has_dpo*/,
   0    /* c_has_dpra*/,
   0    /* c_has_i_ce*/,
   0    /* c_has_qdpo*/,
   0    /* c_has_qdpo_ce*/,
   0    /* c_has_qdpo_clk*/,
   0    /* c_has_qdpo_rst*/,
   1    /* c_has_qdpo_srst*/,
   1    /* c_has_qspo*/,
   0    /* c_has_qspo_ce*/,
   0    /* c_has_qspo_rst*/,
   1    /* c_has_qspo_srst*/,
   0    /* c_has_rd_en*/,
   0    /* c_has_spo*/,
   0    /* c_has_spra*/,
   0    /* c_has_we*/,
   1    /* c_latency*/,
   "dds_OFFSET_ADDER_MEM.mif"    /* c_mem_init_file*/,
   0    /* c_mem_type*/,
   0    /* c_mux_type*/,
   0    /* c_qce_joined*/,
   0    /* c_qualify_we*/,
   1    /* c_read_mif*/,
   0    /* c_reg_a_d_inputs*/,
   0    /* c_reg_dpra_input*/,
   0    /* c_sync_enable*/,
   32    /* c_width*/
)
BU604(
   .A(BU604_A),
   .CLK(BU604_CLK),
   .QSPO(BU604_QSPO),
   .QSPO_SRST(BU604_QSPO_SRST)
);

wire [31 : 0] BU706_A;
   assign BU706_A[0] = n13;
   assign BU706_A[1] = n14;
   assign BU706_A[2] = n15;
   assign BU706_A[3] = n16;
   assign BU706_A[4] = n17;
   assign BU706_A[5] = n18;
   assign BU706_A[6] = n19;
   assign BU706_A[7] = n20;
   assign BU706_A[8] = n21;
   assign BU706_A[9] = n22;
   assign BU706_A[10] = n23;
   assign BU706_A[11] = n24;
   assign BU706_A[12] = n25;
   assign BU706_A[13] = n26;
   assign BU706_A[14] = n27;
   assign BU706_A[15] = n28;
   assign BU706_A[16] = n29;
   assign BU706_A[17] = n30;
   assign BU706_A[18] = n31;
   assign BU706_A[19] = n32;
   assign BU706_A[20] = n33;
   assign BU706_A[21] = n34;
   assign BU706_A[22] = n35;
   assign BU706_A[23] = n36;
   assign BU706_A[24] = n37;
   assign BU706_A[25] = n38;
   assign BU706_A[26] = n39;
   assign BU706_A[27] = n40;
   assign BU706_A[28] = n41;
   assign BU706_A[29] = n42;
   assign BU706_A[30] = n43;
   assign BU706_A[31] = n44;
wire [31 : 0] BU706_B;
   assign BU706_B[0] = n1672;
   assign BU706_B[1] = n1673;
   assign BU706_B[2] = n1674;
   assign BU706_B[3] = n1675;
   assign BU706_B[4] = n1676;
   assign BU706_B[5] = n1677;
   assign BU706_B[6] = n1678;
   assign BU706_B[7] = n1679;
   assign BU706_B[8] = n1680;
   assign BU706_B[9] = n1681;
   assign BU706_B[10] = n1682;
   assign BU706_B[11] = n1683;
   assign BU706_B[12] = n1684;
   assign BU706_B[13] = n1685;
   assign BU706_B[14] = n1686;
   assign BU706_B[15] = n1687;
   assign BU706_B[16] = n1688;
   assign BU706_B[17] = n1689;
   assign BU706_B[18] = n1690;
   assign BU706_B[19] = n1691;
   assign BU706_B[20] = n1692;
   assign BU706_B[21] = n1693;
   assign BU706_B[22] = n1694;
   assign BU706_B[23] = n1695;
   assign BU706_B[24] = n1696;
   assign BU706_B[25] = n1697;
   assign BU706_B[26] = n1698;
   assign BU706_B[27] = n1699;
   assign BU706_B[28] = n1700;
   assign BU706_B[29] = n1701;
   assign BU706_B[30] = n1702;
   assign BU706_B[31] = n1703;
wire [31 : 0] BU706_Q;
   assign n58 = BU706_Q[13];
   assign n59 = BU706_Q[14];
   assign n60 = BU706_Q[15];
   assign n61 = BU706_Q[16];
   assign n62 = BU706_Q[17];
   assign n63 = BU706_Q[18];
   assign n64 = BU706_Q[19];
   assign n65 = BU706_Q[20];
   assign n66 = BU706_Q[21];
   assign n67 = BU706_Q[22];
   assign n68 = BU706_Q[23];
   assign n69 = BU706_Q[24];
   assign n70 = BU706_Q[25];
   assign n71 = BU706_Q[26];
   assign n72 = BU706_Q[27];
   assign n73 = BU706_Q[28];
   assign n74 = BU706_Q[29];
   assign n75 = BU706_Q[30];
   assign n76 = BU706_Q[31];
wire BU706_CLK;
   assign BU706_CLK = n169;
C_ADDSUB_V7_0 #(
   0    /* c_add_mode*/,
   "00000000000000000000000000000000"    /* c_ainit_val*/,
   1    /* c_a_type*/,
   32    /* c_a_width*/,
   1    /* c_bypass_enable*/,
   0    /* c_bypass_low*/,
   0    /* c_b_constant*/,
   1    /* c_b_type*/,
   "00000000000000000000000000000000"    /* c_b_value*/,
   32    /* c_b_width*/,
   0    /* c_enable_rlocs*/,
   0    /* c_has_aclr*/,
   0    /* c_has_add*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_a_signed*/,
   0    /* c_has_bypass*/,
   0    /* c_has_bypass_with_cin*/,
   0    /* c_has_b_in*/,
   0    /* c_has_b_out*/,
   0    /* c_has_b_signed*/,
   0    /* c_has_ce*/,
   0    /* c_has_c_in*/,
   0    /* c_has_c_out*/,
   0    /* c_has_ovfl*/,
   1    /* c_has_q*/,
   0    /* c_has_q_b_out*/,
   0    /* c_has_q_c_out*/,
   0    /* c_has_q_ovfl*/,
   1    /* c_has_s*/,
   0    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   31    /* c_high_bit*/,
   1    /* c_latency*/,
   0    /* c_low_bit*/,
   32    /* c_out_width*/,
   0    /* c_pipe_stages*/,
   "00000000000000000000000000000000"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/
)
BU706(
   .A(BU706_A),
   .B(BU706_B),
   .Q(BU706_Q),
   .CLK(BU706_CLK)
);

wire [18 : 0] BU1255_A;
   assign BU1255_A[0] = n58;
   assign BU1255_A[1] = n59;
   assign BU1255_A[2] = n60;
   assign BU1255_A[3] = n61;
   assign BU1255_A[4] = n62;
   assign BU1255_A[5] = n63;
   assign BU1255_A[6] = n64;
   assign BU1255_A[7] = n65;
   assign BU1255_A[8] = n66;
   assign BU1255_A[9] = n67;
   assign BU1255_A[10] = n68;
   assign BU1255_A[11] = n69;
   assign BU1255_A[12] = n70;
   assign BU1255_A[13] = n71;
   assign BU1255_A[14] = n72;
   assign BU1255_A[15] = n73;
   assign BU1255_A[16] = n74;
   assign BU1255_A[17] = n75;
   assign BU1255_A[18] = n76;
wire [9 : 0] BU1255_B;
   assign BU1255_B[0] = n90;
   assign BU1255_B[1] = n91;
   assign BU1255_B[2] = n92;
   assign BU1255_B[3] = n93;
   assign BU1255_B[4] = n94;
   assign BU1255_B[5] = n95;
   assign BU1255_B[6] = n96;
   assign BU1255_B[7] = n97;
   assign BU1255_B[8] = n98;
   assign BU1255_B[9] = n99;
wire [9 : 0] BU1255_Q;
   assign n80 = BU1255_Q[0];
   assign n81 = BU1255_Q[1];
   assign n82 = BU1255_Q[2];
   assign n83 = BU1255_Q[3];
   assign n84 = BU1255_Q[4];
   assign n85 = BU1255_Q[5];
   assign n86 = BU1255_Q[6];
   assign n87 = BU1255_Q[7];
   assign n88 = BU1255_Q[8];
   assign n89 = BU1255_Q[9];
wire BU1255_CLK;
   assign BU1255_CLK = n169;
wire BU1255_SCLR;
   assign BU1255_SCLR = n170;
C_ADDSUB_V7_0 #(
   0    /* c_add_mode*/,
   "0000000000"    /* c_ainit_val*/,
   1    /* c_a_type*/,
   19    /* c_a_width*/,
   1    /* c_bypass_enable*/,
   0    /* c_bypass_low*/,
   0    /* c_b_constant*/,
   0    /* c_b_type*/,
   "0000000000000000000"    /* c_b_value*/,
   10    /* c_b_width*/,
   0    /* c_enable_rlocs*/,
   0    /* c_has_aclr*/,
   0    /* c_has_add*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_a_signed*/,
   0    /* c_has_bypass*/,
   0    /* c_has_bypass_with_cin*/,
   0    /* c_has_b_in*/,
   0    /* c_has_b_out*/,
   0    /* c_has_b_signed*/,
   0    /* c_has_ce*/,
   0    /* c_has_c_in*/,
   0    /* c_has_c_out*/,
   0    /* c_has_ovfl*/,
   1    /* c_has_q*/,
   0    /* c_has_q_b_out*/,
   0    /* c_has_q_c_out*/,
   0    /* c_has_q_ovfl*/,
   1    /* c_has_s*/,
   1    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   18    /* c_high_bit*/,
   1    /* c_latency*/,
   9    /* c_low_bit*/,
   10    /* c_out_width*/,
   0    /* c_pipe_stages*/,
   "0000000000"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/
)
BU1255(
   .A(BU1255_A),
   .B(BU1255_B),
   .Q(BU1255_Q),
   .CLK(BU1255_CLK),
   .SCLR(BU1255_SCLR)
);

wire BU901_CLK;
   assign BU901_CLK = n169;
wire BU901_SDOUT;
   assign n2508 = BU901_SDOUT;
wire BU901_SCLR;
   assign BU901_SCLR = n170;
C_SHIFT_FD_V7_0 #(
   "00"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   1    /* c_fill_data*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_ce*/,
   0    /* c_has_d*/,
   0    /* c_has_lsb_2_msb*/,
   0    /* c_has_q*/,
   1    /* c_has_sclr*/,
   0    /* c_has_sdin*/,
   1    /* c_has_sdout*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   0    /* c_shift_type*/,
   "00"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/,
   2    /* c_width*/
)
BU901(
   .CLK(BU901_CLK),
   .SDOUT(BU901_SDOUT),
   .SCLR(BU901_SCLR)
);

wire [1 : 0] BU909_Q;
wire BU909_CLK;
   assign BU909_CLK = n169;
wire BU909_Q_THRESH0;
   assign n2507 = BU909_Q_THRESH0;
wire BU909_CE;
   assign BU909_CE = n2508;
wire BU909_SCLR;
   assign BU909_SCLR = n170;
C_COUNTER_BINARY_V7_0 #(
   "00"    /* c_ainit_val*/,
   "01"    /* c_count_by*/,
   0    /* c_count_mode*/,
   "11"    /* c_count_to*/,
   0    /* c_enable_rlocs*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   1    /* c_has_ce*/,
   0    /* c_has_iv*/,
   0    /* c_has_l*/,
   0    /* c_has_load*/,
   1    /* c_has_q_thresh0*/,
   0    /* c_has_q_thresh1*/,
   1    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   0    /* c_has_thresh0*/,
   0    /* c_has_thresh1*/,
   0    /* c_has_up*/,
   0    /* c_load_enable*/,
   0    /* c_load_low*/,
   0    /* c_pipe_stages*/,
   1    /* c_restrict_count*/,
   "00"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/,
   "11"    /* c_thresh0_value*/,
   "00"    /* c_thresh1_value*/,
   1    /* c_thresh_early*/,
   2    /* c_width*/
)
BU909(
   .Q(BU909_Q),
   .CLK(BU909_CLK),
   .Q_THRESH0(BU909_Q_THRESH0),
   .CE(BU909_CE),
   .SCLR(BU909_SCLR)
);

wire [8 : 0] BU1194_A;
   assign BU1194_A[0] = n2610;
   assign BU1194_A[1] = n2611;
   assign BU1194_A[2] = n2612;
   assign BU1194_A[3] = n2613;
   assign BU1194_A[4] = n2614;
   assign BU1194_A[5] = n2615;
   assign BU1194_A[6] = n2616;
   assign BU1194_A[7] = n2617;
   assign BU1194_A[8] = n2618;
wire [8 : 0] BU1194_B;
   assign BU1194_B[0] = n2718;
   assign BU1194_B[1] = n2719;
   assign BU1194_B[2] = n2720;
   assign BU1194_B[3] = n2721;
   assign BU1194_B[4] = n2722;
   assign BU1194_B[5] = n2723;
   assign BU1194_B[6] = n2724;
   assign BU1194_B[7] = n2725;
   assign BU1194_B[8] = n2726;
wire [9 : 0] BU1194_Q;
   assign n90 = BU1194_Q[0];
   assign n91 = BU1194_Q[1];
   assign n92 = BU1194_Q[2];
   assign n93 = BU1194_Q[3];
   assign n94 = BU1194_Q[4];
   assign n95 = BU1194_Q[5];
   assign n96 = BU1194_Q[6];
   assign n97 = BU1194_Q[7];
   assign n98 = BU1194_Q[8];
   assign n99 = BU1194_Q[9];
wire BU1194_CLK;
   assign BU1194_CLK = n169;
wire BU1194_CE;
   assign BU1194_CE = n2507;
wire BU1194_SCLR;
   assign BU1194_SCLR = n170;
C_ADDSUB_V7_0 #(
   0    /* c_add_mode*/,
   "0000000000"    /* c_ainit_val*/,
   0    /* c_a_type*/,
   9    /* c_a_width*/,
   0    /* c_bypass_enable*/,
   0    /* c_bypass_low*/,
   0    /* c_b_constant*/,
   0    /* c_b_type*/,
   "0000000000"    /* c_b_value*/,
   9    /* c_b_width*/,
   0    /* c_enable_rlocs*/,
   0    /* c_has_aclr*/,
   0    /* c_has_add*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_a_signed*/,
   0    /* c_has_bypass*/,
   0    /* c_has_bypass_with_cin*/,
   0    /* c_has_b_in*/,
   0    /* c_has_b_out*/,
   0    /* c_has_b_signed*/,
   1    /* c_has_ce*/,
   0    /* c_has_c_in*/,
   0    /* c_has_c_out*/,
   0    /* c_has_ovfl*/,
   1    /* c_has_q*/,
   0    /* c_has_q_b_out*/,
   0    /* c_has_q_c_out*/,
   0    /* c_has_q_ovfl*/,
   1    /* c_has_s*/,
   1    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   9    /* c_high_bit*/,
   1    /* c_latency*/,
   0    /* c_low_bit*/,
   10    /* c_out_width*/,
   0    /* c_pipe_stages*/,
   "0000000000"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/
)
BU1194(
   .A(BU1194_A),
   .B(BU1194_B),
   .Q(BU1194_Q),
   .CLK(BU1194_CLK),
   .CE(BU1194_CE),
   .SCLR(BU1194_SCLR)
);

wire [7 : 0] BU1008_A;
   assign BU1008_A[0] = n2509;
   assign BU1008_A[1] = n2510;
   assign BU1008_A[2] = n2511;
   assign BU1008_A[3] = n2512;
   assign BU1008_A[4] = n2513;
   assign BU1008_A[5] = n2514;
   assign BU1008_A[6] = n2515;
   assign BU1008_A[7] = n2516;
wire [7 : 0] BU1008_B;
   assign BU1008_B[0] = n2559;
   assign BU1008_B[1] = n2560;
   assign BU1008_B[2] = n2561;
   assign BU1008_B[3] = n2562;
   assign BU1008_B[4] = n2563;
   assign BU1008_B[5] = n2564;
   assign BU1008_B[6] = n2565;
   assign BU1008_B[7] = n2566;
wire [8 : 0] BU1008_Q;
   assign n2610 = BU1008_Q[0];
   assign n2611 = BU1008_Q[1];
   assign n2612 = BU1008_Q[2];
   assign n2613 = BU1008_Q[3];
   assign n2614 = BU1008_Q[4];
   assign n2615 = BU1008_Q[5];
   assign n2616 = BU1008_Q[6];
   assign n2617 = BU1008_Q[7];
   assign n2618 = BU1008_Q[8];
wire BU1008_CLK;
   assign BU1008_CLK = n169;
wire BU1008_CE;
   assign BU1008_CE = n2507;
wire BU1008_SCLR;
```

```
   assign BU1008_SCLR = n170;
C_ADDSUB_V7_0 #(
   0    /* c_add_mode*/,
   "000000000"    /* c_ainit_val*/,
   0    /* c_a_type*/,
   8    /* c_a_width*/,
   0    /* c_bypass_enable*/,
   0    /* c_bypass_low*/,
   0    /* c_b_constant*/,
   0    /* c_b_type*/,
   "000000000"    /* c_b_value*/,
   8    /* c_b_width*/,
   0    /* c_enable_rlocs*/,
   0    /* c_has_aclr*/,
   0    /* c_has_add*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_a_signed*/,
   0    /* c_has_bypass*/,
   0    /* c_has_bypass_with_cin*/,
   0    /* c_has_b_in*/,
   0    /* c_has_b_out*/,
   0    /* c_has_b_signed*/,
   1    /* c_has_ce*/,
   0    /* c_has_c_in*/,
   0    /* c_has_c_out*/,
   0    /* c_has_ovfl*/,
   1    /* c_has_q*/,
   0    /* c_has_q_b_out*/,
   0    /* c_has_q_c_out*/,
   0    /* c_has_q_ovfl*/,
   1    /* c_has_s*/,
   1    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   8    /* c_high_bit*/,
   1    /* c_latency*/,
   0    /* c_low_bit*/,
   9    /* c_out_width*/,
   0    /* c_pipe_stages*/,
   "000000000"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/
)
BU1008(
   .A(BU1008_A),
   .B(BU1008_B),
   .Q(BU1008_Q),
   .CLK(BU1008_CLK),
   .CE(BU1008_CE),
   .SCLR(BU1008_SCLR)
);

wire [3 : 0] BU939_I;
   assign BU939_I[0] = n2518;
   assign BU939_I[1] = n2519;
   assign BU939_I[2] = n2520;
   assign BU939_I[3] = n2516;
wire BU939_T;
   assign BU939_T = 1'b0;
wire BU939_EN;
   assign BU939_EN = 1'b0;
wire BU939_Q;
wire BU939_CLK;
   assign BU939_CLK = 1'b0;
wire BU939_CE;
   assign BU939_CE = 1'b0;
wire BU939_ACLR;
   assign BU939_ACLR = 1'b0;
wire BU939_ASET;
```

```
   assign BU939_ASET = 1'b0;
wire BU939_AINIT;
   assign BU939_AINIT = 1'b0;
wire BU939_SCLR;
   assign BU939_SCLR = 1'b0;
wire BU939_SSET;
   assign BU939_SSET = 1'b0;
wire BU939_SINIT;
   assign BU939_SINIT = 1'b0;
wire BU939_O;
   assign n2517 = BU939_O;
C_GATE_BIT_V7_0 #(
   "0"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   4    /* c_gate_type*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_ce*/,
   1    /* c_has_o*/,
   1    /* c_has_q*/,
   0    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   4    /* c_inputs*/,
   "0000"    /* c_input_inv_mask*/,
   0    /* c_pipe_stages*/,
   "0"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   1    /* c_sync_priority*/
)
BU939(
   .I(BU939_I),
   .T(BU939_T),
   .EN(BU939_EN),
   .Q(BU939_Q),
   .CLK(BU939_CLK),
   .CE(BU939_CE),
   .ACLR(BU939_ACLR),
   .ASET(BU939_ASET),
   .AINIT(BU939_AINIT),
   .SCLR(BU939_SCLR),
   .SSET(BU939_SSET),
   .SINIT(BU939_SINIT),
   .O(BU939_O)
);

wire BU944_CLK;
   assign BU944_CLK = n169;
wire BU944_SDIN;
   assign BU944_SDIN = n2517;
wire [12 : 0] BU944_Q;
   assign n2518 = BU944_Q[0];
   assign n2519 = BU944_Q[2];
   assign n2520 = BU944_Q[3];
   assign n2509 = BU944_Q[5];
   assign n2510 = BU944_Q[6];
   assign n2511 = BU944_Q[7];
   assign n2512 = BU944_Q[8];
   assign n2513 = BU944_Q[9];
   assign n2514 = BU944_Q[10];
   assign n2515 = BU944_Q[11];
   assign n2516 = BU944_Q[12];
wire BU944_CE;
   assign BU944_CE = n2507;
wire BU944_SINIT;
   assign BU944_SINIT = n170;
C_SHIFT_FD_V7_0 #(
   "1000000000000"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   5    /* c_fill_data*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   1    /* c_has_ce*/,
   0    /* c_has_d*/,
   0    /* c_has_lsb_2_msb*/,
   1    /* c_has_q*/,
   0    /* c_has_sclr*/,
   1    /* c_has_sdin*/,
   0    /* c_has_sdout*/,
   1    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   0    /* c_shift_type*/,
   "1000000000000"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/,
   13    /* c_width*/
)
BU944(
   .CLK(BU944_CLK),
   .SDIN(BU944_SDIN),
   .Q(BU944_Q),
   .CE(BU944_CE),
   .SINIT(BU944_SINIT)
);

wire [3 : 0] BU973_I;
   assign BU973_I[0] = n2568;
   assign BU973_I[1] = n2569;
   assign BU973_I[2] = n2562;
   assign BU973_I[3] = n2566;
wire BU973_T;
   assign BU973_T = 1'b0;
wire BU973_EN;
   assign BU973_EN = 1'b0;
wire BU973_Q;
wire BU973_CLK;
   assign BU973_CLK = 1'b0;
wire BU973_CE;
   assign BU973_CE = 1'b0;
wire BU973_ACLR;
   assign BU973_ACLR = 1'b0;
wire BU973_ASET;
   assign BU973_ASET = 1'b0;
wire BU973_AINIT;
   assign BU973_AINIT = 1'b0;
wire BU973_SCLR;
   assign BU973_SCLR = 1'b0;
wire BU973_SSET;
   assign BU973_SSET = 1'b0;
wire BU973_SINIT;
   assign BU973_SINIT = 1'b0;
wire BU973_O;
   assign n2567 = BU973_O;
C_GATE_BIT_V7_0 #(
   "0"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   4    /* c_gate_type*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_ce*/,
   1    /* c_has_o*/,
   1    /* c_has_q*/,
   0    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   4    /* c_inputs*/,
   "0000"    /* c_input_inv_mask*/,
   0   /* c_pipe_stages*/,
   "0"   /* c_sinit_val*/,
   0   /* c_sync_enable*/,
   1   /* c_sync_priority*/
)
BU973(
   .I(BU973_I),
   .T(BU973_T),
   .EN(BU973_EN),
   .Q(BU973_Q),
   .CLK(BU973_CLK),
   .CE(BU973_CE),
   .ACLR(BU973_ACLR),
   .ASET(BU973_ASET),
   .AINIT(BU973_AINIT),
   .SCLR(BU973_SCLR),
   .SSET(BU973_SSET),
   .SINIT(BU973_SINIT),
   .O(BU973_O)
);

wire BU978_CLK;
   assign BU978_CLK = n169;
wire BU978_SDIN;
   assign BU978_SDIN = n2567;
wire [13 : 0] BU978_Q;
   assign n2568 = BU978_Q[0];
   assign n2569 = BU978_Q[5];
   assign n2559 = BU978_Q[6];
   assign n2560 = BU978_Q[7];
   assign n2561 = BU978_Q[8];
   assign n2562 = BU978_Q[9];
   assign n2563 = BU978_Q[10];
   assign n2564 = BU978_Q[11];
   assign n2565 = BU978_Q[12];
   assign n2566 = BU978_Q[13];
wire BU978_CE;
   assign BU978_CE = n2507;
wire BU978_SINIT;
   assign BU978_SINIT = n170;
C_SHIFT_FD_V7_0 #(
   "10000000000000"   /* c_ainit_val*/,
   0   /* c_enable_rlocs*/,
   5   /* c_fill_data*/,
   0   /* c_has_aclr*/,
   0   /* c_has_ainit*/,
   0   /* c_has_aset*/,
   1   /* c_has_ce*/,
   0   /* c_has_d*/,
   0   /* c_has_lsb_2_msb*/,
   1   /* c_has_q*/,
   0   /* c_has_sclr*/,
   1   /* c_has_sdin*/,
   0   /* c_has_sdout*/,
   1   /* c_has_sinit*/,
   0   /* c_has_sset*/,
   0   /* c_shift_type*/,
   "10000000000000"   /* c_sinit_val*/,
   0   /* c_sync_enable*/,
   0   /* c_sync_priority*/,
   14   /* c_width*/
)
BU978(
   .CLK(BU978_CLK),
   .SDIN(BU978_SDIN),
   .Q(BU978_Q),
   .CE(BU978_CE),
   .SINIT(BU978_SINIT)
);
```

```
wire [7 : 0] BU1140_A;
   assign BU1140_A[0] = n2619;
   assign BU1140_A[1] = n2620;
   assign BU1140_A[2] = n2621;
   assign BU1140_A[3] = n2622;
   assign BU1140_A[4] = n2623;
   assign BU1140_A[5] = n2624;
   assign BU1140_A[6] = n2625;
   assign BU1140_A[7] = n2626;
wire [7 : 0] BU1140_B;
   assign BU1140_B[0] = n2665;
   assign BU1140_B[1] = n2666;
   assign BU1140_B[2] = n2667;
   assign BU1140_B[3] = n2668;
   assign BU1140_B[4] = n2669;
   assign BU1140_B[5] = n2670;
   assign BU1140_B[6] = n2671;
   assign BU1140_B[7] = n2672;
wire [8 : 0] BU1140_Q;
   assign n2718 = BU1140_Q[0];
   assign n2719 = BU1140_Q[1];
   assign n2720 = BU1140_Q[2];
   assign n2721 = BU1140_Q[3];
   assign n2722 = BU1140_Q[4];
   assign n2723 = BU1140_Q[5];
   assign n2724 = BU1140_Q[6];
   assign n2725 = BU1140_Q[7];
   assign n2726 = BU1140_Q[8];
wire BU1140_CLK;
   assign BU1140_CLK = n169;
wire BU1140_CE;
   assign BU1140_CE = n2507;
wire BU1140_SCLR;
   assign BU1140_SCLR = n170;
C_ADDSUB_V7_0 #(
   0    /* c_add_mode*/,
   "000000000"    /* c_ainit_val*/,
   0    /* c_a_type*/,
   8    /* c_a_width*/,
   0    /* c_bypass_enable*/,
   0    /* c_bypass_low*/,
   0    /* c_b_constant*/,
   0    /* c_b_type*/,
   "000000000"    /* c_b_value*/,
   8    /* c_b_width*/,
   0    /* c_enable_rlocs*/,
   0    /* c_has_aclr*/,
   0    /* c_has_add*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_a_signed*/,
   0    /* c_has_bypass*/,
   0    /* c_has_bypass_with_cin*/,
   0    /* c_has_b_in*/,
   0    /* c_has_b_out*/,
   0    /* c_has_b_signed*/,
   1    /* c_has_ce*/,
   0    /* c_has_c_in*/,
   0    /* c_has_c_out*/,
   0    /* c_has_ovfl*/,
   1    /* c_has_q*/,
   0    /* c_has_q_b_out*/,
   0    /* c_has_q_c_out*/,
   0    /* c_has_q_ovfl*/,
   1    /* c_has_s*/,
   1    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   8    /* c_high_bit*/,
   1    /* c_latency*/,
   0    /* c_low_bit*/,
   9    /* c_out_width*/,
   0    /* c_pipe_stages*/,
   "000000000"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/
)
BU1140(
   .A(BU1140_A),
   .B(BU1140_B),
   .Q(BU1140_Q),
   .CLK(BU1140_CLK),
   .CE(BU1140_CE),
   .SCLR(BU1140_SCLR)
);

wire [1 : 0] BU1063_I;
   assign BU1063_I[0] = n2628;
   assign BU1063_I[1] = n2626;
wire BU1063_T;
   assign BU1063_T = 1'b0;
wire BU1063_EN;
   assign BU1063_EN = 1'b0;
wire BU1063_Q;
wire BU1063_CLK;
   assign BU1063_CLK = 1'b0;
wire BU1063_CE;
   assign BU1063_CE = 1'b0;
wire BU1063_ACLR;
   assign BU1063_ACLR = 1'b0;
wire BU1063_ASET;
   assign BU1063_ASET = 1'b0;
wire BU1063_AINIT;
   assign BU1063_AINIT = 1'b0;
wire BU1063_SCLR;
   assign BU1063_SCLR = 1'b0;
wire BU1063_SSET;
   assign BU1063_SSET = 1'b0;
wire BU1063_SINIT;
   assign BU1063_SINIT = 1'b0;
wire BU1063_O;
   assign n2627 = BU1063_O;
C_GATE_BIT_V7_0 #(
   "0"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   4    /* c_gate_type*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_ce*/,
   1    /* c_has_o*/,
   1    /* c_has_q*/,
   0    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   2    /* c_inputs*/,
   "00"    /* c_input_inv_mask*/,
   0    /* c_pipe_stages*/,
   "0"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   1    /* c_sync_priority*/
)
BU1063(
   .I(BU1063_I),
   .T(BU1063_T),
   .EN(BU1063_EN),
   .Q(BU1063_Q),
   .CLK(BU1063_CLK),
   .CE(BU1063_CE),
   .ACLR(BU1063_ACLR),
   .ASET(BU1063_ASET),
   .AINIT(BU1063_AINIT),
   .SCLR(BU1063_SCLR),
   .SSET(BU1063_SSET),
   .SINIT(BU1063_SINIT),
   .O(BU1063_O)
);

wire BU1068_CLK;
   assign BU1068_CLK = n169;
wire BU1068_SDIN;
   assign BU1068_SDIN = n2627;
wire [14 : 0] BU1068_Q;
   assign n2628 = BU1068_Q[0];
   assign n2619 = BU1068_Q[7];
   assign n2620 = BU1068_Q[8];
   assign n2621 = BU1068_Q[9];
   assign n2622 = BU1068_Q[10];
   assign n2623 = BU1068_Q[11];
   assign n2624 = BU1068_Q[12];
   assign n2625 = BU1068_Q[13];
   assign n2626 = BU1068_Q[14];
wire BU1068_CE;
   assign BU1068_CE = n2507;
wire BU1068_SINIT;
   assign BU1068_SINIT = n170;
C_SHIFT_FD_V7_0 #(
   "100000000000000"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   5    /* c_fill_data*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   1    /* c_has_ce*/,
   0    /* c_has_d*/,
   0    /* c_has_lsb_2_msb*/,
   1    /* c_has_q*/,
   0    /* c_has_sclr*/,
   1    /* c_has_sdin*/,
   0    /* c_has_sdout*/,
   1    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   0    /* c_shift_type*/,
   "100000000000000"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/,
   15    /* c_width*/
)
BU1068(
   .CLK(BU1068_CLK),
   .SDIN(BU1068_SDIN),
   .Q(BU1068_Q),
   .CE(BU1068_CE),
   .SINIT(BU1068_SINIT)
);

wire [3 : 0] BU1101_I;
   assign BU1101_I[0] = n2674;
   assign BU1101_I[1] = n2675;
   assign BU1101_I[2] = n2668;
   assign BU1101_I[3] = n2672;
wire BU1101_T;
   assign BU1101_T = 1'b0;
wire BU1101_EN;
   assign BU1101_EN = 1'b0;
wire BU1101_Q;
wire BU1101_CLK;
   assign BU1101_CLK = 1'b0;
wire BU1101_CE;
   assign BU1101_CE = 1'b0;
```

```
wire BU1101_ACLR;
   assign BU1101_ACLR = 1'b0;
wire BU1101_ASET;
   assign BU1101_ASET = 1'b0;
wire BU1101_AINIT;
   assign BU1101_AINIT = 1'b0;
wire BU1101_SCLR;
   assign BU1101_SCLR = 1'b0;
wire BU1101_SSET;
   assign BU1101_SSET = 1'b0;
wire BU1101_SINIT;
   assign BU1101_SINIT = 1'b0;
wire BU1101_O;
   assign n2673 = BU1101_O;
C_GATE_BIT_V7_0 #(
   "0"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   4    /* c_gate_type*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_ce*/,
   1    /* c_has_o*/,
   1    /* c_has_q*/,
   0    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   4    /* c_inputs*/,
   "0000"    /* c_input_inv_mask*/,
   0    /* c_pipe_stages*/,
   "0"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   1    /* c_sync_priority*/
)
BU1101(
   .I(BU1101_I),
   .T(BU1101_T),
   .EN(BU1101_EN),
   .Q(BU1101_Q),
   .CLK(BU1101_CLK),
   .CE(BU1101_CE),
   .ACLR(BU1101_ACLR),
   .ASET(BU1101_ASET),
   .AINIT(BU1101_AINIT),
   .SCLR(BU1101_SCLR),
   .SSET(BU1101_SSET),
   .SINIT(BU1101_SINIT),
   .O(BU1101_O)
);

wire BU1106_CLK;
   assign BU1106_CLK = n169;
wire BU1106_SDIN;
   assign BU1106_SDIN = n2673;
wire [15 : 0] BU1106_Q;
   assign n2674 = BU1106_Q[0];
   assign n2675 = BU1106_Q[2];
   assign n2665 = BU1106_Q[8];
   assign n2666 = BU1106_Q[9];
   assign n2667 = BU1106_Q[10];
   assign n2668 = BU1106_Q[11];
   assign n2669 = BU1106_Q[12];
   assign n2670 = BU1106_Q[13];
   assign n2671 = BU1106_Q[14];
   assign n2672 = BU1106_Q[15];
wire BU1106_CE;
   assign BU1106_CE = n2507;
wire BU1106_SINIT;
   assign BU1106_SINIT = n170;
C_SHIFT_FD_V7_0 #(
   "1000000000000000"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   5    /* c_fill_data*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   1    /* c_has_ce*/,
   0    /* c_has_d*/,
   0    /* c_has_lsb_2_msb*/,
   1    /* c_has_q*/,
   0    /* c_has_sclr*/,
   1    /* c_has_sdin*/,
   0    /* c_has_sdout*/,
   1    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   0    /* c_shift_type*/,
   "1000000000000000"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/,
   16    /* c_width*/
)
BU1106(
   .CLK(BU1106_CLK),
   .SDIN(BU1106_SDIN),
   .Q(BU1106_Q),
   .CE(BU1106_CE),
   .SINIT(BU1106_SINIT)
);

wire [9 : 0] BU1353_D;
   assign BU1353_D[0] = n80;
   assign BU1353_D[1] = n81;
   assign BU1353_D[2] = n82;
   assign BU1353_D[3] = n83;
   assign BU1353_D[4] = n84;
   assign BU1353_D[5] = n85;
   assign BU1353_D[6] = n86;
   assign BU1353_D[7] = n87;
   assign BU1353_D[8] = n88;
   assign BU1353_D[9] = n89;
wire [9 : 0] BU1353_Q;
   assign n3397 = BU1353_Q[0];
   assign n3398 = BU1353_Q[1];
   assign n3399 = BU1353_Q[2];
   assign n3400 = BU1353_Q[3];
   assign n3401 = BU1353_Q[4];
   assign n3402 = BU1353_Q[5];
   assign n3403 = BU1353_Q[6];
   assign n3404 = BU1353_Q[7];
   assign n3405 = BU1353_Q[8];
   assign n3406 = BU1353_Q[9];
wire BU1353_CLK;
   assign BU1353_CLK = n169;
C_REG_FD_V7_0 #(
   "0000000000"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_ce*/,
   0    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   "0000000000"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/,
   10    /* c_width*/
)
BU1353(
   .D(BU1353_D),
   .Q(BU1353_Q),
   .CLK(BU1353_CLK)
);

wire [9 : 0] BU1345_addr;
   assign BU1345_addr[9] = n3406;
   assign BU1345_addr[8] = n3405;
   assign BU1345_addr[7] = n3404;
   assign BU1345_addr[6] = n3403;
   assign BU1345_addr[5] = n3402;
   assign BU1345_addr[4] = n3401;
   assign BU1345_addr[3] = n3400;
   assign BU1345_addr[2] = n3399;
   assign BU1345_addr[1] = n3398;
   assign BU1345_addr[0] = n3397;
wire BU1345_clk;
   assign BU1345_clk = n169;
wire [11 : 0] BU1345_din;
   assign BU1345_din[11] = 1'b0;
   assign BU1345_din[10] = 1'b0;
   assign BU1345_din[9] = 1'b0;
   assign BU1345_din[8] = 1'b0;
   assign BU1345_din[7] = 1'b0;
   assign BU1345_din[6] = 1'b0;
   assign BU1345_din[5] = 1'b0;
   assign BU1345_din[4] = 1'b0;
   assign BU1345_din[3] = 1'b0;
   assign BU1345_din[2] = 1'b0;
   assign BU1345_din[1] = 1'b0;
   assign BU1345_din[0] = 1'b0;
wire [11 : 0] BU1345_dout;
   assign n3418 = BU1345_dout[11];
   assign n3417 = BU1345_dout[10];
   assign n3416 = BU1345_dout[9];
   assign n3415 = BU1345_dout[8];
   assign n3414 = BU1345_dout[7];
   assign n3413 = BU1345_dout[6];
   assign n3412 = BU1345_dout[5];
   assign n3411 = BU1345_dout[4];
   assign n3410 = BU1345_dout[3];
   assign n3409 = BU1345_dout[2];
   assign n3408 = BU1345_dout[1];
   assign n3407 = BU1345_dout[0];
wire BU1345_en;
   assign BU1345_en = 1'b0;
wire BU1345_nd;
   assign BU1345_nd = 1'b0;
wire BU1345_rfd;
wire BU1345_rdy;
wire BU1345_sinit;
   assign BU1345_sinit = 1'b0;
wire BU1345_we;
   assign BU1345_we = 1'b0;
BLKMEMSP_V6_0 #(
   10    /* c_addr_width*/,
   "000000000000000000"    /* c_default_data*/,
   1024    /* c_depth*/,
   0    /* c_enable_rlocs*/,
   0    /* c_has_default_data*/,
   0    /* c_has_din*/,
   0    /* c_has_en*/,
   0    /* c_has_limit_data_pitch*/,
   0    /* c_has_nd*/,
   0    /* c_has_rdy*/,
   0    /* c_has_rfd*/,
   0    /* c_has_sinit*/,
   0    /* c_has_we*/,
   18    /* c_limit_data_pitch*/,
   "dds_SINCOS_TABLE_TRIG_ROM.mif"    /* c_mem_init_file*/,
   0    /* c_pipe_stages*/,
   0    /* c_reg_inputs*/,
   "0000000000000000000"    /* c_sinit_value*/,
   12    /* c_width*/,
   0    /* c_write_mode*/,
   "0"    /* c_ybottom_addr*/,
   1    /* c_yclk_is_rising*/,
   1    /* c_yen_is_high*/,
   "hierarchy1"    /* c_yhierarchy*/,
   0    /* c_ymake_bmm*/,
   "4kx4"    /* c_yprimitive_type*/,
   1    /* c_ysinit_is_high*/,
   "1024"    /* c_ytop_addr*/,
   0    /* c_yuse_single_primitive*/,
   1    /* c_ywe_is_high*/,
   1    /* c_yydisable_warnings*/
)
BU1345(
   .ADDR(BU1345_addr),
   .CLK(BU1345_clk),
   .DIN(BU1345_din),
   .DOUT(BU1345_dout),
   .EN(BU1345_en),
   .ND(BU1345_nd),
   .RFD(BU1345_rfd),
   .RDY(BU1345_rdy),
   .SINIT(BU1345_sinit),
   .WE(BU1345_we)
);

wire [11 : 0] BU1376_D;
   assign BU1376_D[0] = n3407;
   assign BU1376_D[1] = n3408;
   assign BU1376_D[2] = n3409;
   assign BU1376_D[3] = n3410;
   assign BU1376_D[4] = n3411;
   assign BU1376_D[5] = n3412;
   assign BU1376_D[6] = n3413;
   assign BU1376_D[7] = n3414;
   assign BU1376_D[8] = n3415;
   assign BU1376_D[9] = n3416;
   assign BU1376_D[10] = n3417;
   assign BU1376_D[11] = n3418;
wire [11 : 0] BU1376_Q;
   assign n173 = BU1376_Q[0];
   assign n174 = BU1376_Q[1];
   assign n175 = BU1376_Q[2];
   assign n176 = BU1376_Q[3];
   assign n177 = BU1376_Q[4];
   assign n178 = BU1376_Q[5];
   assign n179 = BU1376_Q[6];
   assign n180 = BU1376_Q[7];
   assign n181 = BU1376_Q[8];
   assign n182 = BU1376_Q[9];
   assign n183 = BU1376_Q[10];
   assign n184 = BU1376_Q[11];
wire BU1376_CLK;
   assign BU1376_CLK = n169;
wire BU1376_CE;
   assign BU1376_CE = n3419;
wire BU1376_SCLR;
   assign BU1376_SCLR = n170;
C_REG_FD_V7_0 #(
   "000000000000"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   1    /* c_has_ce*/,
   1    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   "000000000000"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/,
   12    /* c_width*/
)
BU1376(
   .D(BU1376_D),
   .Q(BU1376_Q),
   .CLK(BU1376_CLK),
   .CE(BU1376_CE),
   .SCLR(BU1376_SCLR)
);

wire [0 : 0] BU1402_I;
   assign BU1402_I[0] = 1'b1;
wire BU1402_T;
   assign BU1402_T = 1'b0;
wire BU1402_EN;
   assign BU1402_EN = 1'b0;
wire BU1402_Q;
wire BU1402_CLK;
   assign BU1402_CLK = 1'b0;
wire BU1402_CE;
   assign BU1402_CE = 1'b0;
wire BU1402_ACLR;
   assign BU1402_ACLR = 1'b0;
wire BU1402_ASET;
   assign BU1402_ASET = 1'b0;
wire BU1402_AINIT;
   assign BU1402_AINIT = 1'b0;
wire BU1402_SCLR;
   assign BU1402_SCLR = 1'b0;
wire BU1402_SSET;
   assign BU1402_SSET = 1'b0;
wire BU1402_SINIT;
   assign BU1402_SINIT = 1'b0;
wire BU1402_O;
   assign n171 = BU1402_O;
C_GATE_BIT_V7_0 #(
   "0"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   7    /* c_gate_type*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_ce*/,
   1    /* c_has_o*/,
   1    /* c_has_q*/,
   0    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   1    /* c_inputs*/,
   "0"    /* c_input_inv_mask*/,
   0    /* c_pipe_stages*/,
   "0"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   1    /* c_sync_priority*/
)
BU1402(
   .I(BU1402_I),
   .T(BU1402_T),
   .EN(BU1402_EN),
   .Q(BU1402_Q),
   .CLK(BU1402_CLK),
   .CE(BU1402_CE),
   .ACLR(BU1402_ACLR),
   .ASET(BU1402_ASET),
   .AINIT(BU1402_AINIT),
   .SCLR(BU1402_SCLR),
   .SSET(BU1402_SSET),
   .SINIT(BU1402_SINIT),
   .O(BU1402_O)
);

wire BU1406_CLK;
   assign BU1406_CLK = n169;
wire BU1406_SDIN;
   assign BU1406_SDIN = n79;
wire [2 : 0] BU1406_Q;
   assign n172 = BU1406_Q[0];
   assign n3419 = BU1406_Q[1];
wire BU1406_SCLR;
   assign BU1406_SCLR = n170;
C_SHIFT_FD_V7_0 #(
   "000"    /* c_ainit_val*/,
   0    /* c_enable_rlocs*/,
   5    /* c_fill_data*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   0    /* c_has_ce*/,
   0    /* c_has_d*/,
   0    /* c_has_lsb_2_msb*/,
   1    /* c_has_q*/,
   1    /* c_has_sclr*/,
   1    /* c_has_sdin*/,
   0    /* c_has_sdout*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
   1    /* c_shift_type*/,
   "000"    /* c_sinit_val*/,
   0    /* c_sync_enable*/,
   0    /* c_sync_priority*/,
   3    /* c_width*/
)
BU1406(
   .CLK(BU1406_CLK),
   .SDIN(BU1406_SDIN),
   .Q(BU1406_Q),
   .SCLR(BU1406_SCLR)
);

wire [1 : 0] BU1416_Q;
   assign n167 = BU1416_Q[0];
   assign n168 = BU1416_Q[1];
wire BU1416_CLK;
   assign BU1416_CLK = n169;
wire BU1416_CE;
   assign BU1416_CE = n172;
wire BU1416_SCLR;
   assign BU1416_SCLR = n170;
C_COUNTER_BINARY_V7_0 #(
   "00"    /* c_ainit_val*/,
   "01"    /* c_count_by*/,
   0    /* c_count_mode*/,
   "11"    /* c_count_to*/,
   0    /* c_enable_rlocs*/,
   0    /* c_has_aclr*/,
   0    /* c_has_ainit*/,
   0    /* c_has_aset*/,
   1    /* c_has_ce*/,
   0    /* c_has_iv*/,
   0    /* c_has_l*/,
   0    /* c_has_load*/,
   0    /* c_has_q_thresh0*/,
   0    /* c_has_q_thresh1*/,
   1    /* c_has_sclr*/,
   0    /* c_has_sinit*/,
   0    /* c_has_sset*/,
      0   /* c_has_thresh0*/,
      0   /* c_has_thresh1*/,
      0   /* c_has_up*/,
      0   /* c_load_enable*/,
      0   /* c_load_low*/,
      0   /* c_pipe_stages*/,
      1   /* c_restrict_count*/,
      "00"    /* c_sinit_val*/,
      0   /* c_sync_enable*/,
      0   /* c_sync_priority*/,
      "00"    /* c_thresh0_value*/,
      "00"    /* c_thresh1_value*/,
      0   /* c_thresh_early*/,
      2   /* c_width*/
    )
    BU1416(
      .Q(BU1416_Q),
      .CLK(BU1416_CLK),
      .CE(BU1416_CE),
      .SCLR(BU1416_SCLR)
    );

//synopsys translate_on

Endmodule

//Mixer (Multiplier , Xilinx Core)
////////////////////////////////////////////////////////////////////////////////
// Copyright (c) 1995-2006 Xilinx, Inc.  All rights reserved.
////////////////////////////////////////////////////////////////////////////////
//   ____  ____
//  /   /\/   /
// /___/  \  /    Vendor: Xilinx
// \   \   \/     Version: I.34
//  \   \         Application: netgen
//  /   /         Filename: mixer.v
// /___/   /\     Timestamp: Wed Oct 24 00:25:37 2007
// \   \  /  \
//  \___\/\___\
//
// Command		: -intstyle ise -w -sim -ofmt verilog F:\thesis1\_cg\mixer.ngc F:\thesis1\_cg\mixer.v
// Device	: 2vp30ff1152-5
// Input file: F:/thesis1/_cg/mixer.ngc
// Output file		: F:/thesis1/_cg/mixer.v
// # of Modules		: 1
// Design Name		: mixer
// Xilinx	: C:\Xilinx
//
// Purpose:
//     This verilog netlist is a verification model and uses simulation
//     primitives which may not represent the true implementation of the
//     device, however the netlist is functionally correct and should not
//     be modified. This file cannot be synthesized and should only be used
//     with supported simulation tools.
//
// Reference:
//     Development System Reference Guide, Chapter 23
//     Synthesis and Simulation Design Guide, Chapter 6
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1 ns/1 ps

module mixer (
  sclr, clk, a, b, p
);
  input sclr;
  input clk;
  input [11 : 0] a;
  input [11 : 0] b;
  output [11 : 0] p;

  // The synopsys directives "translate_off/translate_on" specified
  // below are supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity
  // synthesis tools. Ensure they are correct for your synthesis tool(s)

  // synopsys translate_off

  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_35_2 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/tmp ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_22_3 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_21_4 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_20_5 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_19_6 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_18_7 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_17_8 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_16_9 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_15_10 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_14_11 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_13_12 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_12_13 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_11_14 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_10_15 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_9_16 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_8_17 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_7_18 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_6_19 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_5_20 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_4_21 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_3_22 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_2_23 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_1_24 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_0_25 ;
  wire \BU2/N1 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Pdelay/N1 ;
  wire \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Pdelay/N0 ;
  wire NLW_VCC_P_UNCONNECTED;
  wire NLW_GND_G_UNCONNECTED;
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<35>_UNCONNECT
ED ;
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<34>_UNCONNECT
ED ;
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<33>_UNCONNECT
ED ;
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<32>_UNCONNECT
ED ;
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<31>_UNCONNECT
ED ;
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<30>_UNCONNECT
ED ;
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<29>_UNCONNECT
ED ;
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<28>_UNCONNECT
ED ;
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<27>_UNCONNECT
ED ;
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<26>_UNCONNECT
ED ;
```

```
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<25>_UNCONNECT
ED ;
  wire
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<24>_UNCONNECT
ED ;
  wire [11 : 0] a_26;
  wire [11 : 0] b_27;
  wire [22 : 0] \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> ;
  wire [0 : 0] \BU2/zero_detect ;
  assign
    a_26[11] = a[11],
    a_26[10] = a[10],
    a_26[9] = a[9],
    a_26[8] = a[8],
    a_26[7] = a[7],
    a_26[6] = a[6],
    a_26[5] = a[5],
    a_26[4] = a[4],
    a_26[3] = a[3],
    a_26[2] = a[2],
    a_26[1] = a[1],
    a_26[0] = a[0],
    b_27[11] = b[11],
    b_27[10] = b[10],
    b_27[9] = b[9],
    b_27[8] = b[8],
    b_27[7] = b[7],
    b_27[6] = b[6],
    b_27[5] = b[5],
    b_27[4] = b[4],
    b_27[3] = b[3],
    b_27[2] = b[2],
    b_27[1] = b[1],
    b_27[0] = b[0],
    p[11] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_35_2 ,
    p[10] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_22_3 ,
    p[9] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_21_4 ,
    p[8] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_20_5 ,
    p[7] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_19_6 ,
    p[6] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_18_7 ,
    p[5] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_17_8 ,
    p[4] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_16_9 ,
    p[3] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_15_10 ,
    p[2] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_14_11 ,
    p[1] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_13_12 ,
    p[0] = \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_12_13 ;
  VCC VCC_0 (
    .P(NLW_VCC_P_UNCONNECTED)
  );
  GND GND_1 (
    .G(NLW_GND_G_UNCONNECTED)
  );
  MULT18X18S \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>  (
    .C(clk),
    .CE(\BU2/N1 ),
    .R(sclr),
    .A({a_26[11], a_26[11], a_26[11], a_26[11], a_26[11], a_26[11], a_26[11], a_26[10], a_26[9], a_26[8], a_26[7], a_26[6], a_26[5],
a_26[4], a_26[3]
, a_26[2], a_26[1], a_26[0]}),
    .B({b_27[11], b_27[11], b_27[11], b_27[11], b_27[11], b_27[11], b_27[11], b_27[10], b_27[9], b_27[8], b_27[7], b_27[6],
b_27[5], b_27[4], b_27[3]
, b_27[2], b_27[1], b_27[0]}),

.P({\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<35>_UNCONN
ECTED ,
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<34>_UNCONNECT
ED ,
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<33>_UNCONNECT
ED ,
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<32>_UNCONNECT
ED ,
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<31>_UNCONNECT
ED ,
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<30>_UNCONNECT
ED ,
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<29>_UNCONNECT
ED ,
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<28>_UNCONNECT
ED ,
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<27>_UNCONNECT
ED ,
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<26>_UNCONNECT
ED ,
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<25>_UNCONNECT
ED ,
\NLW_BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Mmult_pi_mult<0><0>_P<24>_UNCONNECT
ED ,
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/tmp ,
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [22],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [21],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [20],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [19],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [18],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [17],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [16],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [15],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [14],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [13],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [12],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [11],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [10],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [9],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [8],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [7],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [6],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [5],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [4],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [3],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [2],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [1],
\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [0]})
  );
  defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_35 .INIT = 1'b0;
  FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_35  (
    .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/tmp ),
    .R(sclr),
    .C(clk),
    .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_35_2 )
  );
  defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_22 .INIT = 1'b0;
  FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_22  (
    .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [22]),
    .R(sclr),
    .C(clk),
    .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_22_3 )
  );
  defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_21 .INIT = 1'b0;
  FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_21  (
    .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [21]),
    .R(sclr),
    .C(clk),
    .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_21_4 )
  );
  defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_20 .INIT = 1'b0;
  FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_20  (
    .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [20]),
    .R(sclr),
    .C(clk),
    .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_20_5 )
```

```
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_19 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_19  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [19]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_19_6 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_18 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_18  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [18]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_18_7 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_17 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_17  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [17]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_17_8 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_16 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_16  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [16]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_16_9 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_15 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_15  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [15]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_15_10 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_14 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_14  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [14]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_14_11 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_13 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_13  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [13]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_13_12 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_12 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_12  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [12]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_12_13 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_11 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_11  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [11]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_11_14 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_10 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_10  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [10]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_10_15 )
```

```
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_9 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_9  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [9]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_9_16 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_8 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_8  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [8]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_8_17 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_7 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_7  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [7]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_7_18 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_6 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_6  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [6]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_6_19 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_5 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_5  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [5]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_5_20 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_4 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_4  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [4]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_4_21 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_3 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_3  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [3]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_3_22 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_2 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_2  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [2]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_2_23 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_1 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_1  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [1]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_1_24 )
);
defparam \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_0 .INIT = 1'b0;
FDR \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_0  (
  .D(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Mreg<0><0> [0]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/pi_Preg<0>_0_0_25 )
  );
  VCC \BU2/XST_VCC  (
    .P(\BU2/N1 )
  );
  GND \BU2/XST_GND  (
    .G(\BU2/zero_detect [0])
  );
  VCC \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Pdelay/XST_VCC  (
    .P(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Pdelay/N1 )
  );
  GND \BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Pdelay/XST_GND  (
    .G(\BU2/U0/gEMBEDDED_MULT.gEMB_MULTS_only.gMULT18.iMULT18/Pdelay/N0 )
  );

// synopsys translate_on

endmodule

// synopsys translate_off

`timescale  1 ps / 1 ps

module glbl ();

    parameter ROC_WIDTH = 100000;
    parameter TOC_WIDTH = 0;

    wire GSR;
    wire GTS;
    wire PRLD;

    reg GSR_int;
    reg GTS_int;
    reg PRLD_int;

//--------   JTAG Globals --------------
    wire JTAG_TDO_GLBL;
    wire JTAG_TCK_GLBL;
    wire JTAG_TDI_GLBL;
    wire JTAG_TMS_GLBL;
    wire JTAG_TRST_GLBL;

    reg JTAG_CAPTURE_GLBL;
    reg JTAG_RESET_GLBL;
    reg JTAG_SHIFT_GLBL;
    reg JTAG_UPDATE_GLBL;

    reg JTAG_SEL1_GLBL = 0;
    reg JTAG_SEL2_GLBL = 0 ;
    reg JTAG_SEL3_GLBL = 0;
    reg JTAG_SEL4_GLBL = 0;

    reg JTAG_USER_TDO1_GLBL = 1'bz;
    reg JTAG_USER_TDO2_GLBL = 1'bz;
    reg JTAG_USER_TDO3_GLBL = 1'bz;
    reg JTAG_USER_TDO4_GLBL = 1'bz;

    assign (weak1, weak0) GSR = GSR_int;
    assign (weak1, weak0) GTS = GTS_int;
    assign (weak1, weak0) PRLD = PRLD_int;

    initial begin
               GSR_int = 1'b1;
               PRLD_int = 1'b1;
               #(ROC_WIDTH)
               GSR_int = 1'b0;
               PRLD_int = 1'b0;
    end

    initial begin
		GTS_int = 1'b1;
		#(TOC_WIDTH)
		GTS_int = 1'b0;
   end

endmodule

// synopsys translate_on


// Anti-Aliasing Filter (FIR Filter, Xilinx Core)
/*******************************************************************************
*     This file is owned and controlled by Xilinx and must be used             *
*     solely for design, simulation, implementation and creation of            *
*     design files limited to Xilinx devices or technologies. Use              *
*     with non-Xilinx devices or technologies is expressly prohibited          *
*     and immediately terminates your license.                                 *
*                                                                              *
*     XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"            *
*     SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR                  *
*     XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION          *
*     AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION              *
*     OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS                *
*     IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,                  *
*     AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE         *
*     FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY                 *
*     WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE                  *
*     IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR           *
*     REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF          *
*     INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          *
*     FOR A PARTICULAR PURPOSE.                                                *
*                                                                              *
*     Xilinx products are not intended for use in life support                 *
*     appliances, devices, or systems. Use in such applications are            *
*     expressly prohibited.                                                    *
*                                                                              *
*     (c) Copyright 1995-2006 Xilinx, Inc.                                     *
*     All rights reserved.                                                     *
*******************************************************************************/
// The synopsys directives "translate_off/translate_on" specified below are
// supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity synthesis
// tools. Ensure they are correct for your synthesis tool(s).

// You must compile the wrapper file fir_da.v when simulating
// the core, fir_da. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Help".

`timescale 1ns/1ps

module fir_da(
		ND,
		RDY,
		CLK,
		RST,
		RFD,
		DIN,
		SEL_I,
		SEL_O,
		DOUT);


input ND;
output RDY;
input CLK;
input RST;
output RFD;
input [11 : 0] DIN;
output [1 : 0] SEL_I;
output [1 : 0] SEL_O;
output [23 : 0] DOUT;

// synopsys translate_off

   C_DA_FIR_V9_0 #(
                    12,         // c_baat
                    4,          // c_channels
                    0,          // c_coeff_type
                    12,         // c_coeff_width
                    0,          // c_data_type
                    12,         // c_data_width
                    0,          // c_enable_rlocs
                    0,          // c_filter_type
                    1,          // c_has_reset
                    1,          // c_has_sel_i
                    1,          // c_has_sel_o
                    0,          // c_has_sin_f
                    0,          // c_has_sin_r
                    0,          // c_has_sout_f
                    0,          // c_has_sout_r
                    12,         // c_latency
                    "fir_da.mif",         // c_mem_init_file
                    1,          // c_optimize
                    1,          // c_polyphase_factor
                    1,          // c_reg_output
                    0,          // c_reload
                    978,        // c_reload_delay
                    1,          // c_reload_mem_type
                    0,          // c_response
                    24,         // c_result_width
                    0,          // c_saturate
                    0,          // c_shape
                    118,        // c_taps
                    0,          // c_use_model_func
                    1)          // c_zpf
          inst (
                    .ND(ND),
                    .RDY(RDY),
                    .CLK(CLK),
                    .RST(RST),
                    .RFD(RFD),
                    .DIN(DIN),
                    .SEL_I(SEL_I),
                    .SEL_O(SEL_O),
                    .DOUT(DOUT),
                    .LD_DIN(),
                    .COEF_LD(),
                    .LD_WE(),
                    .DOUT_I(),
                    .DOUT_Q());


// synopsys translate_on

// FPGA Express black box declaration
// synopsys attribute fpga_dont_touch "true"
// synthesis attribute fpga_dont_touch of fir_da is "true"

// XST black box declaration
// box_type "black_box"
// synthesis attribute box_type of fir_da is "black_box"

endmodule


// Downsampler
`timescale 1ns / 1ps
module sync_fir(
input               clk,
                    clr,
input               [11:0]      fir_da_out_1,
input               [1:0]                   sel_o_1,
input               rdy_fir_da_1,
input       rfd_fir_da_2,
output      nd_fir_da_2,
output      [11:0]      fir_da_in_2);

parameter ptr_depth = 2;

reg         [5:0]                           count;
reg         [11:0]      stack [(2**(ptr_depth)-1):0];
reg         [ptr_depth-1:0]         rd_ptr,
                                    wr_ptr;
reg         rd,
            wr;
wire        empty;

assign fir_da_in_2 = stack[rd_ptr];
assign empty = (rd_ptr == wr_ptr)? 1'b1 : 1'b0;
assign nd_fir_da_2 = rd;

always@(posedge(clk))
begin
            if (clr)
                        count <= 0;
            else if (rdy_fir_da_1)
                        count <= count + 1;
end

always@(rdy_fir_da_1,count)
begin
            if((rdy_fir_da_1 == 1) && (count <= 3))
                        wr <= 1;
            else
                        wr <= 0;
end
always@(empty,rfd_fir_da_2)
begin
            if ((empty!=1) && (rfd_fir_da_2==1))
                        rd <= 1;
            else
                        rd <= 0;
end
always@(posedge(clk))
begin
            if (clr)
            begin
                        rd_ptr <= 0;
                        wr_ptr <= 0;
            end
            else if (rd & wr)
            begin
                        rd_ptr <= rd_ptr + 1;
                        wr_ptr <= wr_ptr + 1;
                        stack[wr_ptr] <= fir_da_out_1;
            end
            else if (rd)
            begin
                        rd_ptr <= rd_ptr + 1;
            end
            else if (wr)
            begin
                        stack[wr_ptr] <= fir_da_out_1;
                        wr_ptr <= wr_ptr + 1;
            end
end

endmodule
```

// Baseband Filter

```
/*******************************************************************************
*     This file is owned and controlled by Xilinx and must be used             *
*     solely for design, simulation, implementation and creation of            *
*     design files limited to Xilinx devices or technologies. Use              *
*     with non-Xilinx devices or technologies is expressly prohibited          *
*     and immediately terminates your license.                                 *
*                                                                              *
*     XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"            *
*     SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR                  *
*     XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION          *
*     AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION              *
*     OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS                 *
*     IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,                   *
*     AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE         *
*     FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY                 *
*     WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE                  *
*     IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR           *
*     REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF          *
*     INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          *
*     FOR A PARTICULAR PURPOSE.                                                *
*                                                                              *
*     Xilinx products are not intended for use in life support                 *
*     appliances, devices, or systems. Use in such applications are            *
*     expressly prohibited.                                                    *
*                                                                              *
*     (c) Copyright 1995-2006 Xilinx, Inc.                                     *
*     All rights reserved.                                                     *
*******************************************************************************/
// The synopsys directives "translate_off/translate_on" specified below are
// supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity synthesis
// tools. Ensure they are correct for your synthesis tool(s).

// You must compile the wrapper file fir_da_sec.v when simulating
// the core, fir_da_sec. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Help".

`timescale 1ns/1ps

module fir_da_sec(
        ND,
        RDY,
        CLK,
        RST,
        RFD,
        DIN,
        SEL_I,
        SEL_O,
        DOUT);


input ND;
output RDY;
input CLK;
input RST;
output RFD;
input [11 : 0] DIN;
output [1 : 0] SEL_I;
output [1 : 0] SEL_O;
output [24 : 0] DOUT;

// synopsys translate_off

   C_DA_FIR_V9_0 #(
            1,         // c_baat
            4,         // c_channels
            0,         // c_coeff_type
            12,        // c_coeff_width
            0,         // c_data_type
```

```
                        12,             // c_data_width
                        0,              // c_enable_rlocs
                        0,              // c_filter_type
                        1,              // c_has_reset
                        1,              // c_has_sel_i
                        1,              // c_has_sel_o
                        0,              // c_has_sin_f
                        0,              // c_has_sin_r
                        0,              // c_has_sout_f
                        0,              // c_has_sout_r
                        23,             // c_latency
                        "fir_da_sec.mif",       // c_mem_init_file
                        1,              // c_optimize
                        1,              // c_polyphase_factor
                        1,              // c_reg_output
                        0,              // c_reload
                        2002,           // c_reload_delay
                        1,              // c_reload_mem_type
                        0,              // c_response
                        25,             // c_result_width
                        0,              // c_saturate
                        0,              // c_shape
                        248,            // c_taps
                        0,              // c_use_model_func
                        1)              // c_zpf
            inst (
                        .ND(ND),
                        .RDY(RDY),
                        .CLK(CLK),
                        .RST(RST),
                        .RFD(RFD),
                        .DIN(DIN),
                        .SEL_I(SEL_I),
                        .SEL_O(SEL_O),
                        .DOUT(DOUT),
                        .LD_DIN(),
                        .COEF_LD(),
                        .LD_WE(),
                        .DOUT_I(),
                        .DOUT_Q());


// synopsys translate_on

// FPGA Express black box declaration
// synopsys attribute fpga_dont_touch "true"
// synthesis attribute fpga_dont_touch of fir_da_sec is "true"

// XST black box declaration
// box_type "black_box"
// synthesis attribute box_type of fir_da_sec is "black_box"

Endmodule

// Squaring Circuit 1
`timescale 1ns / 1ps
//`default_nettype none
module squaring_ckt(

input                   clk,
                        clr,
input                   [11:0]      fir2_out,
input                   [1:0]                   sel_o_2,
input                   rdy_fir_da_2,
output      reg         rdy_sqr_ckt,
output      reg         [23:0]      square_out);
//intermediate signals

reg                     [11:0]      fir2_out_reg;
reg                     [1:0]       sel_o_2_reg_a,
                                        sel_o_2_reg_b;

reg                                     rdy_sqr_ckt_a,
                                        rdy_sqr_ckt_b,
                                        rdy_sqr_ckt_c;

wire                [23:0]              mul_out;
reg                 [23:0]              add_a_in,
                                        add_b_in;
wire                [23:0]              add_out;

always @ (posedge clk)
begin
        if (clr)
        begin
                fir2_out_reg <= 0;
                sel_o_2_reg_a <= 0;
                rdy_sqr_ckt_a <= 0;
        end
        else if ((rdy_fir_da_2==1) && ((sel_o_2==2'd0)||(sel_o_2==2'd1)))
        begin
                fir2_out_reg  <= fir2_out;
                sel_o_2_reg_a <= sel_o_2;
                if (sel_o_2==2'd1)
                        rdy_sqr_ckt_a <= 1;
        end
        else
                rdy_sqr_ckt_a <= 0;
end
always @ (posedge clk)
begin
        if (clr)
        begin
                rdy_sqr_ckt_b <= 0;
                sel_o_2_reg_b <= 0;
        end
        else
        begin
                rdy_sqr_ckt_b <= rdy_sqr_ckt_a;
                sel_o_2_reg_b <= sel_o_2_reg_a;
        end

end
mul_sq MUL_SQ (
                .sclr(clr),
                .clk(clk),
                .a(fir2_out_reg),
                .b(fir2_out_reg),
                .p(mul_out)
);

always @ (posedge clk)
begin
        if (clr) begin
                add_a_in <= 0;
                add_b_in <= 0;
                rdy_sqr_ckt_c <= 0;
        end
        else
        begin
                case (sel_o_2_reg_b)
                        2'd0:add_a_in <= mul_out;
                        2'd1:add_b_in <= mul_out;
                endcase
                rdy_sqr_ckt_c <= rdy_sqr_ckt_b;
        end
end

assign add_out = add_a_in + add_b_in;
```

```
always @ (posedge clk)
begin
        if (clr)
        begin
                square_out <= 0;
                rdy_sqr_ckt <= 0;
        end
        else
        begin
                square_out <= add_out;
                rdy_sqr_ckt <= rdy_sqr_ckt_c;
        end
end
endmodule

// LUT Multipler (Xilinx Core)
////////////////////////////////////////////////////////////////////////////////
// Copyright (c) 1995-2006 Xilinx, Inc.  All rights reserved.
////////////////////////////////////////////////////////////////////////////////
//   ____  ____
//  /   /\/   /
// /___/  \  /    Vendor: Xilinx
// \   \   \/     Version: I.34
//  \   \         Application: netgen
//  /   /         Filename: mul_sq.v
// /___/   /\     Timestamp: Wed Oct 24 00:47:58 2007
// \   \  /  \
//  \___\/\___\
//
// Command        : -intstyle ise -w -sim -ofmt verilog F:\thesis1\_cg\mul_sq.ngc F:\thesis1\_cg\mul_sq.v
// Device   : 2vp30ff1152-5
// Input file: F:/thesis1/_cg/mul_sq.ngc
// Output file        : F:/thesis1/_cg/mul_sq.v
// # of Modules       : 1
// Design Name        : mul_sq
// Xilinx       : C:\Xilinx
//
// Purpose:
//     This verilog netlist is a verification model and uses simulation
//     primitives which may not represent the true implementation of the
//     device, however the netlist is functionally correct and should not
//     be modified. This file cannot be synthesized and should only be used
//     with supported simulation tools.
//
// Reference:
//     Development System Reference Guide, Chapter 23
//     Synthesis and Simulation Design Guide, Chapter 6
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1 ns/1 ps

module mul_sq (
  sclr, clk, a, b, p
);
  input sclr;
  input clk;
  input [11 : 0] a;
  input [11 : 0] b;
  output [23 : 0] p;

  // The synopsys directives "translate_off/translate_on" specified
  // below are supported by XST, FPGA Compiler II, Mentor Graphics and Synplicity
  // synthesis tools. Ensure they are correct for your synthesis tool(s)

  // synopsys translate_off

  wire \BU2/U0/gLUT.iLUT/lut_sig81 ;
  wire \BU2/N87 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig76 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig80 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig75 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig79 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig74 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig78 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig73 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig77 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig72 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig76 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig71 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig75 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig70 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig74 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig69 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig73 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig68 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig72 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig67 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig71 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig66 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig70 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig65 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig69 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig64 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig67 ;
  wire \BU2/N86 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig63 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig66 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig62 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig65 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig61 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig64 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig60 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig63 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig59 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig62 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig58 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig61 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig57 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig60 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig56 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig59 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig55 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig58 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig54 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig57 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig53 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig56 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig52 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig55 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig51 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig53 ;
  wire \BU2/N85 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig50 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig52 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig49 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig51 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig48 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig50 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig47 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig49 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig46 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig48 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig45 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig47 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig44 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig46 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig43 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig45 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig42 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig44 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig41 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig43 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig40 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig42 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig39 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig41 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig38 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig40 ;
  wire \BU2/N84 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig37 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig38 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig36 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig37 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig35 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig36 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig34 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig35 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig33 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig34 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig32 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig33 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig31 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig32 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig30 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig31 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig29 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig30 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig28 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig29 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig27 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig28 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig26 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig27 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig25 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig25 ;
  wire \BU2/N83 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig24 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig24 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig23 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig23 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig22 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig22 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig21 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig21 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig20 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig20 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig19 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig19 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig18 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig18 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig17 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig17 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig16 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig16 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig15 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig15 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig14 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig14 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig13 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig13 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig12 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig11 ;
  wire \BU2/N82 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig11 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig10 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig10 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig9 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig9 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig8 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig8 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig7 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig7 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig6 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig6 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig5 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig5 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig4 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig4 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig3 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig3 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig2 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig2 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig1_2 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig1 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig0 ;
  wire \BU2/U0/gLUT.iLUT/ma_sig0 ;
  wire \BU2/U0/gLUT.iLUT/lut_sig ;
  wire \BU2/U0/gLUT.iLUT/ma_sig ;
  wire \BU2/N81 ;
  wire \BU2/U0/gLUT.iLUT/N81 ;
  wire \BU2/U0/gLUT.iLUT/N80 ;
  wire \BU2/U0/gLUT.iLUT/N79 ;
  wire \BU2/U0/gLUT.iLUT/N78 ;
  wire \BU2/U0/gLUT.iLUT/N77 ;
  wire \BU2/U0/gLUT.iLUT/N76 ;
  wire \BU2/U0/gLUT.iLUT/N75 ;
  wire \BU2/U0/gLUT.iLUT/N74 ;
  wire \BU2/U0/gLUT.iLUT/N73 ;
  wire \BU2/U0/gLUT.iLUT/N72 ;
  wire \BU2/U0/gLUT.iLUT/N71 ;
  wire \BU2/U0/gLUT.iLUT/N70 ;
  wire \BU2/U0/gLUT.iLUT/N69 ;
  wire \BU2/U0/gLUT.iLUT/N68 ;
  wire \BU2/U0/gLUT.iLUT/N67 ;
  wire \BU2/N80 ;
  wire \BU2/U0/gLUT.iLUT/N64 ;
  wire \BU2/U0/gLUT.iLUT/N63 ;
  wire \BU2/U0/gLUT.iLUT/N62 ;
  wire \BU2/U0/gLUT.iLUT/N61 ;
  wire \BU2/U0/gLUT.iLUT/N60 ;
  wire \BU2/U0/gLUT.iLUT/N59 ;
  wire \BU2/U0/gLUT.iLUT/N58 ;
  wire \BU2/U0/gLUT.iLUT/N57 ;
  wire \BU2/U0/gLUT.iLUT/N56 ;
  wire \BU2/U0/gLUT.iLUT/N55 ;
  wire \BU2/U0/gLUT.iLUT/N54 ;
  wire \BU2/U0/gLUT.iLUT/N53 ;
  wire \BU2/U0/gLUT.iLUT/N52 ;
  wire \BU2/U0/gLUT.iLUT/N51 ;
  wire \BU2/U0/gLUT.iLUT/N50 ;
  wire \BU2/U0/gLUT.iLUT/N48 ;
  wire \BU2/U0/gLUT.iLUT/N47 ;
  wire \BU2/U0/gLUT.iLUT/N46 ;
  wire \BU2/U0/gLUT.iLUT/N45 ;
  wire \BU2/U0/gLUT.iLUT/N44 ;
  wire \BU2/U0/gLUT.iLUT/N43 ;
  wire \BU2/U0/gLUT.iLUT/N42 ;
  wire \BU2/U0/gLUT.iLUT/N41 ;
  wire \BU2/U0/gLUT.iLUT/N40 ;
  wire \BU2/U0/gLUT.iLUT/N39 ;
  wire \BU2/U0/gLUT.iLUT/N38 ;
  wire \BU2/U0/gLUT.iLUT/N37 ;
  wire \BU2/U0/gLUT.iLUT/N36 ;
  wire \BU2/U0/gLUT.iLUT/N34 ;
  wire \BU2/U0/gLUT.iLUT/N33 ;
  wire \BU2/U0/gLUT.iLUT/N32 ;
  wire \BU2/U0/gLUT.iLUT/N31 ;
  wire \BU2/U0/gLUT.iLUT/N30 ;
  wire \BU2/U0/gLUT.iLUT/N29 ;
  wire \BU2/U0/gLUT.iLUT/N28 ;
  wire \BU2/U0/gLUT.iLUT/N27 ;
  wire \BU2/U0/gLUT.iLUT/N26 ;
  wire \BU2/U0/gLUT.iLUT/N25 ;
  wire \BU2/U0/gLUT.iLUT/N24 ;
  wire \BU2/U0/gLUT.iLUT/N23 ;
  wire \BU2/U0/gLUT.iLUT/N22 ;
  wire \BU2/U0/gLUT.iLUT/N20 ;
  wire \BU2/U0/gLUT.iLUT/N19 ;
  wire \BU2/U0/gLUT.iLUT/N18 ;
  wire \BU2/U0/gLUT.iLUT/N17 ;
  wire \BU2/U0/gLUT.iLUT/N16 ;
  wire \BU2/U0/gLUT.iLUT/N15 ;
  wire \BU2/U0/gLUT.iLUT/N14 ;
  wire \BU2/U0/gLUT.iLUT/N13 ;
  wire \BU2/U0/gLUT.iLUT/N12 ;
  wire \BU2/U0/gLUT.iLUT/N11 ;
  wire \BU2/U0/gLUT.iLUT/N10 ;
  wire \BU2/U0/gLUT.iLUT/N9 ;
  wire \BU2/U0/gLUT.iLUT/N8 ;
  wire \BU2/N1 ;
  wire NLW_VCC_P_UNCONNECTED;
  wire NLW_GND_G_UNCONNECTED;
  wire [11 : 0] a_3;
  wire [11 : 0] b_4;
  wire [23 : 0] p_5;
  wire [12 : 0] \BU2/U0/gLUT.iLUT/pp_cout<5> ;
  wire [12 : 0] \BU2/U0/gLUT.iLUT/pp_cout<4> ;
  wire [12 : 0] \BU2/U0/gLUT.iLUT/pp_cout<3> ;
  wire [12 : 0] \BU2/U0/gLUT.iLUT/pp_cout<2> ;
  wire [12 : 0] \BU2/U0/gLUT.iLUT/pp_cout<1> ;
  wire [12 : 0] \BU2/U0/gLUT.iLUT/pp_cout<0> ;
  wire [15 : 0] \BU2/U0/gLUT.iLUT/Madd__add0004_cy ;
  wire [15 : 0] \BU2/U0/gLUT.iLUT/s2_add_out<1> ;
  wire [20 : 0] \BU2/U0/gLUT.iLUT/s2_add_out<0> ;
  wire [15 : 0] \BU2/U0/gLUT.iLUT/Madd__add0003_cy ;
  wire [15 : 0] \BU2/U0/gLUT.iLUT/s1_add_out<1> ;
  wire [13 : 0] \BU2/U0/gLUT.iLUT/pp_out_reg<5> ;
  wire [12 : 0] \BU2/U0/gLUT.iLUT/Madd__add0002_cy ;
  wire [13 : 2] \BU2/U0/gLUT.iLUT/pp_out_reg<4> ;
  wire [13 : 0] \BU2/U0/gLUT.iLUT/pp_out_reg<3> ;
  wire [12 : 0] \BU2/U0/gLUT.iLUT/Madd__add0001_cy ;
  wire [13 : 2] \BU2/U0/gLUT.iLUT/pp_out_reg<2> ;
  wire [15 : 4] \BU2/U0/gLUT.iLUT/s1_add_out<0> ;
  wire [13 : 0] \BU2/U0/gLUT.iLUT/pp_out_reg<1> ;
  wire [12 : 0] \BU2/U0/gLUT.iLUT/Madd__add0000_cy ;
  wire [13 : 2] \BU2/U0/gLUT.iLUT/pp_out_reg<0> ;
  wire [24 : 24] \BU2/U0/gLUT.iLUT/s5_add_out<0> ;
  wire [24 : 8] \BU2/U0/gLUT.iLUT/sum1<0> ;
  wire [0 : 0] \BU2/zero_detect ;
  assign
    a_3[11] = a[11],
    a_3[10] = a[10],
    a_3[9] = a[9],
    a_3[8] = a[8],
    a_3[7] = a[7],
    a_3[6] = a[6],
    a_3[5] = a[5],
    a_3[4] = a[4],
    a_3[3] = a[3],
    a_3[2] = a[2],
    a_3[1] = a[1],
    a_3[0] = a[0],
    b_4[11] = b[11],
    b_4[10] = b[10],
    b_4[9] = b[9],
    b_4[8] = b[8],
    b_4[7] = b[7],
    b_4[6] = b[6],
    b_4[5] = b[5],
    b_4[4] = b[4],
    b_4[3] = b[3],
    b_4[2] = b[2],
    b_4[1] = b[1],
    b_4[0] = b[0],
    p[23] = p_5[23],
    p[22] = p_5[22],
    p[21] = p_5[21],
    p[20] = p_5[20],
    p[19] = p_5[19],
    p[18] = p_5[18],
    p[17] = p_5[17],
    p[16] = p_5[16],
    p[15] = p_5[15],
    p[14] = p_5[14],
    p[13] = p_5[13],
    p[12] = p_5[12],
    p[11] = p_5[11],
    p[10] = p_5[10],
    p[9] = p_5[9],
    p[8] = p_5[8],
    p[7] = p_5[7],
    p[6] = p_5[6],
    p[5] = p_5[5],
    p[4] = p_5[4],
    p[3] = p_5[3],
    p[2] = p_5[2],
    p[1] = p_5[1],
    p[0] = p_5[0];
  VCC VCC_0 (
    .P(NLW_VCC_P_UNCONNECTED)
  );
  GND GND_1 (
    .G(NLW_GND_G_UNCONNECTED)
  );
  defparam \BU2/U0/gLUT.iLUT/lut_sig8111 .INIT = 8'hB7;
  LUT3 \BU2/U0/gLUT.iLUT/lut_sig8111  (
    .I0(b_4[10]),
    .I1(a_3[11]),
    .I2(b_4[11]),
    .O(\BU2/N87 )
  );
  defparam \BU2/U0/gLUT.iLUT/lut_sig6711 .INIT = 8'h48;
  LUT3 \BU2/U0/gLUT.iLUT/lut_sig6711  (
    .I0(b_4[9]),
    .I1(a_3[11]),
    .I2(b_4[8]),
    .O(\BU2/N86 )
  );
  defparam \BU2/U0/gLUT.iLUT/lut_sig5311 .INIT = 8'h48;
  LUT3 \BU2/U0/gLUT.iLUT/lut_sig5311  (
    .I0(b_4[7]),
    .I1(a_3[11]),
    .I2(b_4[6]),
    .O(\BU2/N85 )
  );
  defparam \BU2/U0/gLUT.iLUT/lut_sig4011 .INIT = 8'h48;
  LUT3 \BU2/U0/gLUT.iLUT/lut_sig4011  (
    .I0(b_4[5]),
    .I1(a_3[11]),
    .I2(b_4[4]),
    .O(\BU2/N84 )
  );
  defparam \BU2/U0/gLUT.iLUT/lut_sig2511 .INIT = 8'h48;
  LUT3 \BU2/U0/gLUT.iLUT/lut_sig2511  (
    .I0(b_4[3]),
    .I1(a_3[11]),
    .I2(b_4[2]),
    .O(\BU2/N83 )
  );
```

```
defparam \BU2/U0/gLUT.iLUT/lut_sig1111 .INIT = 8'h48;
LUT3 \BU2/U0/gLUT.iLUT/lut_sig1111  (
  .I0(b_4[1]),
  .I1(a_3[11]),
  .I2(b_4[0]),
  .O(\BU2/N82 )
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<15>1 .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<15>1  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [20]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [15]),
  .O(\BU2/N81 )
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<15>1 .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<15>1  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [15]),
  .O(\BU2/N80 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig781 .INIT = 16'h8777;
LUT4 \BU2/U0/gLUT.iLUT/lut_sig781  (
  .I0(b_4[10]),
  .I1(a_3[9]),
  .I2(a_3[8]),
  .I3(b_4[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig78 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig751 .INIT = 16'h8777;
LUT4 \BU2/U0/gLUT.iLUT/lut_sig751  (
  .I0(b_4[10]),
  .I1(a_3[6]),
  .I2(a_3[5]),
  .I3(b_4[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig75 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig701 .INIT = 16'h8777;
LUT4 \BU2/U0/gLUT.iLUT/lut_sig701  (
  .I0(b_4[10]),
  .I1(a_3[1]),
  .I2(a_3[0]),
  .I3(b_4[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig70 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig721 .INIT = 16'h8777;
LUT4 \BU2/U0/gLUT.iLUT/lut_sig721  (
  .I0(b_4[10]),
  .I1(a_3[3]),
  .I2(a_3[2]),
  .I3(b_4[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig72 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig711 .INIT = 16'h8777;
LUT4 \BU2/U0/gLUT.iLUT/lut_sig711  (
  .I0(b_4[10]),
  .I1(a_3[2]),
  .I2(a_3[1]),
  .I3(b_4[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig71 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig741 .INIT = 16'h8777;
LUT4 \BU2/U0/gLUT.iLUT/lut_sig741  (
  .I0(b_4[10]),
  .I1(a_3[5]),
  .I2(a_3[4]),
  .I3(b_4[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig74 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig731 .INIT = 16'h8777;
LUT4 \BU2/U0/gLUT.iLUT/lut_sig731  (
  .I0(b_4[10]),
  .I1(a_3[4]),
  .I2(a_3[3]),
  .I3(b_4[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig73 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig761 .INIT = 16'h8777;
LUT4 \BU2/U0/gLUT.iLUT/lut_sig761  (
  .I0(b_4[10]),
  .I1(a_3[7]),
  .I2(a_3[6]),
  .I3(b_4[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig76 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig771 .INIT = 16'h8777;
LUT4 \BU2/U0/gLUT.iLUT/lut_sig771  (
  .I0(b_4[10]),
  .I1(a_3[8]),
  .I2(a_3[7]),
  .I3(b_4[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig77 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig801 .INIT = 16'h8777;
LUT4 \BU2/U0/gLUT.iLUT/lut_sig801  (
  .I0(b_4[10]),
  .I1(a_3[11]),
  .I2(a_3[10]),
  .I3(b_4[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig80 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig791 .INIT = 16'h8777;
LUT4 \BU2/U0/gLUT.iLUT/lut_sig791  (
  .I0(b_4[10]),
  .I1(a_3[10]),
  .I2(a_3[9]),
  .I3(b_4[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig79 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig9_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig9_Result1  (
  .I0(b_4[1]),
  .I1(a_3[9]),
  .I2(b_4[0]),
  .I3(a_3[10]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig9 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig7_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig7_Result1  (
  .I0(b_4[1]),
  .I1(a_3[7]),
  .I2(b_4[0]),
  .I3(a_3[8]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig7 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig8_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig8_Result1  (
  .I0(b_4[1]),
  .I1(a_3[8]),
  .I2(b_4[0]),
  .I3(a_3[9]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig8 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig6_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig6_Result1  (
  .I0(b_4[1]),
  .I1(a_3[6]),
  .I2(b_4[0]),
  .I3(a_3[7]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig6 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig66_Result1 .INIT = 16'h7888;
```

```
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig66_Result1  (
  .I0(b_4[9]),
  .I1(a_3[10]),
  .I2(b_4[8]),
  .I3(a_3[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig66 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig65_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig65_Result1  (
  .I0(b_4[9]),
  .I1(a_3[9]),
  .I2(b_4[8]),
  .I3(a_3[10]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig65 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig63_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig63_Result1  (
  .I0(b_4[9]),
  .I1(a_3[7]),
  .I2(b_4[8]),
  .I3(a_3[8]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig63 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig64_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig64_Result1  (
  .I0(b_4[9]),
  .I1(a_3[8]),
  .I2(b_4[8]),
  .I3(a_3[9]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig64 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig62_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig62_Result1  (
  .I0(b_4[9]),
  .I1(a_3[6]),
  .I2(b_4[8]),
  .I3(a_3[7]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig62 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig61_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig61_Result1  (
  .I0(b_4[9]),
  .I1(a_3[5]),
  .I2(b_4[8]),
  .I3(a_3[6]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig61 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig60_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig60_Result1  (
  .I0(b_4[9]),
  .I1(a_3[4]),
  .I2(b_4[8]),
  .I3(a_3[5]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig60 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig59_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig59_Result1  (
  .I0(b_4[9]),
  .I1(a_3[3]),
  .I2(b_4[8]),
  .I3(a_3[4]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig59 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig5_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig5_Result1  (
  .I0(b_4[1]),
  .I1(a_3[5]),
  .I2(b_4[0]),
  .I3(a_3[6]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig5 )
```

```
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig58_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig58_Result1  (
  .I0(b_4[9]),
  .I1(a_3[2]),
  .I2(b_4[8]),
  .I3(a_3[3]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig58 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig57_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig57_Result1  (
  .I0(b_4[9]),
  .I1(a_3[1]),
  .I2(b_4[8]),
  .I3(a_3[2]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig57 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig56_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig56_Result1  (
  .I0(b_4[9]),
  .I1(a_3[0]),
  .I2(b_4[8]),
  .I3(a_3[1]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig56 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig51_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig51_Result1  (
  .I0(b_4[7]),
  .I1(a_3[9]),
  .I2(b_4[6]),
  .I3(a_3[10]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig51 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig52_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig52_Result1  (
  .I0(b_4[7]),
  .I1(a_3[10]),
  .I2(b_4[6]),
  .I3(a_3[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig52 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig50_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig50_Result1  (
  .I0(b_4[7]),
  .I1(a_3[8]),
  .I2(b_4[6]),
  .I3(a_3[9]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig50 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig4_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig4_Result1  (
  .I0(b_4[1]),
  .I1(a_3[4]),
  .I2(b_4[0]),
  .I3(a_3[5]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig4 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig49_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig49_Result1  (
  .I0(b_4[7]),
  .I1(a_3[7]),
  .I2(b_4[6]),
  .I3(a_3[8]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig49 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig47_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig47_Result1  (
  .I0(b_4[7]),
  .I1(a_3[5]),
  .I2(b_4[6]),
  .I3(a_3[6]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig47 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig48_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig48_Result1  (
  .I0(b_4[7]),
  .I1(a_3[6]),
  .I2(b_4[6]),
  .I3(a_3[7]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig48 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig46_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig46_Result1  (
  .I0(b_4[7]),
  .I1(a_3[4]),
  .I2(b_4[6]),
  .I3(a_3[5]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig46 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig45_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig45_Result1  (
  .I0(b_4[7]),
  .I1(a_3[3]),
  .I2(b_4[6]),
  .I3(a_3[4]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig45 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig44_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig44_Result1  (
  .I0(b_4[7]),
  .I1(a_3[2]),
  .I2(b_4[6]),
  .I3(a_3[3]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig44 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig42_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig42_Result1  (
  .I0(b_4[7]),
  .I1(a_3[0]),
  .I2(b_4[6]),
  .I3(a_3[1]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig42 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig43_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig43_Result1  (
  .I0(b_4[7]),
  .I1(a_3[1]),
  .I2(b_4[6]),
  .I3(a_3[2]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig43 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig3_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig3_Result1  (
  .I0(b_4[1]),
  .I1(a_3[3]),
  .I2(b_4[0]),
  .I3(a_3[4]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig3 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig38_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig38_Result1  (
  .I0(b_4[5]),
  .I1(a_3[10]),
  .I2(b_4[4]),
  .I3(a_3[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig38 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig37_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig37_Result1  (
  .I0(b_4[5]),
  .I1(a_3[9]),
  .I2(a_3[10]),
  .I3(b_4[4]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig37 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig36_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig36_Result1  (
  .I0(b_4[5]),
  .I1(a_3[8]),
  .I2(b_4[4]),
  .I3(a_3[9]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig36 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig35_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig35_Result1  (
  .I0(b_4[5]),
  .I1(a_3[7]),
  .I2(b_4[4]),
  .I3(a_3[8]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig35 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig34_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig34_Result1  (
  .I0(b_4[5]),
  .I1(a_3[6]),
  .I2(b_4[4]),
  .I3(a_3[7]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig34 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig32_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig32_Result1  (
  .I0(b_4[5]),
  .I1(a_3[4]),
  .I2(b_4[4]),
  .I3(a_3[5]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig32 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig31_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig31_Result1  (
  .I0(b_4[5]),
  .I1(a_3[3]),
  .I2(b_4[4]),
  .I3(a_3[4]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig31 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig33_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig33_Result1  (
  .I0(b_4[5]),
  .I1(a_3[5]),
  .I2(b_4[4]),
  .I3(a_3[6]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig33 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig30_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig30_Result1  (
  .I0(b_4[5]),
  .I1(a_3[2]),
  .I2(b_4[4]),
  .I3(a_3[3]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig30 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig2_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig2_Result1  (
  .I0(b_4[1]),
  .I1(a_3[2]),
  .I2(b_4[0]),
  .I3(a_3[3]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig2 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig28_Result1 .INIT = 16'h7888;
```

```
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig28_Result1  (
  .I0(b_4[5]),
  .I1(a_3[0]),
  .I2(b_4[4]),
  .I3(a_3[1]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig28 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig24_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig24_Result1  (
  .I0(b_4[3]),
  .I1(a_3[10]),
  .I2(b_4[2]),
  .I3(a_3[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig24 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig29_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig29_Result1  (
  .I0(b_4[5]),
  .I1(a_3[1]),
  .I2(b_4[4]),
  .I3(a_3[2]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig29 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig23_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig23_Result1  (
  .I0(b_4[3]),
  .I1(a_3[9]),
  .I2(a_3[10]),
  .I3(b_4[2]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig23 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig22_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig22_Result1  (
  .I0(b_4[3]),
  .I1(a_3[8]),
  .I2(b_4[2]),
  .I3(a_3[9]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig22 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig20_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig20_Result1  (
  .I0(b_4[3]),
  .I1(a_3[6]),
  .I2(b_4[2]),
  .I3(a_3[7]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig20 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig1_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig1_Result1  (
  .I0(b_4[1]),
  .I1(a_3[1]),
  .I2(b_4[0]),
  .I3(a_3[2]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig1_2 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig21_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig21_Result1  (
  .I0(b_4[3]),
  .I1(a_3[7]),
  .I2(b_4[2]),
  .I3(a_3[8]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig21 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig19_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig19_Result1  (
  .I0(b_4[3]),
  .I1(a_3[5]),
  .I2(b_4[2]),
  .I3(a_3[6]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig19 )
```

```
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig18_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig18_Result1  (
  .I0(b_4[3]),
  .I1(a_3[4]),
  .I2(b_4[2]),
  .I3(a_3[5]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig18 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig16_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig16_Result1  (
  .I0(b_4[3]),
  .I1(a_3[2]),
  .I2(b_4[2]),
  .I3(a_3[3]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig16 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig15_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig15_Result1  (
  .I0(b_4[3]),
  .I1(a_3[1]),
  .I2(b_4[2]),
  .I3(a_3[2]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig15 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig17_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig17_Result1  (
  .I0(b_4[3]),
  .I1(a_3[3]),
  .I2(b_4[2]),
  .I3(a_3[4]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig17 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig14_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig14_Result1  (
  .I0(b_4[3]),
  .I1(a_3[0]),
  .I2(b_4[2]),
  .I3(a_3[1]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig14 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig10_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig10_Result1  (
  .I0(b_4[1]),
  .I1(a_3[10]),
  .I2(b_4[0]),
  .I3(a_3[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig10 )
);
defparam \BU2/U0/gLUT.iLUT/Mxor_lut_sig0_Result1 .INIT = 16'h7888;
LUT4 \BU2/U0/gLUT.iLUT/Mxor_lut_sig0_Result1  (
  .I0(b_4[1]),
  .I1(a_3[0]),
  .I2(b_4[0]),
  .I3(a_3[1]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig0 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig251 .INIT = 8'h28;
LUT3 \BU2/U0/gLUT.iLUT/lut_sig251  (
  .I0(a_3[11]),
  .I1(b_4[2]),
  .I2(b_4[3]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig25 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig811 .INIT = 8'h9F;
LUT3 \BU2/U0/gLUT.iLUT/lut_sig811  (
  .I0(b_4[11]),
  .I1(b_4[10]),
  .I2(a_3[11]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig81 )
```

```
);
defparam \BU2/U0/gLUT.iLUT/lut_sig401 .INIT = 8'h28;
LUT3 \BU2/U0/gLUT.iLUT/lut_sig401  (
  .I0(a_3[11]),
  .I1(b_4[4]),
  .I2(b_4[5]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig40 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig531 .INIT = 8'h28;
LUT3 \BU2/U0/gLUT.iLUT/lut_sig531  (
  .I0(a_3[11]),
  .I1(b_4[6]),
  .I2(b_4[7]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig53 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig671 .INIT = 8'h28;
LUT3 \BU2/U0/gLUT.iLUT/lut_sig671  (
  .I0(a_3[11]),
  .I1(b_4[8]),
  .I2(b_4[9]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig67 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig111 .INIT = 8'h28;
LUT3 \BU2/U0/gLUT.iLUT/lut_sig111  (
  .I0(a_3[11]),
  .I1(b_4[0]),
  .I2(b_4[1]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig11 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig691 .INIT = 4'h7;
LUT2 \BU2/U0/gLUT.iLUT/lut_sig691  (
  .I0(a_3[0]),
  .I1(b_4[10]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig69 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig551 .INIT = 4'h8;
LUT2 \BU2/U0/gLUT.iLUT/lut_sig551  (
  .I0(a_3[0]),
  .I1(b_4[8]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig55 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig411 .INIT = 4'h8;
LUT2 \BU2/U0/gLUT.iLUT/lut_sig411  (
  .I0(a_3[0]),
  .I1(b_4[6]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig41 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig271 .INIT = 4'h8;
LUT2 \BU2/U0/gLUT.iLUT/lut_sig271  (
  .I0(a_3[0]),
  .I1(b_4[4]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig27 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig131 .INIT = 4'h8;
LUT2 \BU2/U0/gLUT.iLUT/lut_sig131  (
  .I0(a_3[0]),
  .I1(b_4[2]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig13 )
);
defparam \BU2/U0/gLUT.iLUT/lut_sig1 .INIT = 4'h8;
LUT2 \BU2/U0/gLUT.iLUT/lut_sig1  (
  .I0(a_3[0]),
  .I1(b_4[0]),
  .O(\BU2/U0/gLUT.iLUT/lut_sig )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[13].ppsub.stageMSB.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [12]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig81 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [13])
);
```

```
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[12].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [11]),
  .LI(\BU2/N87 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [12])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[12].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [11]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig76 ),
  .S(\BU2/N87 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [12])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[12].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[11]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig76 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[11].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [10]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig80 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [11])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[11].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [10]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig75 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig80 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [11])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[11].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[11]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig75 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[10].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [9]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig79 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [10])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[10].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [9]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig74 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig79 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [10])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[10].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[10]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig74 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[9].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [8]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig78 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [9])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[9].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [8]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig73 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig78 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [9])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[9].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[9]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig73 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[8].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [7]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig77 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [8])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[8].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [7]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig72 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig77 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [8])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[8].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[8]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig72 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[7].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [6]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig76 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [7])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[7].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [6]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig71 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig76 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [7])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[7].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[7]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig71 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[6].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [5]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig75 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [6])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[6].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [5]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig70 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig75 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [6])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[6].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[6]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig70 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[5].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [4]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig74 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [5])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[5].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [4]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig69 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig74 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [5])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[5].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[5]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig69 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[4].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [3]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig73 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [4])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[4].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [3]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig68 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig73 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [4])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[4].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[4]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig68 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[3].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [2]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig72 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [3])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[3].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [2]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig67 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig72 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [3])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[3].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[3]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig67 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[2].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [1]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig71 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [2])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[2].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [1]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig66 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig71 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [2])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[2].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[2]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig66 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[1].ppsub.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [0]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig70 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [1])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[1].ppsub.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<5> [0]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig65 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig70 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [1])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[1].ppsub.stageN.ma1  (
  .I0(b_4[10]),
  .I1(a_3[1]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig65 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[0].ppsub.stageLSB.xorcy1  (
  .CI(\BU2/N1 ),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig69 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [0])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[0].ppsub.stageLSB.muxcy1  (
  .CI(\BU2/N1 ),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig64 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig69 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<5> [0])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[5].carrychain[0].ppsub.stageLSB.ma1  (
  .I0(b_4[10]),
  .I1(a_3[0]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig64 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[13].ppadd.b_is_even.stageMSB.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [12]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig67 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [13])
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[12].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [11]),
  .LI(\BU2/N86 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [12])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[12].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [11]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig63 ),
  .S(\BU2/N86 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [12])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[12].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[11]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig63 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[11].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [10]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig66 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [11])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[11].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [10]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig62 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig66 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [11])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[11].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[10]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig62 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[10].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [9]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig65 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [10])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[10].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [9]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig61 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig65 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [10])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[10].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[9]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig61 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[9].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [8]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig64 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [9])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[9].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [8]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig60 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig64 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [9])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[9].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[8]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig60 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[8].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [7]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig63 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [8])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[8].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [7]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig59 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig63 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [8])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[8].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[7]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig59 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[7].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [6]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig62 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [7])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[7].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [6]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig58 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig62 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [7])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[7].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[6]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig58 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[6].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [5]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig61 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [6])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[6].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [5]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig57 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig61 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [6])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[6].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[5]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig57 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[5].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [4]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig60 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [5])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[5].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [4]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig56 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig60 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [5])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[5].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[4]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig56 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[4].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [3]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig59 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [4])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[4].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [3]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig55 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig59 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [4])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[4].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[3]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig55 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[3].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [2]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig58 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [3])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[3].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [2]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig54 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig58 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [3])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[3].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[2]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig54 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[2].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [1]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig57 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [2])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[2].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [1]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig53 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig57 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [2])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[2].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[1]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig53 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[1].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [0]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig56 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [1])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[1].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<4> [0]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig52 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig56 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [1])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[1].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[9]),
  .I1(a_3[0]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig52 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[0].ppadd.b_is_even.stageLSB.xorcy0  (
  .CI(\BU2/zero_detect [0]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig55 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [0])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[0].ppadd.b_is_even.stageLSB.muxcy00  (
  .CI(\BU2/zero_detect [0]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig51 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig55 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<4> [0])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[4].carrychain[0].ppadd.b_is_even.stageLSB.ma0  (
  .I0(b_4[8]),
  .I1(a_3[0]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig51 )
```

```
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[13].ppadd.b_is_even.stageMSB.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [12]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig53 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [13])
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[12].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [11]),
  .LI(\BU2/N85 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [12])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[12].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [11]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig50 ),
  .S(\BU2/N85 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [12])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[12].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[11]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig50 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[11].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [10]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig52 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [11])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[11].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [10]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig49 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig52 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [11])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[11].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[10]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig49 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[10].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [9]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig51 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [10])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[10].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [9]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig48 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig51 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [10])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[10].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[9]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig48 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[9].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [8]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig50 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [9])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[9].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [8]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig47 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig50 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [9])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[9].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[8]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig47 )
);
```

```
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[8].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [7]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig49 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [8])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[8].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [7]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig46 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig49 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [8])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[8].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[7]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig46 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[7].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [6]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig48 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [7])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[7].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [6]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig45 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig48 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [7])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[7].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[6]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig45 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[6].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [5]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig47 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [6])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[6].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [5]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig44 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig47 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [6])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[6].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[5]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig44 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[5].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [4]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig46 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [5])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[5].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [4]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig43 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig46 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [5])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[5].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[4]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig43 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[4].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [3]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig45 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [4])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[4].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [3]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig42 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig45 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [4])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[4].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[3]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig42 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[3].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [2]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig44 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [3])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[3].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [2]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig41 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig44 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [3])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[3].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[2]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig41 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[2].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [1]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig43 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [2])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[2].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [1]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig40 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig43 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [2])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[2].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[1]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig40 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[1].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [0]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig42 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [1])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[1].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<3> [0]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig39 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig42 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [1])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[1].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[7]),
  .I1(a_3[0]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig39 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[0].ppadd.b_is_even.stageLSB.xorcy0  (
  .CI(\BU2/zero_detect [0]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig41 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [0])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[0].ppadd.b_is_even.stageLSB.muxcy00  (
  .CI(\BU2/zero_detect [0]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig38 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig41 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<3> [0])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[3].carrychain[0].ppadd.b_is_even.stageLSB.ma0  (
  .I0(b_4[6]),
  .I1(a_3[0]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig38 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[13].ppadd.b_is_even.stageMSB.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [12]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig40 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [13])
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[12].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [11]),
  .LI(\BU2/N84 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [12])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[12].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [11]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig37 ),
  .S(\BU2/N84 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [12])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[12].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[11]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig37 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[11].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [10]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig38 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [11])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[11].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [10]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig36 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig38 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [11])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[11].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[10]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig36 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[10].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [9]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig37 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [10])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[10].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [9]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig35 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig37 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [10])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[10].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[9]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig35 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[9].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [8]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig36 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [9])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[9].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [8]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig34 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig36 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [9])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[9].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[8]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig34 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[8].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [7]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig35 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [8])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[8].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [7]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig33 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig35 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [8])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[8].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[7]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig33 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[7].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [6]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig34 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [7])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[7].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [6]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig32 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig34 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [7])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[7].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[6]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig32 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[6].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [5]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig33 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [6])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[6].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [5]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig31 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig33 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [6])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[6].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[5]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig31 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[5].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [4]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig32 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [5])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[5].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [4]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig30 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig32 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [5])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[5].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[4]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig30 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[4].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [3]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig31 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [4])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[4].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [3]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig29 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig31 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [4])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[4].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[3]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig29 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[3].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [2]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig30 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [3])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[3].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [2]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig28 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig30 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [3])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[3].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[2]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig28 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[2].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [1]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig29 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [2])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[2].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [1]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig27 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig29 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [2])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[2].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[1]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig27 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[1].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [0]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig28 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [1])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[1].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<2> [0]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig26 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig28 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [1])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[1].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[5]),
  .I1(a_3[0]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig26 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[0].ppadd.b_is_even.stageLSB.xorcy0  (
  .CI(\BU2/zero_detect [0]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig27 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [0])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[0].ppadd.b_is_even.stageLSB.muxcy00  (
  .CI(\BU2/zero_detect [0]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig25 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig27 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<2> [0])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[2].carrychain[0].ppadd.b_is_even.stageLSB.ma0  (
  .I0(b_4[4]),
  .I1(a_3[0]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig25 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[13].ppadd.b_is_even.stageMSB.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [12]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig25 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [13])
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[12].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [11]),
  .LI(\BU2/N83 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [12])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[12].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [11]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig24 ),
  .S(\BU2/N83 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [12])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[12].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[11]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig24 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[11].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [10]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig24 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [11])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[11].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [10]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig23 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig24 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [11])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[11].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[10]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig23 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[10].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [9]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig23 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [10])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[10].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [9]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig22 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig23 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [10])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[10].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[9]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig22 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[9].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [8]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig22 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [9])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[9].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [8]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig21 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig22 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [9])
```

```
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[9].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[8]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig21 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[8].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [7]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig21 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [8])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[8].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [7]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig20 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig21 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [8])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[8].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[7]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig20 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[7].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [6]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig20 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [7])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[7].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [6]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig19 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig20 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [7])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[7].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[6]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig19 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[6].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [5]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig19 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [6])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[6].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [5]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig18 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig19 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [6])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[6].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[5]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig18 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[5].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [4]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig18 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [5])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[5].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [4]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig17 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig18 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [5])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[5].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[4]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig17 )
);
```

```
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[4].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [3]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig17 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [4])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[4].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [3]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig16 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig17 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [4])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[4].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[3]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig16 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[3].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [2]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig16 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [3])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[3].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [2]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig15 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig16 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [3])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[3].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[2]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig15 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[2].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [1]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig15 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [2])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[2].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [1]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig14 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig15 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [2])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[2].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[1]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig14 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[1].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [0]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig14 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [1])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[1].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<1> [0]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig13 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig14 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [1])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[1].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[3]),
  .I1(a_3[0]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig13 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[0].ppadd.b_is_even.stageLSB.xorcy0  (
  .CI(\BU2/zero_detect [0]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig13 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [0])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[0].ppadd.b_is_even.stageLSB.muxcy00  (
  .CI(\BU2/zero_detect [0]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig12 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig13 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<1> [0])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[1].carrychain[0].ppadd.b_is_even.stageLSB.ma0  (
  .I0(b_4[2]),
  .I1(a_3[0]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig12 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[13].ppadd.b_is_even.stageMSB.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [12]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig11 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [13])
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[12].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [11]),
  .LI(\BU2/N82 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [12])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[12].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [11]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig11 ),
  .S(\BU2/N82 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [12])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[12].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[11]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig11 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[11].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [10]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig10 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [11])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[11].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [10]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig10 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig10 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [11])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[11].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[10]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig10 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[10].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [9]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig9 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [10])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[10].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [9]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig9 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig9 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [10])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[10].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[9]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig9 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[9].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [8]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig8 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [9])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[9].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [8]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig8 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig8 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [9])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[9].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[8]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig8 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[8].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [7]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig7 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [8])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[8].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [7]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig7 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig7 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [8])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[8].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[7]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig7 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[7].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [6]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig6 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [7])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[7].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [6]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig6 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig6 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [7])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[7].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[6]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig6 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[6].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [5]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig5 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [6])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[6].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [5]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig5 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig5 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [6])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[6].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[5]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig5 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[5].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [4]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig4 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [5])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[5].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [4]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig4 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig4 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [5])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[5].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[4]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig4 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[4].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [3]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig3 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [4])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[4].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [3]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig3 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig3 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [4])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[4].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[3]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig3 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[3].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [2]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig2 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [3])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[3].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [2]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig2 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig2 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [3])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[3].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[2]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig2 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[2].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [1]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig1_2 ),
  .O(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [2])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[2].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [1]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig1 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig1_2 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [2])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[2].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[1]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig1 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[1].ppadd.b_is_even.stageN.xorcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [0]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig0 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [1])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[1].ppadd.b_is_even.stageN.muxcy0  (
  .CI(\BU2/U0/gLUT.iLUT/pp_cout<0> [0]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig0 ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig0 ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [1])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[1].ppadd.b_is_even.stageN.ma0  (
  .I0(b_4[1]),
  .I1(a_3[0]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig0 )
);
XORCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[0].ppadd.b_is_even.stageLSB.xorcy0  (
  .CI(\BU2/zero_detect [0]),
  .LI(\BU2/U0/gLUT.iLUT/lut_sig ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [0])
);
MUXCY \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[0].ppadd.b_is_even.stageLSB.muxcy00  (
  .CI(\BU2/zero_detect [0]),
  .DI(\BU2/U0/gLUT.iLUT/ma_sig ),
  .S(\BU2/U0/gLUT.iLUT/lut_sig ),
  .O(\BU2/U0/gLUT.iLUT/pp_cout<0> [0])
);
MULT_AND \BU2/U0/gLUT.iLUT/NxM_mult.ppgen[0].carrychain[0].ppadd.b_is_even.stageLSB.ma0  (
  .I0(b_4[0]),
  .I1(a_3[0]),
  .LO(\BU2/U0/gLUT.iLUT/ma_sig )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<16>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [15]),
  .LI(\BU2/U0/gLUT.iLUT/N81 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [24])
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<15>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [14]),
  .LI(\BU2/N81 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [23])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<15>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [14]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [20]),
  .S(\BU2/N81 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [15])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<15> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<15>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [20]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [15]),
  .O(\BU2/U0/gLUT.iLUT/N81 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<14>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [13]),
  .LI(\BU2/U0/gLUT.iLUT/N80 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [22])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<14>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [13]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [20]),
  .S(\BU2/U0/gLUT.iLUT/N80 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [14])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<14> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<14>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [20]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [14]),
  .O(\BU2/U0/gLUT.iLUT/N80 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<13>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [12]),
  .LI(\BU2/U0/gLUT.iLUT/N79 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [21])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<13>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [12]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [20]),
  .S(\BU2/U0/gLUT.iLUT/N79 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [13])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<13> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<13>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [20]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [13]),
  .O(\BU2/U0/gLUT.iLUT/N79 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<12>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [11]),
  .LI(\BU2/U0/gLUT.iLUT/N78 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [20])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<12>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [11]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [20]),
  .S(\BU2/U0/gLUT.iLUT/N78 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [12])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<12> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<12>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [20]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [12]),
  .O(\BU2/U0/gLUT.iLUT/N78 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<11>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [10]),
  .LI(\BU2/U0/gLUT.iLUT/N77 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [19])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<11>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [10]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [19]),
  .S(\BU2/U0/gLUT.iLUT/N77 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [11])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<11> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<11>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [19]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [11]),
  .O(\BU2/U0/gLUT.iLUT/N77 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<10>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [9]),
  .LI(\BU2/U0/gLUT.iLUT/N76 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [18])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<10>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [9]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [18]),
  .S(\BU2/U0/gLUT.iLUT/N76 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [10])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<10> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<10>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [18]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [10]),
  .O(\BU2/U0/gLUT.iLUT/N76 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<9>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [8]),
  .LI(\BU2/U0/gLUT.iLUT/N75 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [17])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<9>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [8]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [17]),
  .S(\BU2/U0/gLUT.iLUT/N75 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [9])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<9> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<9>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [17]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [9]),
  .O(\BU2/U0/gLUT.iLUT/N75 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<8>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [7]),
  .LI(\BU2/U0/gLUT.iLUT/N74 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [16])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<8>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [7]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [16]),
  .S(\BU2/U0/gLUT.iLUT/N74 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [8])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<8> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<8>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [16]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [8]),
  .O(\BU2/U0/gLUT.iLUT/N74 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<7>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [6]),
  .LI(\BU2/U0/gLUT.iLUT/N73 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [15])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<7>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [6]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [15]),
  .S(\BU2/U0/gLUT.iLUT/N73 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [7])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<7> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<7>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [15]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [7]),
  .O(\BU2/U0/gLUT.iLUT/N73 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<6>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [5]),
  .LI(\BU2/U0/gLUT.iLUT/N72 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [14])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<6>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [5]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [14]),
  .S(\BU2/U0/gLUT.iLUT/N72 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [6])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<6> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<6>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [14]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [6]),
  .O(\BU2/U0/gLUT.iLUT/N72 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<5>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [4]),
  .LI(\BU2/U0/gLUT.iLUT/N71 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [13])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<5>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [4]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [13]),
  .S(\BU2/U0/gLUT.iLUT/N71 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [5])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<5> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<5>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [13]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [5]),
  .O(\BU2/U0/gLUT.iLUT/N71 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<4>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [3]),
  .LI(\BU2/U0/gLUT.iLUT/N70 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [12])
);
```

```
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<4>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [3]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [12]),
  .S(\BU2/U0/gLUT.iLUT/N70 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [4])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<4> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<4>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [12]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [4]),
  .O(\BU2/U0/gLUT.iLUT/N70 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<3>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [2]),
  .LI(\BU2/U0/gLUT.iLUT/N69 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [11])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<3>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [2]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [11]),
  .S(\BU2/U0/gLUT.iLUT/N69 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [3])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<3> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<3>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [11]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [3]),
  .O(\BU2/U0/gLUT.iLUT/N69 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<2>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [1]),
  .LI(\BU2/U0/gLUT.iLUT/N68 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [10])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<2>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [1]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [10]),
  .S(\BU2/U0/gLUT.iLUT/N68 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [2])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<2> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<2>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [10]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [2]),
  .O(\BU2/U0/gLUT.iLUT/N68 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0004_xor<1>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [0]),
  .LI(\BU2/U0/gLUT.iLUT/N67 ),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [9])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<1>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [0]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [9]),
  .S(\BU2/U0/gLUT.iLUT/N67 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [1])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<1> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<1>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [9]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [1]),
  .O(\BU2/U0/gLUT.iLUT/N67 )
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0004_cy<0>  (
  .CI(\BU2/zero_detect [0]),
  .DI(\BU2/U0/gLUT.iLUT/s2_add_out<0> [8]),
  .S(\BU2/U0/gLUT.iLUT/sum1<0> [8]),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0004_cy [0])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0004_lut<0> .INIT = 4'h6;
```

```
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0004_lut<0>  (
  .I0(\BU2/U0/gLUT.iLUT/s2_add_out<0> [8]),
  .I1(\BU2/U0/gLUT.iLUT/s2_add_out<1> [0]),
  .O(\BU2/U0/gLUT.iLUT/sum1<0> [8])
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<16>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [15]),
  .LI(\BU2/U0/gLUT.iLUT/N64 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [20])
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<15>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [14]),
  .LI(\BU2/N80 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [19])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<15>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [14]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15]),
  .S(\BU2/N80 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [15])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<15> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<15>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [15]),
  .O(\BU2/U0/gLUT.iLUT/N64 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<14>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [13]),
  .LI(\BU2/U0/gLUT.iLUT/N63 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [18])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<14>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [13]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15]),
  .S(\BU2/U0/gLUT.iLUT/N63 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [14])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<14> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<14>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [14]),
  .O(\BU2/U0/gLUT.iLUT/N63 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<13>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [12]),
  .LI(\BU2/U0/gLUT.iLUT/N62 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [17])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<13>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [12]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15]),
  .S(\BU2/U0/gLUT.iLUT/N62 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [13])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<13> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<13>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [13]),
  .O(\BU2/U0/gLUT.iLUT/N62 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<12>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [11]),
  .LI(\BU2/U0/gLUT.iLUT/N61 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [16])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<12>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [11]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15]),
  .S(\BU2/U0/gLUT.iLUT/N61 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [12])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<12> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<12>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [12]),
  .O(\BU2/U0/gLUT.iLUT/N61 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<11>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [10]),
  .LI(\BU2/U0/gLUT.iLUT/N60 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [15])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<11>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [10]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15]),
  .S(\BU2/U0/gLUT.iLUT/N60 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [11])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<11> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<11>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [11]),
  .O(\BU2/U0/gLUT.iLUT/N60 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<10>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [9]),
  .LI(\BU2/U0/gLUT.iLUT/N59 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [14])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<10>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [9]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [14]),
  .S(\BU2/U0/gLUT.iLUT/N59 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [10])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<10> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<10>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [14]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [10]),
  .O(\BU2/U0/gLUT.iLUT/N59 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<9>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [8]),
  .LI(\BU2/U0/gLUT.iLUT/N58 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [13])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<9>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [8]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [13]),
  .S(\BU2/U0/gLUT.iLUT/N58 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [9])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<9> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<9>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [13]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [9]),
  .O(\BU2/U0/gLUT.iLUT/N58 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<8>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [7]),
  .LI(\BU2/U0/gLUT.iLUT/N57 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [12])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<8>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [7]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [12]),
  .S(\BU2/U0/gLUT.iLUT/N57 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [8])
);
```

```
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<8> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<8>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [12]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [8]),
  .O(\BU2/U0/gLUT.iLUT/N57 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<7>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [6]),
  .LI(\BU2/U0/gLUT.iLUT/N56 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [11])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<7>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [6]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [11]),
  .S(\BU2/U0/gLUT.iLUT/N56 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [7])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<7> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<7>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [11]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [7]),
  .O(\BU2/U0/gLUT.iLUT/N56 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<6>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [5]),
  .LI(\BU2/U0/gLUT.iLUT/N55 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [10])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<6>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [5]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [10]),
  .S(\BU2/U0/gLUT.iLUT/N55 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [6])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<6> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<6>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [10]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [6]),
  .O(\BU2/U0/gLUT.iLUT/N55 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<5>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [4]),
  .LI(\BU2/U0/gLUT.iLUT/N54 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [9])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<5>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [4]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [9]),
  .S(\BU2/U0/gLUT.iLUT/N54 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [5])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<5> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<5>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [9]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [5]),
  .O(\BU2/U0/gLUT.iLUT/N54 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<4>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [3]),
  .LI(\BU2/U0/gLUT.iLUT/N53 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [8])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<4>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [3]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [8]),
  .S(\BU2/U0/gLUT.iLUT/N53 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [4])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<4> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<4>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [8]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [4]),
  .O(\BU2/U0/gLUT.iLUT/N53 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<3>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [2]),
  .LI(\BU2/U0/gLUT.iLUT/N52 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [7])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<3>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [2]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [7]),
  .S(\BU2/U0/gLUT.iLUT/N52 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [3])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<3> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<3>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [7]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [3]),
  .O(\BU2/U0/gLUT.iLUT/N52 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<2>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [1]),
  .LI(\BU2/U0/gLUT.iLUT/N51 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [6])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<2>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [1]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [6]),
  .S(\BU2/U0/gLUT.iLUT/N51 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [2])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<2> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<2>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [6]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [2]),
  .O(\BU2/U0/gLUT.iLUT/N51 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0003_xor<1>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [0]),
  .LI(\BU2/U0/gLUT.iLUT/N50 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [5])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<1>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [0]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [5]),
  .S(\BU2/U0/gLUT.iLUT/N50 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [1])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<1> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<1>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [5]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [1]),
  .O(\BU2/U0/gLUT.iLUT/N50 )
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0003_cy<0>  (
  .CI(\BU2/zero_detect [0]),
  .DI(\BU2/U0/gLUT.iLUT/s1_add_out<0> [4]),
  .S(\BU2/U0/gLUT.iLUT/s2_add_out<0> [4]),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0003_cy [0])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0003_lut<0> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0003_lut<0>  (
  .I0(\BU2/U0/gLUT.iLUT/s1_add_out<0> [4]),
  .I1(\BU2/U0/gLUT.iLUT/s1_add_out<1> [0]),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [4])
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<13>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [12]),
  .LI(\BU2/U0/gLUT.iLUT/N48 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [15])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<13> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<13>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [13]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [13]),
  .O(\BU2/U0/gLUT.iLUT/N48 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<12>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [11]),
  .LI(\BU2/U0/gLUT.iLUT/N47 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [14])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<12>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [11]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [13]),
  .S(\BU2/U0/gLUT.iLUT/N47 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [12])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<12> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<12>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [13]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [12]),
  .O(\BU2/U0/gLUT.iLUT/N47 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<11>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [10]),
  .LI(\BU2/U0/gLUT.iLUT/N46 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [13])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<11>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [10]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [13]),
  .S(\BU2/U0/gLUT.iLUT/N46 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [11])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<11> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<11>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [13]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [11]),
  .O(\BU2/U0/gLUT.iLUT/N46 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<10>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [9]),
  .LI(\BU2/U0/gLUT.iLUT/N45 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [12])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<10>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [9]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [12]),
  .S(\BU2/U0/gLUT.iLUT/N45 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [10])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<10> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<10>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [12]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [10]),
  .O(\BU2/U0/gLUT.iLUT/N45 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<9>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [8]),
  .LI(\BU2/U0/gLUT.iLUT/N44 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [11])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<9>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [8]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [11]),
  .S(\BU2/U0/gLUT.iLUT/N44 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [9])
);
```

```
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<9> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<9>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [11]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [9]),
  .O(\BU2/U0/gLUT.iLUT/N44 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<8>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [7]),
  .LI(\BU2/U0/gLUT.iLUT/N43 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [10])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<8>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [7]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [10]),
  .S(\BU2/U0/gLUT.iLUT/N43 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [8])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<8> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<8>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [10]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [8]),
  .O(\BU2/U0/gLUT.iLUT/N43 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<7>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [6]),
  .LI(\BU2/U0/gLUT.iLUT/N42 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [9])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<7>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [6]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [9]),
  .S(\BU2/U0/gLUT.iLUT/N42 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [7])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<7> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<7>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [9]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [7]),
  .O(\BU2/U0/gLUT.iLUT/N42 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<6>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [5]),
  .LI(\BU2/U0/gLUT.iLUT/N41 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [8])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<6>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [5]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [8]),
  .S(\BU2/U0/gLUT.iLUT/N41 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [6])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<6> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<6>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [8]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [6]),
  .O(\BU2/U0/gLUT.iLUT/N41 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<5>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [4]),
  .LI(\BU2/U0/gLUT.iLUT/N40 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [7])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<5>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [4]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [7]),
  .S(\BU2/U0/gLUT.iLUT/N40 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [5])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<5> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<5>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [7]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [5]),
  .O(\BU2/U0/gLUT.iLUT/N40 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<4>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [3]),
  .LI(\BU2/U0/gLUT.iLUT/N39 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [6])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<4>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [3]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [6]),
  .S(\BU2/U0/gLUT.iLUT/N39 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [4])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<4> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<4>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [6]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [4]),
  .O(\BU2/U0/gLUT.iLUT/N39 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<3>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [2]),
  .LI(\BU2/U0/gLUT.iLUT/N38 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [5])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<3>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [2]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [5]),
  .S(\BU2/U0/gLUT.iLUT/N38 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [3])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<3> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<3>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [5]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [3]),
  .O(\BU2/U0/gLUT.iLUT/N38 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<2>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [1]),
  .LI(\BU2/U0/gLUT.iLUT/N37 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [4])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<2>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [1]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [4]),
  .S(\BU2/U0/gLUT.iLUT/N37 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [2])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<2> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<2>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [4]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [2]),
  .O(\BU2/U0/gLUT.iLUT/N37 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0002_xor<1>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [0]),
  .LI(\BU2/U0/gLUT.iLUT/N36 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [3])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<1>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [0]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [3]),
  .S(\BU2/U0/gLUT.iLUT/N36 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [1])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<1> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<1>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [3]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [1]),
  .O(\BU2/U0/gLUT.iLUT/N36 )
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0002_cy<0>  (
  .CI(\BU2/zero_detect [0]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [2]),
  .S(\BU2/U0/gLUT.iLUT/s2_add_out<1> [2]),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0002_cy [0])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0002_lut<0> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0002_lut<0>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<4> [2]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<5> [0]),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<1> [2])
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<13>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [12]),
  .LI(\BU2/U0/gLUT.iLUT/N34 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [15])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<13> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<13>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [13]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [13]),
  .O(\BU2/U0/gLUT.iLUT/N34 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<12>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [11]),
  .LI(\BU2/U0/gLUT.iLUT/N33 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [14])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<12>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [11]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [13]),
  .S(\BU2/U0/gLUT.iLUT/N33 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [12])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<12> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<12>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [13]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [12]),
  .O(\BU2/U0/gLUT.iLUT/N33 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<11>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [10]),
  .LI(\BU2/U0/gLUT.iLUT/N32 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [13])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<11>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [10]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [13]),
  .S(\BU2/U0/gLUT.iLUT/N32 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [11])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<11> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<11>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [13]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [11]),
  .O(\BU2/U0/gLUT.iLUT/N32 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<10>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [9]),
  .LI(\BU2/U0/gLUT.iLUT/N31 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [12])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<10>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [9]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [12]),
  .S(\BU2/U0/gLUT.iLUT/N31 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [10])
);
```

```
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<10> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<10>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [12]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [10]),
  .O(\BU2/U0/gLUT.iLUT/N31 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<9>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [8]),
  .LI(\BU2/U0/gLUT.iLUT/N30 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [11])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<9>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [8]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [11]),
  .S(\BU2/U0/gLUT.iLUT/N30 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [9])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<9> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<9>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [11]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [9]),
  .O(\BU2/U0/gLUT.iLUT/N30 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<8>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [7]),
  .LI(\BU2/U0/gLUT.iLUT/N29 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [10])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<8>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [7]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [10]),
  .S(\BU2/U0/gLUT.iLUT/N29 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [8])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<8> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<8>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [10]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [8]),
  .O(\BU2/U0/gLUT.iLUT/N29 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<7>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [6]),
  .LI(\BU2/U0/gLUT.iLUT/N28 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [9])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<7>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [6]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [9]),
  .S(\BU2/U0/gLUT.iLUT/N28 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [7])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<7> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<7>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [9]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [7]),
  .O(\BU2/U0/gLUT.iLUT/N28 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<6>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [5]),
  .LI(\BU2/U0/gLUT.iLUT/N27 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [8])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<6>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [5]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [8]),
  .S(\BU2/U0/gLUT.iLUT/N27 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [6])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<6> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<6>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [8]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [6]),
  .O(\BU2/U0/gLUT.iLUT/N27 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<5>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [4]),
  .LI(\BU2/U0/gLUT.iLUT/N26 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [7])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<5>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [4]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [7]),
  .S(\BU2/U0/gLUT.iLUT/N26 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [5])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<5> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<5>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [7]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [5]),
  .O(\BU2/U0/gLUT.iLUT/N26 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<4>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [3]),
  .LI(\BU2/U0/gLUT.iLUT/N25 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [6])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<4>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [3]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [6]),
  .S(\BU2/U0/gLUT.iLUT/N25 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [4])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<4> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<4>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [6]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [4]),
  .O(\BU2/U0/gLUT.iLUT/N25 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<3>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [2]),
  .LI(\BU2/U0/gLUT.iLUT/N24 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [5])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<3>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [2]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [5]),
  .S(\BU2/U0/gLUT.iLUT/N24 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [3])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<3> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<3>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [5]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [3]),
  .O(\BU2/U0/gLUT.iLUT/N24 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<2>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [1]),
  .LI(\BU2/U0/gLUT.iLUT/N23 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [4])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<2>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [1]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [4]),
  .S(\BU2/U0/gLUT.iLUT/N23 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [2])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<2> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<2>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [4]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [2]),
  .O(\BU2/U0/gLUT.iLUT/N23 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0001_xor<1>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [0]),
  .LI(\BU2/U0/gLUT.iLUT/N22 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [3])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<1>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [0]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [3]),
  .S(\BU2/U0/gLUT.iLUT/N22 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [1])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<1> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<1>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [3]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [1]),
  .O(\BU2/U0/gLUT.iLUT/N22 )
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0001_cy<0>  (
  .CI(\BU2/zero_detect [0]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [2]),
  .S(\BU2/U0/gLUT.iLUT/s1_add_out<1> [2]),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0001_cy [0])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0001_lut<0> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0001_lut<0>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<2> [2]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<3> [0]),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<1> [2])
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<13>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [12]),
  .LI(\BU2/U0/gLUT.iLUT/N20 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [15])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<13> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<13>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [13]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [13]),
  .O(\BU2/U0/gLUT.iLUT/N20 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<12>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [11]),
  .LI(\BU2/U0/gLUT.iLUT/N19 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [14])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<12>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [11]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [13]),
  .S(\BU2/U0/gLUT.iLUT/N19 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [12])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<12> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<12>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [13]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [12]),
  .O(\BU2/U0/gLUT.iLUT/N19 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<11>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [10]),
  .LI(\BU2/U0/gLUT.iLUT/N18 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [13])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<11>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [10]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [13]),
  .S(\BU2/U0/gLUT.iLUT/N18 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [11])
);
```

```
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<11> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<11>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [13]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [11]),
  .O(\BU2/U0/gLUT.iLUT/N18 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<10>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [9]),
  .LI(\BU2/U0/gLUT.iLUT/N17 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [12])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<10>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [9]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [12]),
  .S(\BU2/U0/gLUT.iLUT/N17 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [10])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<10> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<10>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [12]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [10]),
  .O(\BU2/U0/gLUT.iLUT/N17 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<9>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [8]),
  .LI(\BU2/U0/gLUT.iLUT/N16 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [11])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<9>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [8]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [11]),
  .S(\BU2/U0/gLUT.iLUT/N16 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [9])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<9> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<9>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [11]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [9]),
  .O(\BU2/U0/gLUT.iLUT/N16 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<8>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [7]),
  .LI(\BU2/U0/gLUT.iLUT/N15 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [10])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<8>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [7]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [10]),
  .S(\BU2/U0/gLUT.iLUT/N15 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [8])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<8> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<8>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [10]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [8]),
  .O(\BU2/U0/gLUT.iLUT/N15 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<7>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [6]),
  .LI(\BU2/U0/gLUT.iLUT/N14 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [9])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<7>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [6]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [9]),
  .S(\BU2/U0/gLUT.iLUT/N14 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [7])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<7> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<7>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [9]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [7]),
  .O(\BU2/U0/gLUT.iLUT/N14 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<6>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [5]),
  .LI(\BU2/U0/gLUT.iLUT/N13 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [8])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<6>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [5]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [8]),
  .S(\BU2/U0/gLUT.iLUT/N13 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [6])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<6> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<6>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [8]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [6]),
  .O(\BU2/U0/gLUT.iLUT/N13 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<5>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [4]),
  .LI(\BU2/U0/gLUT.iLUT/N12 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [7])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<5>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [4]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [7]),
  .S(\BU2/U0/gLUT.iLUT/N12 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [5])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<5> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<5>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [7]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [5]),
  .O(\BU2/U0/gLUT.iLUT/N12 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<4>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [3]),
  .LI(\BU2/U0/gLUT.iLUT/N11 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [6])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<4>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [3]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [6]),
  .S(\BU2/U0/gLUT.iLUT/N11 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [4])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<4> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<4>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [6]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [4]),
  .O(\BU2/U0/gLUT.iLUT/N11 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<3>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [2]),
  .LI(\BU2/U0/gLUT.iLUT/N10 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [5])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<3>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [2]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [5]),
  .S(\BU2/U0/gLUT.iLUT/N10 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [3])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<3> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<3>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [5]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [3]),
  .O(\BU2/U0/gLUT.iLUT/N10 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<2>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [1]),
  .LI(\BU2/U0/gLUT.iLUT/N9 ),
  .O(\BU2/U0/gLUT.iLUT/s1_add_out<0> [4])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<2>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [1]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [4]),
  .S(\BU2/U0/gLUT.iLUT/N9 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [2])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<2> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<2>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [4]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [2]),
  .O(\BU2/U0/gLUT.iLUT/N9 )
);
XORCY \BU2/U0/gLUT.iLUT/Madd__add0000_xor<1>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [0]),
  .LI(\BU2/U0/gLUT.iLUT/N8 ),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [3])
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<1>  (
  .CI(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [0]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [3]),
  .S(\BU2/U0/gLUT.iLUT/N8 ),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [1])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<1> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<1>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [3]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [1]),
  .O(\BU2/U0/gLUT.iLUT/N8 )
);
MUXCY \BU2/U0/gLUT.iLUT/Madd__add0000_cy<0>  (
  .CI(\BU2/zero_detect [0]),
  .DI(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [2]),
  .S(\BU2/U0/gLUT.iLUT/s2_add_out<0> [2]),
  .O(\BU2/U0/gLUT.iLUT/Madd__add0000_cy [0])
);
defparam \BU2/U0/gLUT.iLUT/Madd__add0000_lut<0> .INIT = 4'h6;
LUT2 \BU2/U0/gLUT.iLUT/Madd__add0000_lut<0>  (
  .I0(\BU2/U0/gLUT.iLUT/pp_out_reg<0> [2]),
  .I1(\BU2/U0/gLUT.iLUT/pp_out_reg<1> [0]),
  .O(\BU2/U0/gLUT.iLUT/s2_add_out<0> [2])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_24 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_24  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [24]),
  .R(sclr),
  .C(clk),
  .Q(\BU2/U0/gLUT.iLUT/s5_add_out<0> [24])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_23 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_23  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [23]),
  .R(sclr),
  .C(clk),
  .Q(p_5[23])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_22 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_22  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [22]),
  .R(sclr),
  .C(clk),
  .Q(p_5[22])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_21 .INIT = 1'b0;
```

```
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_21  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [21]),
  .R(sclr),
  .C(clk),
  .Q(p_5[21])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_20 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_20  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [20]),
  .R(sclr),
  .C(clk),
  .Q(p_5[20])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_19 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_19  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [19]),
  .R(sclr),
  .C(clk),
  .Q(p_5[19])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_18 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_18  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [18]),
  .R(sclr),
  .C(clk),
  .Q(p_5[18])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_17 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_17  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [17]),
  .R(sclr),
  .C(clk),
  .Q(p_5[17])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_16 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_16  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [16]),
  .R(sclr),
  .C(clk),
  .Q(p_5[16])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_15 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_15  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [15]),
  .R(sclr),
  .C(clk),
  .Q(p_5[15])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_14 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_14  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [14]),
  .R(sclr),
  .C(clk),
  .Q(p_5[14])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_13 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_13  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [13]),
  .R(sclr),
  .C(clk),
  .Q(p_5[13])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_12 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_12  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [12]),
  .R(sclr),
  .C(clk),
  .Q(p_5[12])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_11 .INIT = 1'b0;
```

```
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_11  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [11]),
  .R(sclr),
  .C(clk),
  .Q(p_5[11])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_10 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_10  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [10]),
  .R(sclr),
  .C(clk),
  .Q(p_5[10])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_9 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_9  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [9]),
  .R(sclr),
  .C(clk),
  .Q(p_5[9])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_8 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_8  (
  .D(\BU2/U0/gLUT.iLUT/sum1<0> [8]),
  .R(sclr),
  .C(clk),
  .Q(p_5[8])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_7 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_7  (
  .D(\BU2/U0/gLUT.iLUT/s2_add_out<0> [7]),
  .R(sclr),
  .C(clk),
  .Q(p_5[7])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_6 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_6  (
  .D(\BU2/U0/gLUT.iLUT/s2_add_out<0> [6]),
  .R(sclr),
  .C(clk),
  .Q(p_5[6])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_5 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_5  (
  .D(\BU2/U0/gLUT.iLUT/s2_add_out<0> [5]),
  .R(sclr),
  .C(clk),
  .Q(p_5[5])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_4 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_4  (
  .D(\BU2/U0/gLUT.iLUT/s2_add_out<0> [4]),
  .R(sclr),
  .C(clk),
  .Q(p_5[4])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_3 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_3  (
  .D(\BU2/U0/gLUT.iLUT/s2_add_out<0> [3]),
  .R(sclr),
  .C(clk),
  .Q(p_5[3])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_2 .INIT = 1'b0;
FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_2  (
  .D(\BU2/U0/gLUT.iLUT/s2_add_out<0> [2]),
  .R(sclr),
  .C(clk),
  .Q(p_5[2])
);
defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_1 .INIT = 1'b0;
```

```
  FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_1  (
    .D(\BU2/U0/gLUT.iLUT/s2_add_out<0> [1]),
    .R(sclr),
    .C(clk),
    .Q(p_5[1])
  );
  defparam \BU2/U0/gLUT.iLUT/s3_add_out_0_0 .INIT = 1'b0;
  FDR \BU2/U0/gLUT.iLUT/s3_add_out_0_0  (
    .D(\BU2/U0/gLUT.iLUT/s2_add_out<0> [0]),
    .R(sclr),
    .C(clk),
    .Q(p_5[0])
  );
  VCC \BU2/XST_VCC  (
    .P(\BU2/N1 )
  );
  GND \BU2/XST_GND  (
    .G(\BU2/zero_detect [0])
  );

// synopsys translate_on

endmodule

// synopsys translate_off

`timescale  1 ps / 1 ps

module glbl ();

    parameter ROC_WIDTH = 100000;
    parameter TOC_WIDTH = 0;

    wire GSR;
    wire GTS;
    wire PRLD;

    reg GSR_int;
    reg GTS_int;
    reg PRLD_int;

//--------   JTAG Globals --------------
    wire JTAG_TDO_GLBL;
    wire JTAG_TCK_GLBL;
    wire JTAG_TDI_GLBL;
    wire JTAG_TMS_GLBL;
    wire JTAG_TRST_GLBL;

    reg JTAG_CAPTURE_GLBL;
    reg JTAG_RESET_GLBL;
    reg JTAG_SHIFT_GLBL;
    reg JTAG_UPDATE_GLBL;

    reg JTAG_SEL1_GLBL = 0;
    reg JTAG_SEL2_GLBL = 0 ;
    reg JTAG_SEL3_GLBL = 0;
    reg JTAG_SEL4_GLBL = 0;

    reg JTAG_USER_TDO1_GLBL = 1'bz;
    reg JTAG_USER_TDO2_GLBL = 1'bz;
    reg JTAG_USER_TDO3_GLBL = 1'bz;
    reg JTAG_USER_TDO4_GLBL = 1'bz;

    assign (weak1, weak0) GSR = GSR_int;
    assign (weak1, weak0) GTS = GTS_int;
    assign (weak1, weak0) PRLD = PRLD_int;

    initial begin
               GSR_int = 1'b1;
               PRLD_int = 1'b1;
		#(ROC_WIDTH)
		GSR_int = 1'b0;
		PRLD_int = 1'b0;
   end

   initial begin
		GTS_int = 1'b1;
		#(TOC_WIDTH)
		GTS_int = 1'b0;
   end

endmodule

// synopsys translate_on

// Squaring Circuit 1
`timescale 1ns / 1ps
module squaring_ckt_1(
input			clk,
			clr,
input			[11:0]	fir2_out,
input			[1:0]	sel_o_2,
input			rdy_fir_da_2,
output	reg		rdy_sqr_ckt,
output	reg		[23:0]	square_out);
//intermediate signals
reg			[11:0]	fir2_out_reg;
reg			[1:0]	sel_o_2_reg_a,
				sel_o_2_reg_b;
reg				rdy_sqr_ckt_a,
				rdy_sqr_ckt_b,
				rdy_sqr_ckt_c;
wire			[23:0]	mul_out;
reg			[23:0]	add_a_in,
				add_b_in;
wire			[23:0]	add_out;

always @ (posedge clk)
begin
	if (clr)
	begin
		fir2_out_reg <= 0;
		sel_o_2_reg_a <= 0;
		rdy_sqr_ckt_a <= 0;
	end
	else if ((rdy_fir_da_2==1) && ((sel_o_2==2'd2)||(sel_o_2==2'd3)))
	begin
		fir2_out_reg  <= fir2_out;
		sel_o_2_reg_a <= sel_o_2;
		if (sel_o_2==2'd3)
			rdy_sqr_ckt_a <= 1;
	end
	else
		rdy_sqr_ckt_a <= 0;
end

always @ (posedge clk)
begin
	if (clr)
	begin
		rdy_sqr_ckt_b <= 0;
		sel_o_2_reg_b <= 0;
	end
	else
	begin
		rdy_sqr_ckt_b <= rdy_sqr_ckt_a;
		sel_o_2_reg_b <= sel_o_2_reg_a;
	end

end
```

```
mul_sq MUL_SQ (
                    .sclr(clr),
                    .clk(clk),
                    .a(fir2_out_reg),
                    .b(fir2_out_reg),
                    .p(mul_out)
);

always @ (posedge clk)
begin
          if (clr) begin
                    add_a_in <= 0;
                    add_b_in <= 0;
                    rdy_sqr_ckt_c <= 0;
          end
          else
          begin
                    case (sel_o_2_reg_b)
                              2'd2:add_a_in <= mul_out;
                              2'd3:add_b_in <= mul_out;
                    endcase
                    rdy_sqr_ckt_c <= rdy_sqr_ckt_b;
          end
end

assign add_out = add_a_in + add_b_in;

always @ (posedge clk)
begin
          if (clr)
          begin
                    square_out <= 0;
                    rdy_sqr_ckt <= 0;
          end
          else
          begin
                    square_out <= add_out;
                    rdy_sqr_ckt <= rdy_sqr_ckt_c;
          end
end
endmodule

// Demultiplexer
`timescale 1ns / 1ps
module clk2_clk3_bridge(rdy_sqr_ckt_0,rdy_sqr_ckt_1,square_0,square_1,
                                        square_out);
parameter n=24;
input     rdy_sqr_ckt_0,
          rdy_sqr_ckt_1;
input     [n-1:0]   square_0,
                    square_1;
output reg [n-1:0]  square_out;

always@(rdy_sqr_ckt_0,rdy_sqr_ckt_1,square_0,square_1)
begin
          if (rdy_sqr_ckt_0)
                    square_out <= square_0;
          else if (rdy_sqr_ckt_1)
                    square_out <= square_1;
          else
                    square_out <= 0;
end
endmodule

// Square Root Circuit
`timescale 1ns / 1ps
//`default_nettype none
module square_root(clk,clr,remainder_in,sum_in,sum_out,
                                        root_out);
```

```
parameter n=24;
parameter n1=3;
parameter depth=n/2;
input                   clk,clr;
input                   [n-1:0]      remainder_in;
input                   [n1-1:0] sum_in;
output      [n-1:0]     root_out;
output      [n1-1:0]    sum_out;
wire                    [n-1:0]      remainder_t_0,// to connect reg and comb logic
                                                          mask_t_0,
                                                          root_t_0;

wire                    [n-1:0]      remainder_o          [0:depth-1];
wire                    [n-1:0]      mask_o      [0:depth-1];
wire                    [n-1:0]      root_o      [0:depth-1];
wire                    [n1-1:0]     sum_o                [0:depth-1];
reg                     [n-1:0] root_o_end;


square_reg_0 #(.n(n),.n1(n1)) SQ_REG_0 (
                                        .clk(clk),
                                        .clr(clr),
                                        .remainder_in(remainder_in),
                                        .remainder_out(remainder_t_0),
                                        .sum_in(sum_in),
                                        .sum_out(sum_o[0]),
                                        .mask_out(mask_t_0),
                                        .root_out(root_t_0));

square_root_bit #(.n(n)) SQ_BIT_0 (
                                        .root_i(root_t_0),
                                        .mask_i(mask_t_0),
                                        .remainder_i(remainder_t_0),
                                        .root_o(root_o[0]),
                                        .mask_o(mask_o[0]),
                                        .remainder_o(remainder_o[0]));

genvar i;
generate
            for (i=1;i<=(depth-1);i=i+1)
            begin:SQ
                        wire [n-1:0]            root_t,
                                                mask_t,
                                                remainder_t;

                        square_reg #(.n(n),.n1(n1)) SQ_REG (
                                                .clk(clk),
                                                .clr(clr),
                                                .remainder_in(remainder_o[i-1]),
                                                .remainder_out(remainder_t),
                                                .sum_in(sum_o[i-1]),
                                                .sum_out(sum_o[i]),
                                                .mask_in(mask_o[i-1]),
                                                .mask_out(mask_t),
                                                .root_in(root_o[i-1]),
                                                .root_out(root_t));

                        square_root_bit #(.n(n)) SQ_BIT (
                                                .root_i(root_t),
                                                .mask_i(mask_t),
                                                .remainder_i(remainder_t),
                                                .root_o(root_o[i]),
                                                .mask_o(mask_o[i]),
                                                .remainder_o(remainder_o[i]));
            end
endgenerate

always @ (root_o[depth-1],remainder_o[depth-1])
begin
            if (remainder_o[depth-1] > root_o[depth-1])
```

```
                root_o_end = root_o[depth-1]+1;
        else
                root_o_end = root_o[depth-1];
end

square_reg_end #(.n(n),.n1(n1)) SQ_REG_END (
                                .clk(clk),
                                .clr(clr),
                                .sum_in(sum_o[depth-1]),
                                .sum_out(sum_out),
                                .root_in(root_o_end),
                                .root_out(root_out));


Endmodule

// Square Reg 0
`timescale 1ns / 1ps
module square_reg_0(clk,clr,remainder_in,remainder_out,sum_in,sum_out,
                                mask_out,root_out);
parameter n=24;
parameter n1=3;
input                   clk,clr;
input                   [n-1:0]     remainder_in;
input                   [n1-1:0] sum_in;
output      reg         [n-1:0]     remainder_out,
                                    mask_out,
                                    root_out;
output      reg         [n1-1:0]    sum_out;

always @(posedge(clk))
begin
                mask_out = 24'd4194304;
                root_out    = 0;
end

always @(posedge clk)
begin
        if (clr)
        begin
                remainder_out <= 0;
                sum_out    <=          0;
        end
        else
        begin
                remainder_out <= remainder_in;
                sum_out    <=          sum_in;
        end
end
endmodule

// Square Root Bit
`timescale 1ns / 1ps
module square_root_bit(root_i,mask_i,remainder_i,root_o,mask_o,remainder_o);
parameter n=24;
input                   [n-1:0]     root_i,
                                    mask_i,
                                    remainder_i;
output reg              [n-1:0]     root_o,
                                    mask_o,
                                    remainder_o;

always@(root_i,mask_i,remainder_i)
begin
        if((root_i+mask_i) <= remainder_i)
        begin
                remainder_o = (remainder_i -(root_i + mask_i));
                root_o = root_i + mask_i + mask_i;
        end
        else
        begin
                remainder_o = remainder_i;
                root_o = root_i;
        end

        root_o[n-2:0]=root_o[n-1:1]; //right shift by 1
        root_o[n-1]=0;                                              //

        mask_o[n-3:0]=mask_i[n-1:2]; // right shift by 2
        mask_o[n-1:n-2]=0;

end
endmodule

// Square Reg
`timescale 1ns / 1ps
module square_reg(clk,clr,remainder_in,remainder_out,sum_in,sum_out,
                        mask_in,mask_out,root_in,root_out);
parameter n=24;
parameter n1=3;
input           clk,clr;
input           [n-1:0]     remainder_in,
                            mask_in,
                            root_in;
input           [n1-1:0]     sum_in;
output  reg     [n-1:0]     remainder_out,
                            mask_out,
                            root_out;
output  reg     [n1-1:0]    sum_out;

always @(posedge clk)
begin
        if (clr)
        begin
                remainder_out <= 0;
                mask_out <= 0;
                root_out <= 0;
                sum_out     <=      0;
        end
        else
        begin
                remainder_out <= remainder_in;
                mask_out <= mask_in;
                root_out <= root_in;
                sum_out     <=      sum_in;
        end
end
endmodule

// Square Reg End
`timescale 1ns / 1ps
module square_reg_end(clk,clr,sum_in,sum_out,root_in,root_out);
parameter n=24;
parameter n1=3;

input           clk,clr;
input           [n-1:0]     root_in;

input           [n1-1:0] sum_in;
output  reg     [n-1:0]     root_out;
output  reg     [n1-1:0]    sum_out;

always @(posedge clk)
begin
        if (clr)
        begin
                root_out <= 0;
                sum_out     <=      0;
        end
        else
```

```
		begin
				root_out <= root_in;
				sum_out	<=		sum_in;
		end
end
endmodule

// Mean Remover
`timescale 1ns / 1ps
module mean_remover1(clk,clr,rdy_root,in,out);

parameter  n=12;

input					clk,
					clr;
input					rdy_root;
input					[n-1:0]	in;
output					[n+1:0]	out;
reg					[n-1:0]	k=25;
reg					[n-1:0]	r1,
							r2;
reg					[(2*n)-1:0]r3;
reg					rdy_root_a,
					rdy_root_b,
					rdy_root_c;
wire					[n-1:0]	diff1;
wire					[(2*n)-1:0]mult,
							add1,
							fback2;


always @ (posedge (clk))
begin
		if (clr)
				r1		<= 0;
		else if (rdy_root)
				r1 <= in;
end

always @ (posedge (clk))
begin
		if (clr)
				rdy_root_a <= 0;
		else
				rdy_root_a <= rdy_root;
end

assign diff1 = r1 - fback2[23:12];

always @ (posedge(clk))
begin
		if (clr)
				r2 <= 0;
		else if (rdy_root_a)
				r2 <= diff1;
end

always @ (posedge (clk))
begin
		if (clr)
				rdy_root_b <= 0;
		else
				rdy_root_b <= rdy_root_a;
end

mul_sq MUL_MEAN (
				.sclr(clr),
				.clk(clk),
				.a(r2),
				.b(k),
                .p(mult)
);

always @ (posedge (clk))
begin
        if (clr)
                rdy_root_c <= 0;
        else
                rdy_root_c <= rdy_root_b;
end

assign add1 = fback2 + mult;

always @ (posedge (clk))
begin
        if (clr)
                r3 <= 0;
        else if (rdy_root_c)
                r3 <= add1;
end

assign fback2 = r3;
//assign out1=fback2;

assign out = {(~r2[11]),r2[11],r2};

endmodule
```

Appendix B

Matlab files and Verilog Testbench Code

1. Matlab File for Generating Input Samples to the Design

```
clear;
clc;

fs=25*10^6; %sampling frequency

fc=590*10^3;  %carrier1(or station 1) frequency
fc1=600*10^3; %carrier2(or station 2) frequency

t =[0:1/fs:0.01]; %sample intervals

m= 0.5*sin(2*pi*2000*t)+0.5*sin(2*pi*1000*t); %modulating signal(for station 1)

m1=0.5*sin(2*pi*3000*t);%modulating signal(for station 2)
am= ammod(m,fc,fs,5,1) + ammod(m1,fc1,fs,5,1); %amplitude modulated signal

am_shrink= am*0.30;

am_adc = round(am_shrink./0.0005); % This can be thought of as the amplitude modulated
                                   % signal after the a/d converter


max_value=length(am_adc);

FILE = fopen('samples_from_matlab.txt', 'w');
  if (FILE == -1)
    error('cannot open file for writing');
  end
for i=1:max_value
   samples=twoscomp(am_adc(i),12);
   fprintf(FILE,'%s\n',samples);
end
fclose(FILE);
```

2. Verilog Testbench Code

```
`timescale 1ns / 1ps
module testbench_single_clk_domain();


reg         clk,
            reset_n;
reg         [13:0]      adc_in;
reg         [2:0]                   rom_addr_0,
                                    rom_addr_1;
wire        [13:0]      dac_0,
                        dac_1;


////////////////////////

parameter  no_of_samples = 250001;//modify
parameter  adc_resolution = 12;
```

```
reg         [adc_resolution-1:0]   sample_mem          [0:no_of_samples-1];
reg         [adc_resolution-1:0] output_samples;
/////////////////////////
integer     fd_out_0,fd_out_1;
integer     i=0,j=0;
//reg       [(2*adc_resolution-1):0]          temp_output;
reg         k=0,l=0;

initial begin
            $readmemb("samples_from_matlab.txt",sample_mem);
            fd_out_0 = $fopen("samples_to_matlab_0.txt", "w");
            fd_out_1 = $fopen("samples_to_matlab_1.txt","w");
end
initial begin
            clk = 0;
            forever begin
                        #40 clk = ~clk;
            end
end
always@(posedge clk)
begin
             output_samples = sample_mem[i];
             #2 adc_in = {output_samples[11],output_samples[11],output_samples};
             i = i + 1;
end
always@(posedge clk)
begin
            $fdisplay(fd_out_0,"%b",{(~dac_0[13]),dac_0[12:0]});
            $fdisplay(fd_out_1,"%b",{(~dac_1[13]),dac_1[12:0]});
            j=j+1;
            if (j==250001)//change
            begin
                        $fclose(fd_out_0);
                        $fclose(fd_out_1);
                        $finish;
            end
end

initial begin
                        rom_addr_0 = 0;
                        rom_addr_1 = 1;
                        reset_n = 1;
                        @ (posedge clk)
                                    #2 reset_n = 0;
                        @ (posedge clk)
                                    #2 reset_n = 1;
```

```
end

          top TOP (
                          .clk(clk),
                          .reset_n(reset_n),
                          .adc_in(adc_in),
                          .rom_addr_0(rom_addr_0),
                          .rom_addr_1(rom_addr_1),
                          .dac_0(dac_0),
                          .dac_1(dac_1)

           );
Endmodule
```

References

[1] Simon Haykin, Communication Systems, Second Edition, John Wiley & Sons, 1983.

[2] Herbert Taub & Donald L. Schilling, Principles of Communication Systems, Tata McGraw-Hill, 1991.

[3] http://en.wikipedia.org/wiki/Superheterodyne_receiver.

[4] United States Frequency Allocations, www.ntia.doc.gov/osmhome/allochrt.pdf.

[5] Jeffrey H. Reed, Software Radio: A Modern Approach to Radio Engineering, Prentice Hall PTR, 2002.

[6] Analog Devices, http://www.analog.com/library/analogdialogue/archives/38-08/dds.pdf.

[7] Analog Devices, AD6645, http://www.analog.com/UploadedFiles/Data_Sheets/AD6645.pdf.

[8] Xilinx, Sine/Cosine Look-Up Table, www.**xilinx**.com/ipcenter/catalog/logicore/docs/sincos.pdf.

[9] Xilinx, DDS v5.0, www.**xilinx**.com/ipcenter/catalog/logicore/docs/**dds**.pdf.

[10] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Magazine*, vol. 6, pp. 4–19, July 1989.

[11] T.-S. Chang and C.-W. Jen, "Hardware-efficient implementations for discrete function transforms using LUT-based FPGAs," *IEE Proceedings Circuits, Devices and Systems*, vol.146, no. 6, pp. 309–315, Nov. 1999.

[12] Xilinx, Application Note, www.xilinx.com/appnotes/theory1.pdf.

[13] Xilinx, Distributed Arithmetic FIR Filter v9.0, www.xilinx.com/ipcenter/catalog/logicore/docs/da_fir.pdf.

[14] M. T. Tommiska, "Area-efficient implementation of a fast square root algorithm," *Proceedings of the 2000 Third IEEE International Caracas Conference on Devices, Circuits and Systems*, pp. S18/1 -S18/4, 2000.

[15] Analog Devices, AD9772A, www.analog.com/UploadedFiles/Data_Sheets/AD9772A.pdf.

[16] J. Mitola, "The software radio architecture," IEEE Communications Magazine, vol.33, No. 5, pp. 26–38, May 1995.

[17] Jung Ko, Vincent C. Gaudet, Robert Hang, "A Tier 3 Software Defined AM Radio," *iwsoc, Fifth International Workshop on System-on-Chip for Real-Time Applications (IWSOC'05)*, pp. 257-261, 2005.

[18] Glossner, J., Iancu, D., Jin Lu, Hokenek, E., Moudgill, M., "A software-defined communications baseband design," *Communications Magazine, IEEE*, vol.41, no.1, pp. 120-128, Jan 2003.

[19] L. Zhigang, L. Wei, Z. Yan, and G. Wei, "A multistandard SDR base band platform," *IEEE International Conference on Computer Networks and Mobile Computing*, pp. 461–464, 2003.

[20] H. Yoshida, H. Tsurumi, and Y. Suzuki., "Broadband RF front-end and software execution procedure in software defined radio," *IEEE Transaction on Vehicular Technology Conference*, vol. 4, pp 2133-2137, 1999.

[21] Nakagawa, Y., Muraguchi, M., Kawamura, H., Ohashi, K., Sakaguchi, K., Araki, K., "Novel Multi-Stage Transmultiplexing Digital Down Converter for Implementation of RFID (ISO18000-3 MODE 2) Reader/Writer," *Vehicular Technology Conference*, 2007. VTC2007-Spring, IEEE 65th, pp.2300-2304, 2007

[22]Nallatech Xtreme DSP Development Kit Pro,
http://www.xilinx.com/ipcenter/dsp/XtremeDSP_Development_Kit_Pro_User_Guide.pdf

[23] Xilinx Virtex 2 Pro FPGA User Guide,
http://www.xilinx.com/support/documentation/user_guides/ug012.pdf.

[24] http://www.xilinx.com/publications/xcellonline/xcell_49/xc_techxclusives49.htm

[25] Rivet, F.; Deval, Y.; Begueret, J.B.; Dallet, D.; Belot, D., "A Disruptive Software-Defined Radio Receiver Architecture Based on Sampled Analog Signal Processing," *Radio Frequency Integrated Circuits (RFIC) Symposium, 2007 IEEE* , vol., no., pp.197-200, 3-5 June 2007.

[26] Mahes, R.; Vinod, A.P., "An Architecture For Integrating Low Complexity and Reconfigurability for Channel filters in Software Defined Radio Receivers," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on* , vol., no., pp.2514-2517, 27-30 May 2007.

[27] Girau, G.; Tomatis, A.; Dovis, F.; Mulassano, P., "Efficient Software Defined Radio Implementations of GNSS Receivers," *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on* , vol., no., pp.1733-1736, 27-30 May 2007.

[28] Abidi, A. A., "The Path to the Software-Defined Radio Receiver," *Solid-State Circuits, IEEE Journal of* , vol.42, no.5, pp.954-966, May 2007.

[29] Reinhart, R.C.; Scardelletti, M.C.; Mortensen, D.J.; Kacpura, T.; Andro, M.; Smith, C.; Liebetreu, J.; Farrington, A., "Hardware Architecture Study for NASA's Space Software Defined Radios," *Wireless and Microwave Technology Conference, 2006. WAMICON '06. IEEE Annual* , vol., no., pp.1-5, 4-5 Dec. 2006.

[30] Yokoyama, Akihisa; Matsumura, Takashi; Harada, Hiroshi, "Implementation of Multi-channel Modern and Multi-channel Radio Broadcast Receiver on Reconfigurable Packet Routing-oriented Signal Processing Platform (RPPP)," *Networking Technologies for Software Defined Radio Networks, 2006. SDR '06.1st IEEE Workshop on* , vol., no., pp.59-67, 25-25 Sept. 2006.

[31] Kruth, A.; Simon, M.; Dufrene, K.; Weigel, R.; Boos, Z.; Heinen, S., "A Multimode Receiver Front-end for Software Defined Radio," *Wireless Technology, 2006. The 9th European Conference on* , vol., no., pp.19-22, 10-12 Sept. 2006.

[32] Bagheri, R.; Mirzaei, A.; Heidari, M.E.; Chehrazi, S.; Minjae Lee; Mikhemar, M.; Tang, W.K.; Abidi, A.A., "Software-defined radio receiver: dream to reality," *Communications Magazine, IEEE* , vol.44, no.8, pp. 111-118, Aug. 2006.

[33] Spacek, J.; Puricer, P., "Front-end Module for GNSS Software Receiver," *Multimedia Signal Processing and Communications, 48th International Symposium ELMAR-2006 focused on* , vol., no., pp.211-214, June 2006.

[34] Yanyang Zhao; Ligen Wang; Jean-Francois Frigon; Chahe Nerguizian; Ke Wu; Renato G. Bosisio, "A Software Defined Radio Receiver Architecture for UWB Communications and Positioning," *Electrical and Computer Engineering, 2006. CCECE '06. Canadian Conference on* , vol., no., pp.255-258, May 2006.

[35] Yuan Lin; Hyunseok Lee; M. Woh; Y. Harel; S. Mahlke; T. Mudge; C. Chakrabarti; K. Flautner, "SODA: A Low-power Architecture For Software Radio," *Computer Architecture, 2006. ISCA '06. 33rd International Symposium on* , vol., no., pp.89-101, 2006.

[36] Di Stefano, A.; Fiscelli, G.; Giaconia, C.G., "An FPGA-Based Software Defined Radio Platform for the 2.4GHz ISM Band," *Research in Microelectronics and Electronics 2006, Ph. D.* , vol., no., pp.73-76, 2006.

[37] Xinyu Xu; Bosisio, R.G.; Ke Wu, "Analysis and implementation of software defined radio receiver platform," *Microwave Conference Proceedings, 2005. APMC 2005. Asia-Pacific Conference Proceedings* , vol.5, no., pp. 4 pp.-, 4-7 Dec. 2005.

[38] Harada, H., "Software defined radio prototype toward cognitive radio communication systems," *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on* , vol., no., pp.539-547, 8-11 Nov. 2005.

[39] Isomaki, P.; Avessta, N., "Rapid Refinable SoC SDR Design," *System-on-Chip, 2005. Proceedings. 2005 International Symposium on* , vol., no., pp. 120-123, 15-17 Nov. 2005.

[40] van Heijster, R.M.E.M., "Universal precision ESM receiver based on software defined radio technology," *Wireless Technology, 2005. The European Conference on* , vol., no., pp. 419-422, 3-4 Oct. 2005.

[41] Chamberlain, M.W., "A software defined HF radio," *Military Communications Conference, 2005. MILCOM 2005. IEEE* , vol., no., pp. 2448-2453 Vol. 4, 17-20 Oct. 2005.

[42] Isomaki, P.; Avessta, N., "Rapid Refinable Software Defined Radio Design," *Wireless Communication Systems, 2005. 2nd International Symposium on* , vol., no., pp. 575-578, 5-7 Sept. 2005.

[43] Blaickner, A.; Albl, S.; Scherr, W., "Configurable computing architectures for wireless and software defined radio - a FPGA prototyping experience using high level design-tool-chains," *System-on-Chip, 2004. Proceedings. 2004 International Symposium on* , vol., no., pp. 111-116, 16-18 Nov. 2004.

[44] Sala, M.; Salidu, F.; Stefani, F.; Kutschenreiter, C.; Baschirotto, A., "Design considerations and implementation of a DSP-based car-radio IF Processor," *Solid-State Circuits, IEEE Journal of* , vol.39, no.7, pp. 1110-1118, July 2004.

[45] Jaamour, Y.; Safadi, M.S., "A study of software radio evaluation and experimentation platforms," *Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on* , vol., no., pp. 205-206, 19-23 April 2004.

[46] Luy, J.-F.; Mueller, T.; Mack, T.; Terzis, A., "Configurable RF receiver architectures," *Microwave Magazine, IEEE* , vol.5, no.1, pp. 75-82, Mar 2004.

[47] Faust, O.; Sputh, B.; Nathan, D.; Rezgui, S.; Weisensee, A.; Allen, A., "A single-chip supervised partial self-reconfigurable architecture for software defined radio," *Parallel and Distributed Processing Symposium, 2003. Proceedings. International* , vol., no., pp. 7 pp.-, 22-26 April 2003.

[48] Ren Guanghui; Zhao Yaqin; Wu Zhilu; Gu Xuemai, "Channelized receiver platform of SDR based on FPGAs," *ASIC, 2003. Proceedings. 5th International Conference on* , vol.2, no., pp. 840-843 Vol.2, 21-24 Oct. 2003.

[49] Fujimaki, A.; Nakazono, K.; Hasegawa, H.; Sato, T.; Akahori, A.; Takeuchi, N.; Furuta, F.; Katayama, M.; Hayakawa, H., "Broad band software-defined radio receivers based on superconductive devices," *Applied Superconductivity, IEEE Transactions on* , vol.11, no.1, pp.318-321, Mar 2001.

[50] Haruyama, S.; Morelos-Zaragoza, R., "A software defined radio platform with direct conversion: SOPRANO ," *Vehicular Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th* , vol.3, no., pp.1558-1560 vol.3, 2001.

[51] Fox, P.W., "Software defined radios-Motorola's Wireless Information Transfer System (WITS)," *EUROCOMM 2000. Information Systems for Enhanced Public Safety and Security. IEEE/AFCEA* , vol., no., pp.43-46, 2000.

[52] Efstathiou, D.; Fridman, L.; Zvonar, Z., "Recent developments in enabling technologies for software defined radio," *Communications Magazine, IEEE* , vol.37, no.8, pp.112-117, Aug 1999.

[53] Cook, P.G.; Bonser, W., "Architectural overview of the SPEAKeasy system," *Selected Areas in Communications, IEEE Journal on* , vol.17, no.4, pp.650-661, Apr 1999.

Vita

Veerendra Bhargav Alluri was born on 20th May 1983 in Kakinada, India. He obtained his Bachelor of Science in Electronics and Communications Engineering in May of 2005 from Andhra University, India. He enrolled in the University of Kentucky's Graduate School in the Fall Semester of 2005.