SANDIA REPORT
SAND2001-1827
Unlimited Release
Printed June 2001

Synthetic Aperture Radar Image Formation in Reconfigurable Logic

Peter A. Dudley

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of
Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

SAND2001-1827
Unlimited Release
Printed June 2001

SYNTHETIC APERTURE RADAR IMAGE FORMATION IN RECONFIGURABLE LOGIC

Peter A. Dudley
Special Processors & Computers
Sandia National Laboratories
PO Box 5800
Albuquerque, NM 87109-0519

ABSTRACT

This paper studies the implementation of polar format, synthetic aperture radar image formation in modern Field Programmable Gate Arrays (FPGA's).

The polar format algorithm is described in rough terms and each of the processing steps is mapped to FPGA logic. This FPGA logic is analyzed with respect to throughput and circuit size for compatibility with airborne image formation.

This page left intentionally blank.

LIST OF FIGURES

1 Summary

This paper studies the implementation of polar format, synthetic aperture radar image formation in modern Field Programmable Gate Arrays (FPGA's).

The polar format algorithm is described in rough terms and each of the processing steps is mapped to FPGA logic. This FPGA logic is analyzed with respect to throughput and circuit size for compatibility with airborne image formation.

2 The Modern FPGA

Modern FPGA's are RAM based programmable logic chips that offer performance and size comparable to application specific integrated circuits but without the high cost of custom masks and fabrication runs. FPGA's are true standard off-the-shelf parts that are programmed at power-up time. In this paper we will use the Xilinx Virtex II FPGA family for our analysis.

2.1 Logic Cell

The basic building block of FPGA logic is the logic cell or LC and both major FPGA vendors employ the same basic structure illustrated in Figure 1. The LC consists of a 16 by 1 Look Up Table (LUT) followed by an optional register. The LUT can be loaded at power up time to implement any combinational function of four inputs. Also the LUT can be used as a RAM to provide single or dual port read/write storage or the RAM can be configured as FIFO memory. Programmable delay lines can also be built up from LUT RAM's.

Not shown are special carry/cascade paths that combine LC's to efficiently implement arithmetic functions like adders and high fan-in combinational functions like multiplexors. Addition operations like adders, accumulators, subtractors or counters require 1 + N LC's where N is the width of the function. For example a 24-bit accumulator requires 25 LC's. Multiplexors require one LC per pair of input bits so a single 64 to 1 multiplexor needs 32 LC's.

The largest devices contain up to 122,880 LC's. Programmable routing resources allow the LC's to be interconnected. In this document we will count a LC as consumed if the LUT or the flip-flop or both are used.

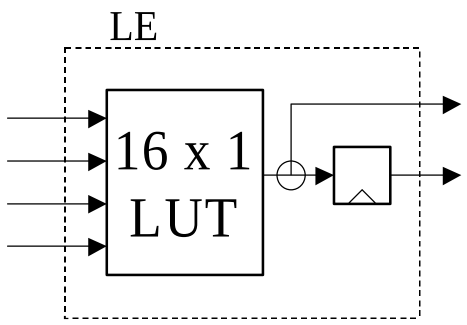


Figure 1: FPGA Logic Cell

2.2 Block Memory

To make use of the large number of transistors available today FPGA vendors are adding blocks of RAM to their arrays. The Xilinx Virtex II chips have up to 192 18K-bit blocks per chip. This memory can be programmed to operate in an amazing variety of configurations. These memories can be RAM or ROM, dual ported or single ported, synchronous or asynchronous, 512 by 36 or 1K by 18 or 2K by 9 or 4K by 4 or 8K by 2 or 16K by 1. Furthermore the two ports can be different widths so that you can access the memory

18 bits wide on port A and 36 bits wide on port B and so on. The blocks can function as First In First Out (FIFO) buffers or Content Addressable Memories (CAM's).

Memories larger than 18K bits can be created by combining multiple block RAM's. For example, a 4K deep by 25 wide memory can be implemented using 6 block RAM's (four configured as 2K by 8 and two configured as 2K by 9).

For this application we will only be using the block RAM's as FIFO's, synchronous dual port RAM's and ROM's.

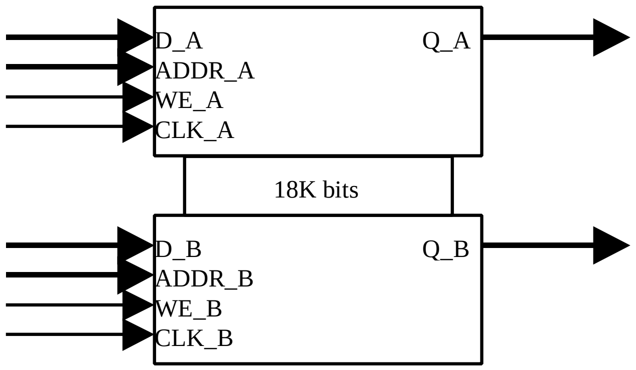


Figure 2: FPGA Block Memory

2.3 *Embedded Multipliers*

For each block memory the Virtex II FPGA's provide an embedded 18 by 18 bit multiplier. They can do 18 by 18 two's complement multiply, 17 by 17 bit unsigned multiply and under some circumstances they can be configured to perform two smaller multiplies such as 5 by5 and 6 by 6 in the same block.

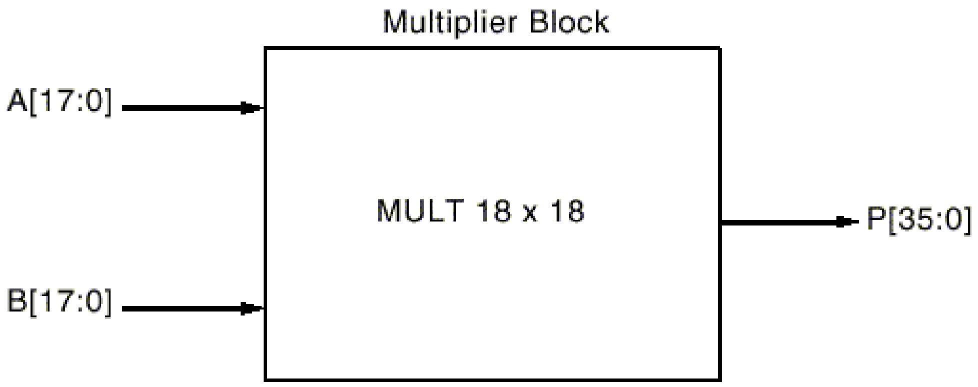


Figure 3: FPGA Embedded Multiplier

3 Polar Format Image Formation Algorithm

While the mathematics of synthetic aperture image formation are beyond the scope of this paper two basic algorithms have been proposed here for polar format[i,ii] processing in FPGA's. The first requires a unique sinc interpolator before a 2D FFT. The second eliminates the sinc interpolator and replaces the first FFT operation with a chirp-Z transform. In either case, a digital receiver producing complex range vectors at about a 1KHz rate precedes the image formation hardware. For this paper, 4096 by 4096 image size will be assumed and the results can be scaled for other sizes.

It should be mentioned here that other SAR image formation algorithms exist. In particular, the Overlap Sub-Aperture (OSA) algorithm may offer advantages in terms of memory size and processing latency. Also high resolution SAR images generally require auto-focus processing not discussed in this paper and OSA processing has additional advantages when auto-focus is taken into account.

Polar format processing was selected for this paper because of its relative ease of presentation. Finally, the image formation algorithm presented here should be considered a skeleton outline for logic estimation purposes. Important details such as auto-focus are missing.

3.1 Polar Format Image Formation with sinc interpolation

A bare bones block diagram of one polar format image formation algorithm is shown in Figure 4. A digital receiver produces N length complex range vectors at about a 1KHz rate. Polar format image formation is inherently an azimuth before range algorithm so an entire N by N image buffer of phase histories must be stored before image formation on the data can begin. The corner turn memory is an N by N image buffer that inputs data in row major order and outputs data in column major order. The effect is that data goes into the corner turn memory in range order and comes out in azimuth order ready for azimuth sinc interpolation.

Sinc interpolation is a resampling process that changes the spacing of samples in the azimuth direction. Unfortunately sinc interpolation is an unusual signal processing operation that does not lend itself to systolic processing in fixed hardware. Basically the sinc interpolator is a FIR filter of approximately 17 taps where the filter coefficients vary in the range and azimuth directions. This means that 17 new filter coefficients must be loaded for each new complex sample into the interpolator. The filter coefficients could be pre-computed and stored in a memory array but if the image size is 4096 by 4096 and the coefficients are 12 bits we would need about a .5 G byte memory organized 204 bits wide by 16 Meg deep. While not impossible to implement in hardware let's file it under "very difficult" for now and hope that the chirp-z algorithm saves us.

After the sinc interpolation step, two N length FFT's separated by a corner turn operation are performed. FFT implementation is discussed in section 4. Finally the complex image is converted to magnitude display.

Please note that pipelined processing of data is implied here. That is, several successive images are being calculated simultaneously. For example if image n is being calculated in the range FFT then image n+1 is being calculated in the azimuth FFT and the digital receiver will be generating image n+2.

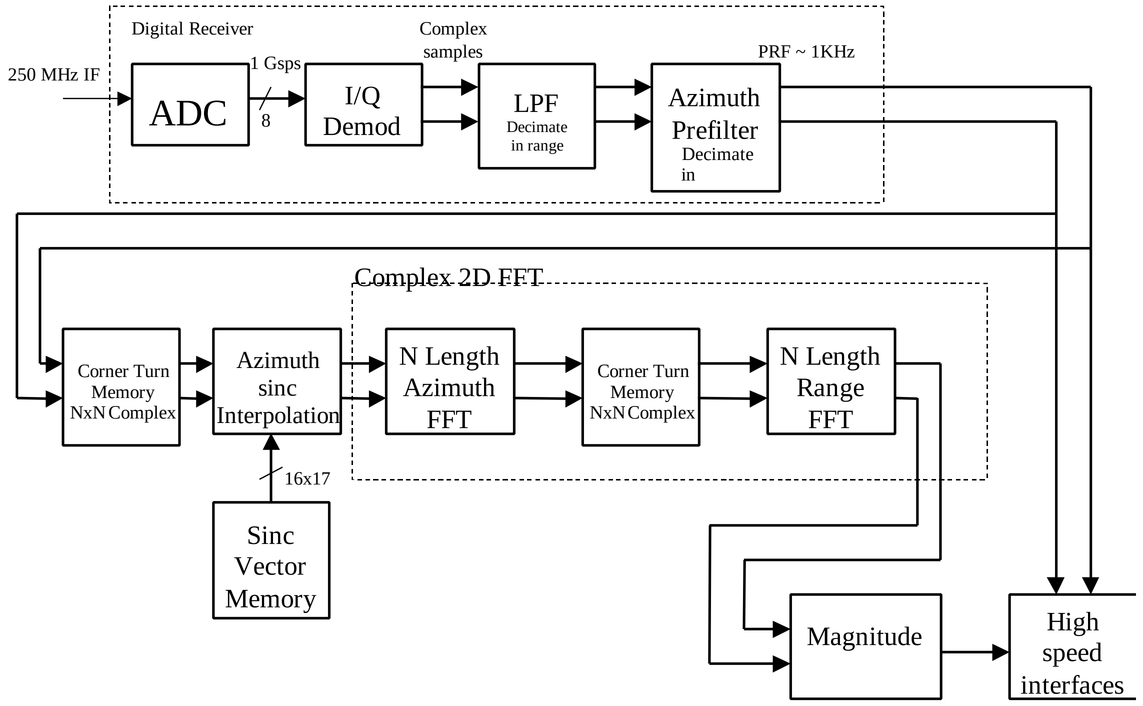


Figure 4: Polar Format Processing with Sinc Interpolator

Observations:

- Digital receiver produces N length complex range vectors at a pulse repetition frequency (PRF) of about 1 kHz.
- For N = 4096, 1KHz * 4096 = 4 Meg complex samples per second.
- The image formation engine is pipelined so each stage is only required to process 4 M samples/second.
- Sinc interpolation is "very difficult" in systolic hardware.

3.2 Polar Format Image Formation with chirp-Z transform

The chirp-Z image formation engine of Figure 6 replaces the sinc interpolator and N length azimuth FFT with a chirp-Z transform. For our purposes the chirp-Z transform in Figure 6 is itself built from three quadrature Direct Digital Synthesis (DDS) complex sinusoid generators, three complex multipliers and two 2N-length FFT blocks. This seems good because we are replacing the sinc interpolator that we do not know how to build with blocks that we do know how to build.

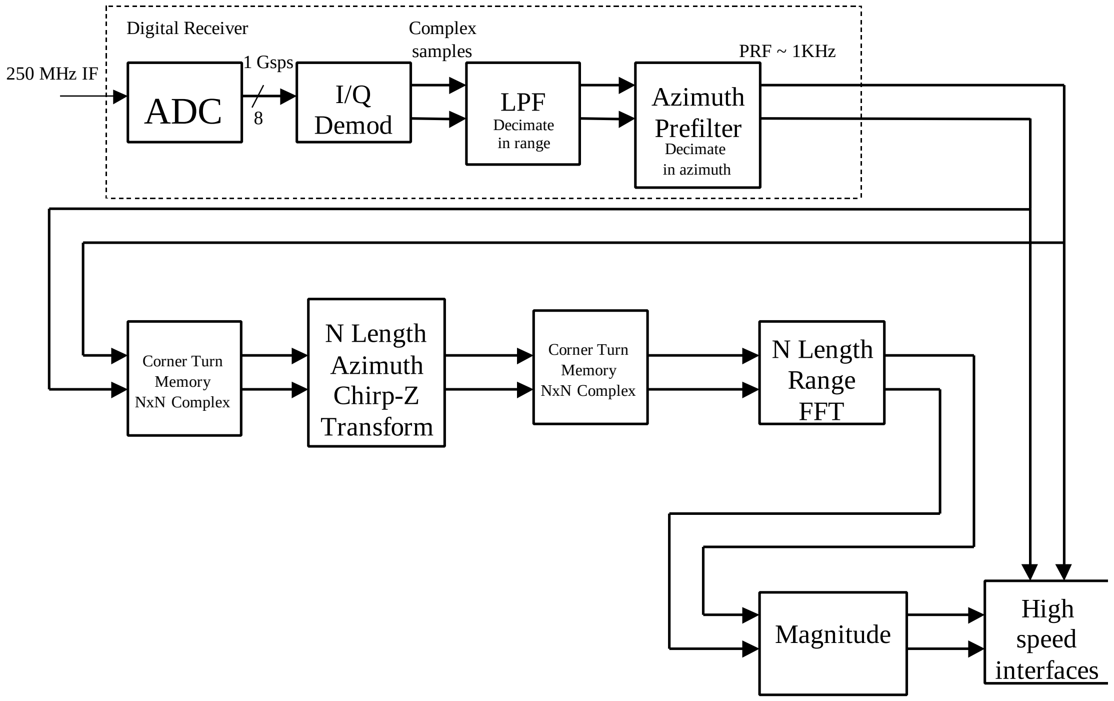


Figure 5: Polar Format Processing with Chirp-Z

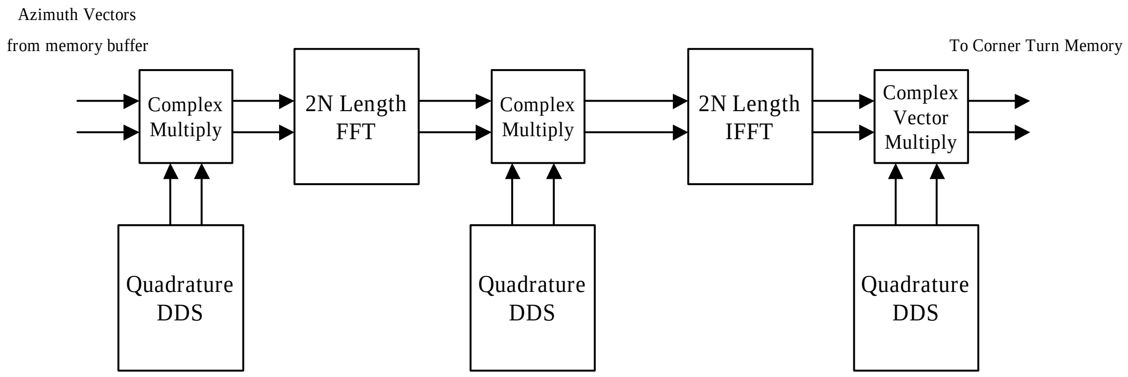


Figure 6: Chirp-Z Transform

The chirp Z approach looks like the better alternative. The quadrature DDS blocks are easily derived from our chirp synthesizers and the corner turn memories and FFT's are required for either approach. Let's go with the chirp Z to eliminate the odd ball sinc interpolator. Nevertheless, the FFT is still a really tough block so let's look at that.

Please note that new chirp parameters for the three quadrature DDS blocks must be calculated and loaded at the PRF rate; that is, once per millisecond.

4 FFT Computation

Many references describe how an N point Discrete Fourier Transform is broken down into successively smaller transforms until it is composed of a series of two point DFT's known as Radix 2 Butterflies. The resulting computational load is proportional to $N*\log_2(N)$ where N is the length of the FFT.[iii]

Figure 7 shows a general block diagram for radix 2 computation of a 4096 point FFT in dedicated logic.[iv]

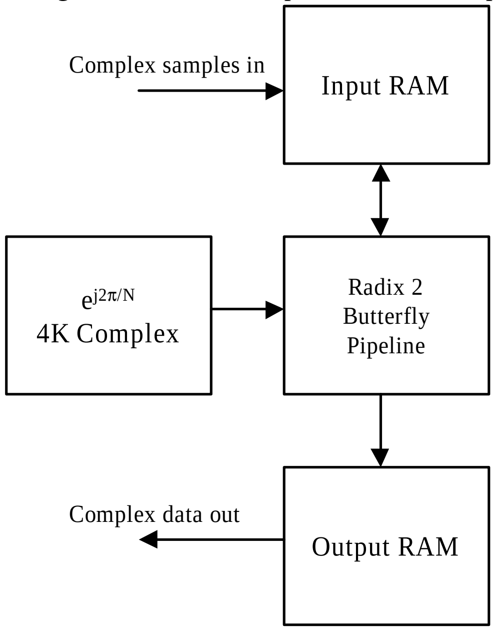
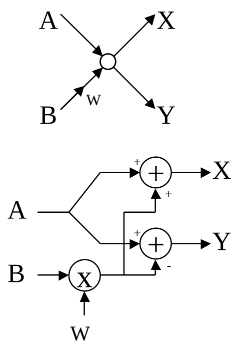


Figure 7: FFT Calculation in FPGA's

Processing proceeds as follows.
1. Data to be transformed is written sequentially into the input RAM at the clock frequency of the system.
2. To compute the N length FFT using a radix 2 butterfly engine, $\log_2(N)$ read/write passes are made through the entire contents of the input RAM where 2 complex numbers are read sequentially from the input RAM and inputted to the butterfly engine. The 2 complex result values are written back to the same addresses in the input RAM (in-place processing).
3. The input RAM is dual port so that reads from one address can occur during the same clock cycle as writes to another address.
4. The W_N^k ROM supplies one complex number per each pair of complex input numbers.
5. On the last pass the results of the FFT are written out to the output RAM in correct (non-bit reversed) order.

The following general points can be made about computing FFT's with dedicated logic as above.
1. Radix 2 FFT needs 12 passes through data to compute 4096 point FFT.
2. New data can be inputted on one out of 12 passes.
3. A radix 4-butterfly engine would require 6 passes on 4096 points. I.e., twice the throughput with four times the logic.
4. The W_N^k ROM supplies data at half the rate as the input RAM so a single ROM can be multiplexed to provide both the real and imaginary coefficients.

Figure 8 shows symbolic and mathematical representations of the radix 2 butterfly. Four unique multiplies and six unique additions are required for each pair of complex input numbers.



$$X=(A_r+B_rW_r-B_iW_i)+(A_i+B_iW_r+B_rW_i)j =(A_r+(B_rW_r-B_iW_i))+(A_i+(B_iW_r+B_rW_i))j$$

$$Y=(A_r-B_rW_r+B_iW_i)+(A_i-B_iW_r-B_rW_i)j = (A_r-(B_rW_r-B_iW_i))+(A_i-(B_iW_r+B_rW_i))j$$

Figure 8: The Radix 2 Butterfly

Figure 9 shows a more detailed block diagram of a possible FFT engine. The logic consists of input and output dual port RAM's, a ROM to supply the complex sinusoidal "twiddle factors", a butterfly pipeline that does the actual multiplies and additions and a control block that sequences the data and twiddle factors through the butterfly engine.

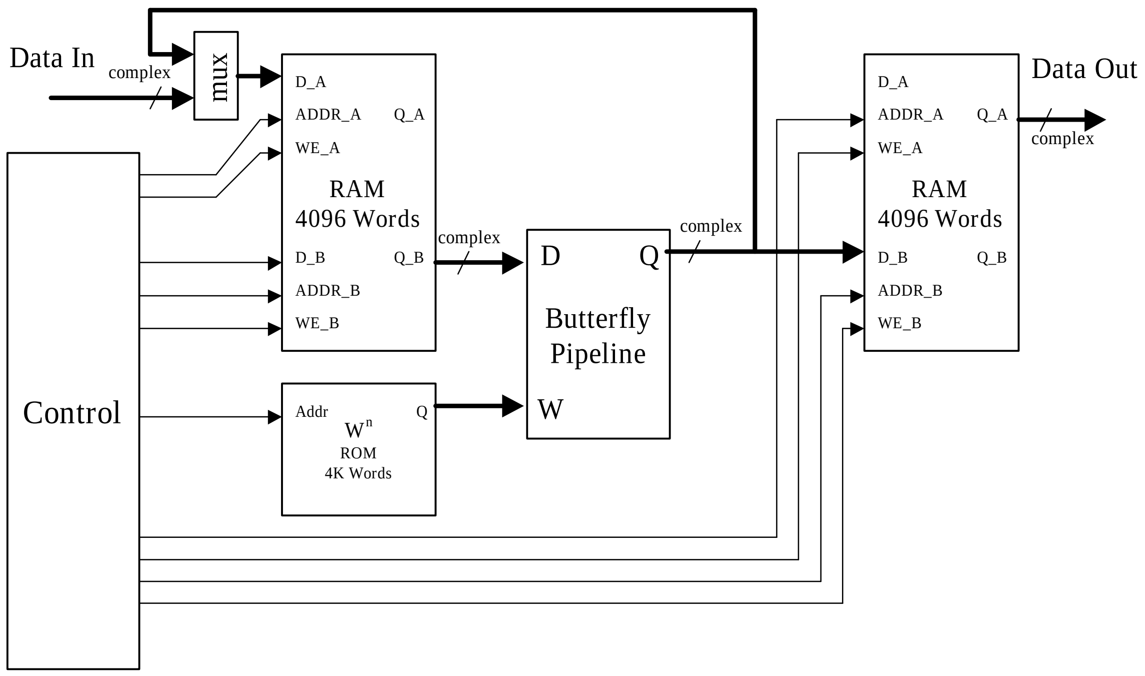


Figure 9: Radix 2 FFT Engine

Figure 10 shows a possible processing pipeline for computing FFT butterflies. The real and imaginary parts of the data samples are applied to the D_r and D_i inputs. The B and A complex inputs are applied sequentially along with the real and imaginary parts of the twiddle factor and a fixed number of clock cycles later the X and Y complex results appear at the Q outputs.

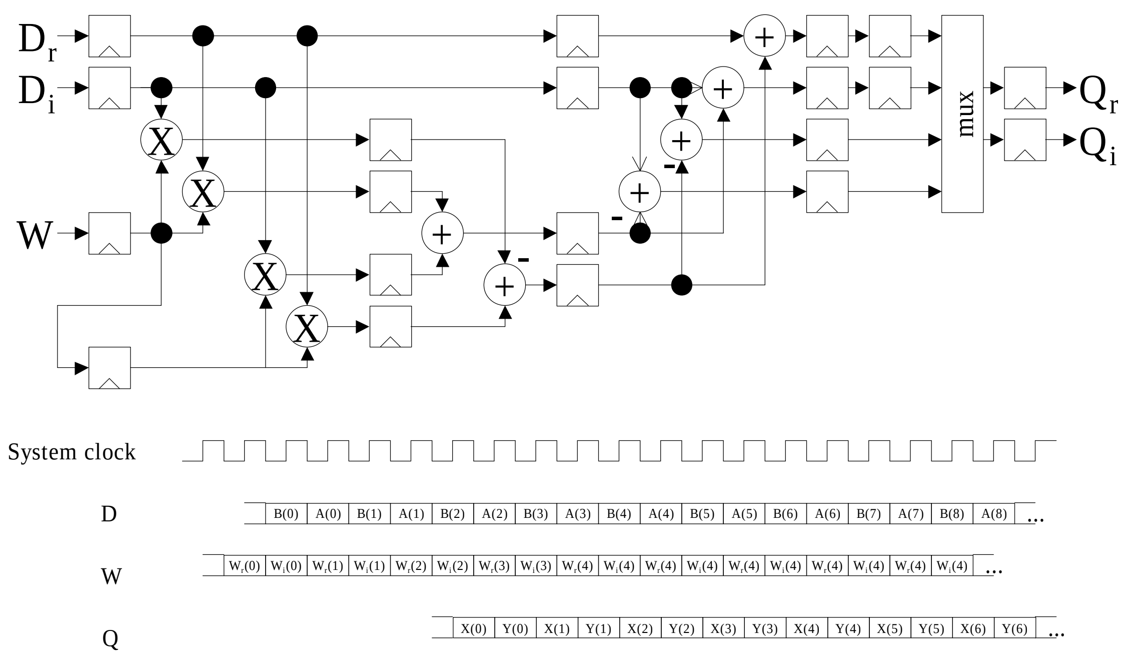

D_r
D_i
W
mux
Q_r
Q_i
System clock
D
B(0) A(0) B(1) A(1) B(2) A(2) B(3) A(3) B(4) A(4) B(5) A(5) B(6) A(6) B(7) A(7) B(8) A(8) ...
W
W_r(0) W_i(0) W_r(1) W_i(1) W_r(2) W_i(2) W_r(3) W_i(3) W_r(4) W_i(4) W_r(4) W_i(4) W_r(4) W_i(4) W_r(4) W_i(4) W_r(4) W_i(4) ...
Q
X(0) Y(0) X(1) Y(1) X(2) Y(2) X(3) Y(3) X(4) Y(4) X(5) Y(5) X(6) Y(6) ...

Figure 10: Radix 2 Butterfly Pipeline

5 Number Systems for SAR Image Formation

5.1 Integer

Xilinx Virtex II FPGA has a natural integer word size of 18 bits two's complement due to the dimensions of the embedded multipliers and block memories.

5.2 Reduced precision floating point

The recent introduction of very large FPGA chips opens up the possibility of doing digital signal processing in floating point math. Compared to integer math, floating point math provides greatly increased dynamic range. Also since many algorithms are initially developed in computer simulations using floating point math, direct mapping to floating point hardware eliminates some of the difficult analysis steps required when porting to fixed point logic.

Figure 11 shows a reduced precision floating point word that matches well to the features of the new FPGA's. Specifically the Xilinx Virtex II chips contain 18 bit embedded multipliers and optimized multiplexing logic. Few good discussions of floating point logic[v] remain in print so this paper includes rough block diagrams showing the logic involved for the purposes of counting logic requirements.

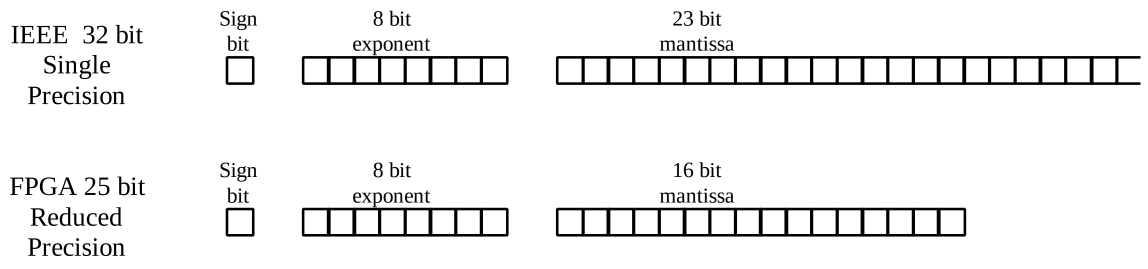


Figure 11: Reduced Precision Floating-Point Format

5.2.1 The floating point multiplier

The reduced precision floating point multiplier below requires about 100 LC's and 1 embedded multiplier.

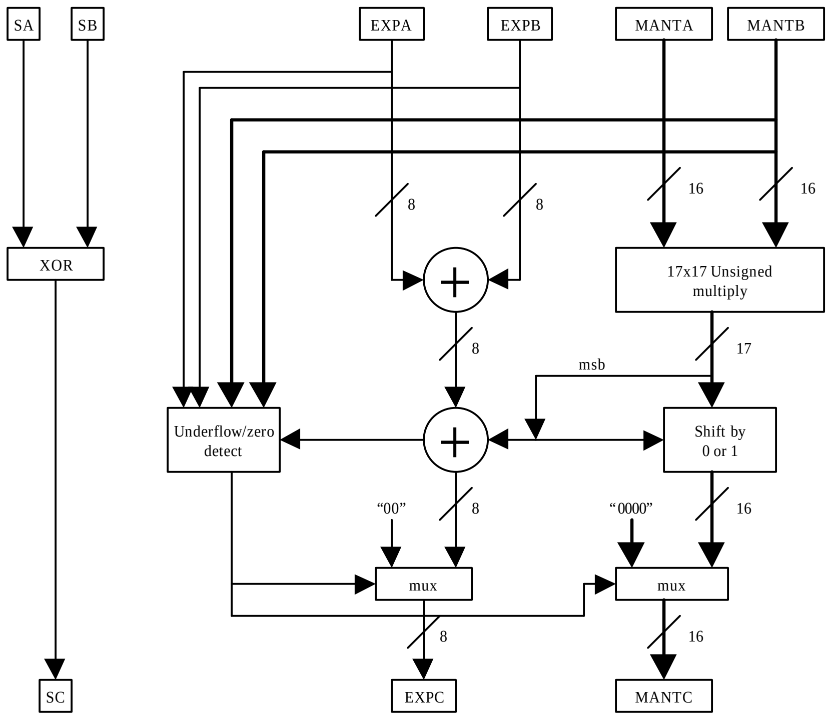


Figure 12: Reduced Precision Floating-Point Multiplier

5.2.2 The floating point adder

The reduced precision floating point adder below requires about 280 LC's.

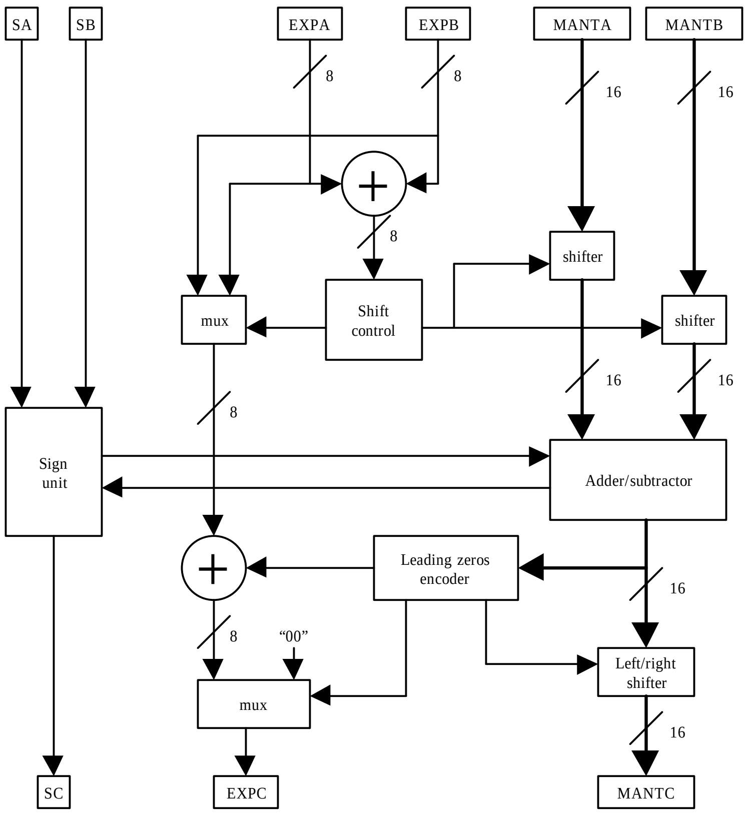


Figure 13: Reduced Precision Floating-Point Adder

5.2.3 Integer to float converter

A final element that might be needed is the integer to floating point converter that uses 62 LC's and 1 embedded multiplier.

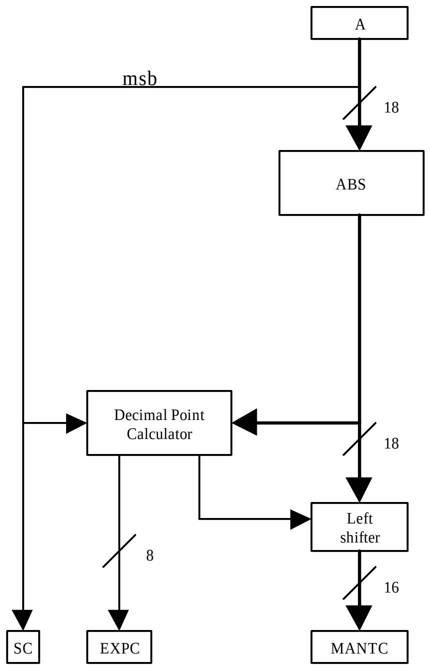


Figure 14: Integer to Floating-Point Converter

6 FPGA Logic Requirements for Polar Format Image Formation

We now have descriptions of the logic required for image formation and we can go through the blocks and add up the resources. We can count logic cells, block memories and embedded multipliers inside the FPGA and external memory chips. Again we are assuming 4096 by 4096 image size.

6.1.1 Integer Logic

Let's count logic assuming 18-bit integer math throughout.

- Corner Turn Memories

Each corner turn memory must hold at least one 4K by 4K 36 bit wide complex image. A common synchronous DRAM configuration is 16M by 8 bits so we would need 5 chips to hold an image. The easiest way to implement the corner turn function is to ping pong two full size image memories such that one is being written row major while the other is being read column major. This requires two full image size memories or 10 chips per corner turn. Each chip uses about 1/3 square inch of board space.

We can guess about the amount of FPGA logic needed to read and write these memories by looking at the data and address bus widths. Let's take the address bus width plus the data bus width and multiply by four (four LC's per address or data bit to control).
(24 address lines + 36 data lines) * (2 banks per CT) = 120 LC's.

- Quadrature DDS's

The DDS synthesizes a complex linear FM chirp by driving two sine lookup ROM's (real and imaginary) with the output of two cascaded accumulators. The value added to the first accumulator determines the chirp rate of the sinusoid and the initial value of the first accumulator determines the starting frequency. The initial value of the second accumulator determines the starting phase of the chirp. An accumulator length of 64 bits is certainly sufficient and we need the two accumulators plus registers for the three chirp parameters for a total of about 64 * 5 = 320 LC's.

The largest practical sine ROM for FPGA implementation is 32K by 13 bits and use two block memories, one embedded multiplier and 30 LC's.

- FFT

The chirp-Z transform requires two 2N length FFT's and the range FFT is an N length transform. Each FFT has input and output RAM's that hold and entire vector of complex samples so the N length FFT needs 4 * 2 * 2 = 16 block memories and the 2N length FFT's require 32 block memories. The twiddle factor ROM needs to be the same length as the input and output vectors but real and imaginary numbers can be multiplexed out on successive clock cycles. Also, the twiddle ROM can contain only the first π of data because the second π is just a negated copy of the same data so the N length ROM needs 2 block memories and the 2N length ROM needs 4.

The butterfly pipeline uses about 500 LC's and 4 embedded multipliers.

The control logic for the FFT engine is more difficult to estimate but we can see from Figure 9 that it drives a number of address busses and control lines totaling about 70 lines in all. Let's multiply by 4 and add 36 for the data input multiplexor. This gives 316 LC's for control and a total of 816 LC's per FFT.

- Complex Vector Multiply

The complex multiply actually requires four scalar multiplies and two adds. Resource requirements are 4 embedded multipliers and about 120 LC's per complex multiplier and there are three complex multipliers per chirp-Z transform.

- Magnitude

This block squares the real and imaginary parts and adds them together. The results pass through a nonlinear mapping function to approximate a square root. Two embedded multipliers, two block memories and 36 LC's should do the trick.

function	quantity	LC's/func	block mems/func	embedded mults/func
Corner Turn Memory	2	120	0	0
Quadrature DDS	3	350	4	2
Complex Multiply	3	120	0	4
8192 Length FFT	2	816	36	4
4096 Length FFT	1	816	18	4
Magnitude Squared	1	36	2	2
FIFO rate buffer	3		8	
Total LC's	4134			
Total Block memories	128			
Total embedded multipliers	32			
Total external memory chips	24			

Table 1: Fixed Point Resource Requirements

6.1.2 Reduced Precision Floating Point

Let's add up the logic resources required to do the image formation algorithm using the reduced precision floating point math of section 5.2. The first thing to note is that the digital receiver produces integer numbers and there is no value in carrying floating point numbers through the first corner turn memory so let's convert to floating point just before the chirp-Z transform. Also the quadrature DDS blocks produce integer values so we have to convert their output to floating point.

Going through the block diagram and converting to floating point operations and widening data paths from 18 to 25 bits we get the following results for logic resource requirements.

function	quantity	LC's/func	block mems/func	embedded mults/func	ext mem chips/func
Integer CT w/fp convert	1	182	0	1	12
FP CT Memory	1	160	0	0	12
Quadrature DDS w/ fp convert	3	422	4	4	
Complex Multiply	3	1000	0	4	
8192 Length FFT	2	2746	54	4	
4096 Length FFT	1	2746	27	4	
Magnitude Squared	1	480	2	2	
FIFO rate buffer	3		12		
Total LC's	13326				
Total Block memories	185				
Total embedded multipliers	39				
Total external memory chips	24				

Table 2: Reduced Precision Floating Point Resource Requirements

7 Throughput Requirements for Polar Format Image Formation

For this analysis we still assume a PRF rate of 1 KHz and a 4K by 4K image size. In this scenario the digital receiver outputs a 4K length complex vector once per millisecond for an average sample rate of 4 million samples per second. Looking at Figure 5 you can see that everything works out well if each block can just keep up with this rate.

The Virtex II FPGA family has a natural clock frequency of about 125 MHz and synchronous DRAM for the corner turn memory is readily available in 133 MHz versions for personal computer main memories so lets assume 125 MHz for the system clock frequency. The complex multipliers, quadrature DDS's and magnitude functions can all be pipelined to run at the system clock frequency so they are no problem.

The FFT blocks' throughput is reduced by the number of passes through the data. The 8192 point FFT's require 13 passes through the data so their overall throughput is 125 MHz/13 = 9.6 MHz which is still more than twice what we need.

The corner turn memories will also require multiple cycles per sample to deal with refresh and other overhead but this is unlikely to add up to more than 8 cycles per sample so we are good here as well.

FIFO rate buffers can be used to match processing rates between computational blocks.

Everything still looks good so let's pull it all together into partitioned hardware.

8 Detailed Hardware Partitioning

8.1 FPGA Partitioning

In order to build working hardware have to divide the logic so that if fits into FPGA chips that have sufficient LC's, block RAM, embedded multipliers and I/O pins. The following members of the Xilinx Virtex II family are listed as likely candidates.

part	LC's	RAM blocks	Embedded multipliers	Maximum I/O pins
XC2V1000	10,240	40	40	432
XC2V3000	28,672	96	96	720
XC2V10,000	122,880	192	192	1108

Table 3: Virtex II FPGA family members

Let's look at an expanded block diagram to see all the bits and pieces. Figure 15 shows the logic for the reduced precision floating point case partitioned into three XC2V3000 FPGA's. Please note that while the entire image former would fit into a single XC2V10,000 this may not be advisable for heat dissipation reasons.

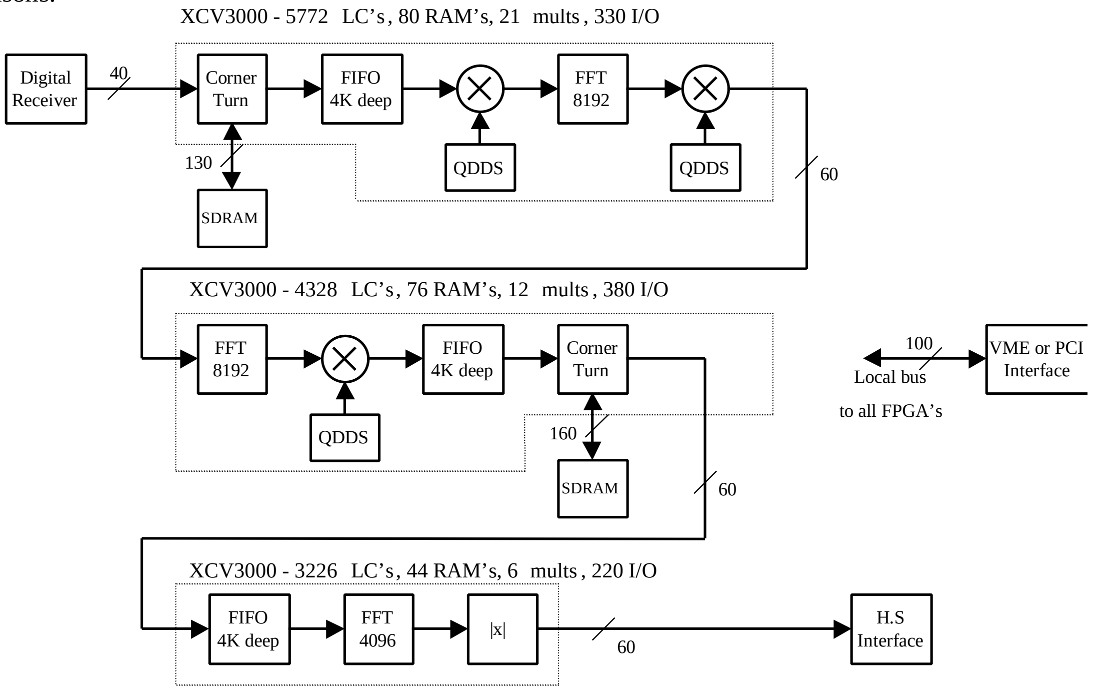


Figure 15: Expanded Block Diagram

8.2 Area, Cost and Power

We can easily estimate the board area required for the implementation of Figure 15.

Part	Package	quantity	sq in./pkg	sq. in	$ each	$	Watts ea.	Watts
XC2V3000 FPGA	FF1152	3	2	6	1500	4500	4	12
MT48LC16M8A2 SDRAM	FB60	24	0.3	7.2	32	768	0.1	2.4
VME/PCI FPGA	PQFP240	1	2	2	200	200	1	1
HS Interface FPGA	PQFP240	1	2	2	200	200	1	1
total sq. in.				17.2				
total component $						5668		
total Watts								16.4

Table 4: Component Area

This area estimate of 17.2 in. sq. compares favorably with the usable area of a 3U cPCI board which is 5" by 3.5" by 2 sides = 35 sq. in.

This component cost of about $6K can be added to the bare board cost and assembly cost of about $2K and combined with test and burn-in for about $5K to get to a very rough $10K to $20K manufacturing cost estimate for an imager former module. This compares favorably with the $250K that is spent for commercial multiprocessor modules to perform SAR image formation.

The estimate for power dissipated in the XC2V3000 FPGA's is only based on the experience that a really hard working FPGA dissipates about 4 Watts. More detailed analysis based on the actual logic implementation and clock speed should be done before committing to this approach.

9 Conclusions

The following points should be made about this analysis of real-time SAR image computation using FPGA's.

- The polar format image formation algorithm presented is missing important details such as:
    - Autofocus
    - Data windowing in azimuth and range
- Therefore all estimates are very rough.
- Nevertheless, SAR image formation in FPGA logic appears to be practical.
- 4K by 4K with 1 kHz presumed PRF using reduced precision floating point math appears to fit on a 3U-size cPCI/VME board.
- The FPGA logic is block RAM limited. In the logic partition presented LC's and multipliers are less than 25% used.
- Deliverable image formation module should cost $10K to $20K to manufacture.
- Module power dissipation would be about 16 Watts.
- The image formation engine described in this document is a very high complexity system requiring several designer-years of work to produce.

10 References

[i] Spotlight-Mode Synthetic Aperture Radar : A Signal Processing Approach
by Charles V. Jakowatz (Editor), Daniel E. Wahl, Paul H. Eichel,Kluwer Academic Publishers; ISBN: 0792396774

[ii] Real-time Polar-Format Processing for Sandia's Testbed Radar Systems, Armin W. Doerry, June 2001SAND2001-1644P

[iii] Rabiner, Gold, "Theory and Application of Digital Signal Processing", ISBN: 0-13-914101-4, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1975.

[iv] High-Performance 1024-Point Complex FFT/IFFT V2.0, Xilinx, Inc. July 5 2000, URL: http://www.xilinx.com/

[v] Introduction to Arithmetic for Digital Systems Designers by Shlomo Waser and Michael J. Flynn, ISBN 0-03-060571-7, CBS College Publishing 1982.

Distribution

1	MS 0529	B. C. Walker	2308
1	MS 0503	A. Schauer	2341
2	MS 0503	P. A. Dudley	2341
1	MS 0519	G. R. Sloan	2345
1	MS 0519	A. W. Doerry	2345
1	MS 0519	D. F. Dubbert	2345
1	MS 0519	S. S. Kawka	2345
1	MS 0519	B. L. Remund	2348
1	MS 0519	T. P. Bielek	2348
1	MS 0519	B. L. Burns	2348
1	MS 0519	S. M. Devonshire	2348
1	MS 0519	J. A. Hollowell	2348
1	MS 0519	M. S. Murray	2348
1	MS 0519	J. W. Redel	2348
1	MS 0519	R. M. Axline	2344
1	MS 0529	C. W. Ottesen	2346
1	MS 1207	C. V. Jakowatz	5912
1	MS0859	R. L. Williams	15351
1	MS 9018	Central Technical Files	8945-1
2	MS 0899	Technical Library	9616
1	MS 0612	Review & Approval Desk for DOE/OSTI	9612