International Journal of Electronics Engineering, 1(1), 2009, pp. 103-112

VLSI Implementation & Design of Complex Multiplier for FFT Using ASIC-VLSI

Amrita Rai[1*], Manjeet Singh[1] & S. V. A. V. Prasad[2]

[1]ECE Department, [2]Head, ECE Department, Lingya's Inst. of Magt. & Tech., Haryana, INDIA

Abstract: *A 16-bit Radix 4 FFT requires a complex multiplier, adder and subtractor that can handle signed two's compliment. The implementation of a complex multiplier uses both the adders and subtractors needed for the FFT. The design of a FFT can be derived from the complex multiplier design. The goal of the device is to have it operate at 100MHz. The complex multiplier must work at 100MHz.Because the design is limited to 40 pins an 8-bit signed two's compliment complex multiplier is implemented. This implementation also contains an 8-bit complex adder and subtractor. If the operation of the entire system is valid, the operation of the adder and subtractor is also valid. If more pins where available a 4 point FFT could be implemented. This could be done with the components of the complex multiplier.*

The schematic is designed first at a logic gate level using design architect (DA). The schematic is simulated to ensure proper functionality. The schematic is used to generate a transistor level layout using schematic driven layout (SDL). The design is checked using a design rule check (DRC) and layout versus schematic (LVS) check. These ensure that the layout rules are followed and that the layout matches the schematic.

Keywords: *16-bit radix 4FFT, complex multiplier, adder, subtractor, VLSI ASIC.*

1. INTRODUCTION

T16-bit Radix 4 FFT Overview: Fixed-point digital signal processors (DSPs) have limited dynamic range to deal with digital data. This application report proposes a scheme to test and scale the result output from each Fast Fourier Transform (FFT) stage in order to fix the accumulation overflow. The radix-4 FFT algorithm is selected since it provides fewer stages than radix-2 algorithm. Thus, the scaling operations are minimized

The FFT is implemented using the butterfly configuration shown in Fig 1. The full 16-bit FFT can be implemented using eight 4-bit FFT modules. The complex multiplier can be used in the design of the 4-bit FFT. [1]

2. IMPLEMENTATION OVERVIEW

The 8-bit signed two's compliment complex multiplier contains both a complex adder and subtractor. A complex multiply is performed in the following manner: [8]

$$(a + jb)\,(c + jd) = (ac - bd) + j\,(bc + ad)$$

The real and imaginary parts of the output will be kept separate. This design requires four signed two's compliment multiplies, as well as a signed two's compliment adder and subtractor. The complex multiply will require the addition of a FIFO, at the beginning, to sample the inputs and at the end to sample the result of the complex multiply. The basic structure is as follows: [3]

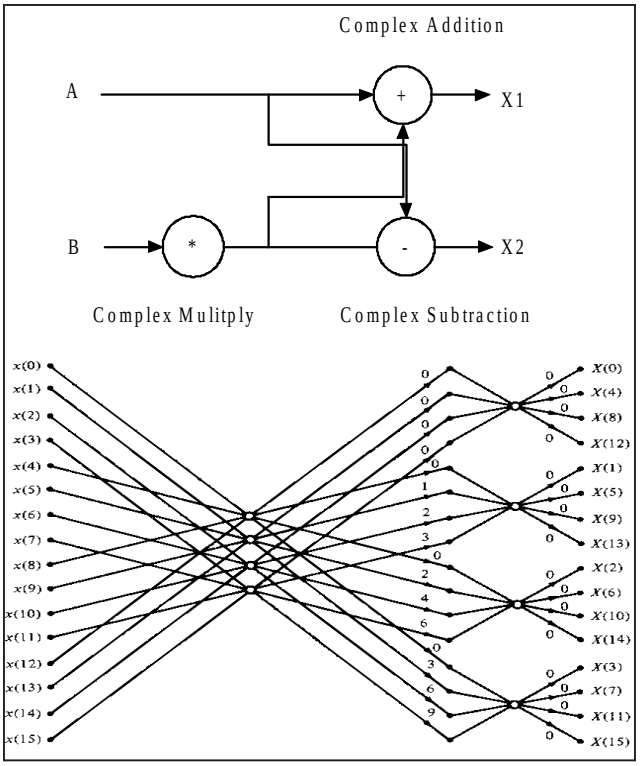

Figure 1: The FFT Butterfly and the Block Diagram

The multiplier, adder and subtractor are all built using full and half adders. Half adders find the sum of two binary numbers and provide a sum and a carry. A full adder is similar

*Corresponding author: amritaskrai@yahoo.com

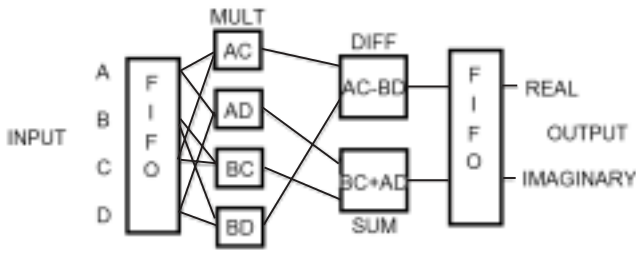

Figure 2: Block Diagram of Complex Multiplier

to the half adder however; it has an input for a carry bit. Full and half adders can be constructed using logic gates; however, this is not the most efficient solution. The ADK library provides a full and half adder block. This block, much like the logic gates provided, has a layout already done. The ADK full adder block provided takes up approximately half

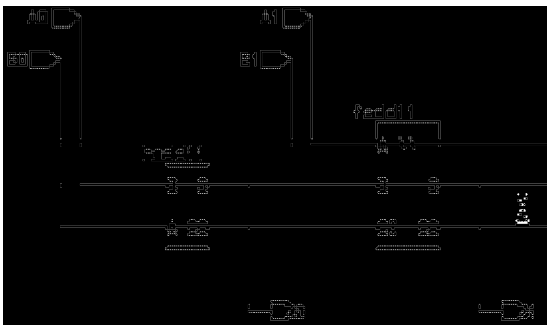

Figure 3: ADK Half Adder and Full Adder

the area of the logic gate implementation. The ADK full adder block uses 31% less power than a full adder implemented in logic gates. The delays of the two different full adders are very similar. The provided full adder in the ADK library will be used to implement the design. [2]

An adder and a subtractor can be constructed using the full adder blocks. This is done using a ripple carry approach. A ripple carry adder computes the sum of the least significant bit first, and then allows any carries to propagate to the next full adder. The sum of the first two bits is the least significant bit of the answer. The ripple carry subtractor operates in a similar manner, however the two's compliment of the second input must be taken. This is done by inverting the second input and setting the carry in high on the first full adder. This is the equivalent of taking the two's compliment. The schematics are as follows: [3, 4]

The multiplier is implement using an add and shift algorithm. Special care must be taken when designing the multiplier so that it properly handles signed two's compliment numbers. Because of the nature of signed two's compliment numbers, no special consideration needed to be taken when creating the adder and subtractor. With the multiplier the sign bit must be properly dealt with or the multiplier will not function properly. The multiplier schematic is shown in Fig. 6. [7]

A set of FIFO will be used on the input to sample and hold the inputted waveform. If the input is changing with time, it will confuse the complex multiplier. The FIFO will

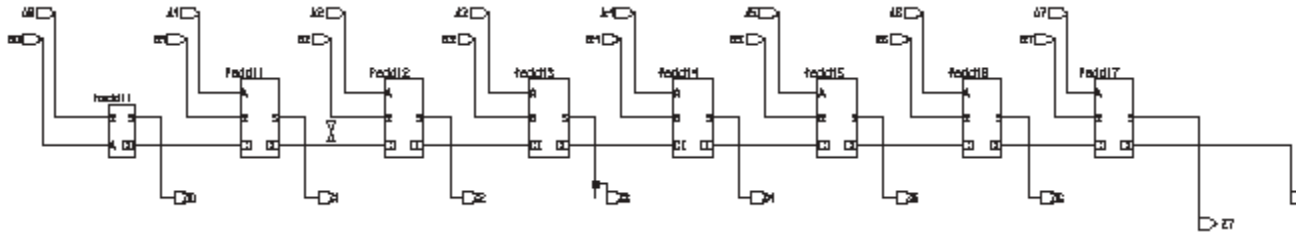

Figure 4: Ripple Carry Adder (8-bit)

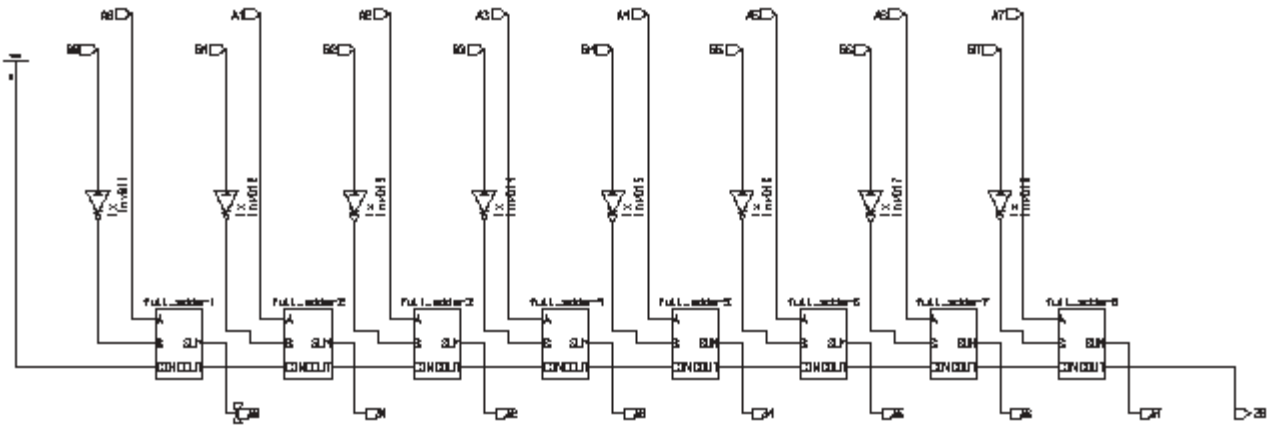

Figure 5: Ripple Carry Subtractor (8-bit)

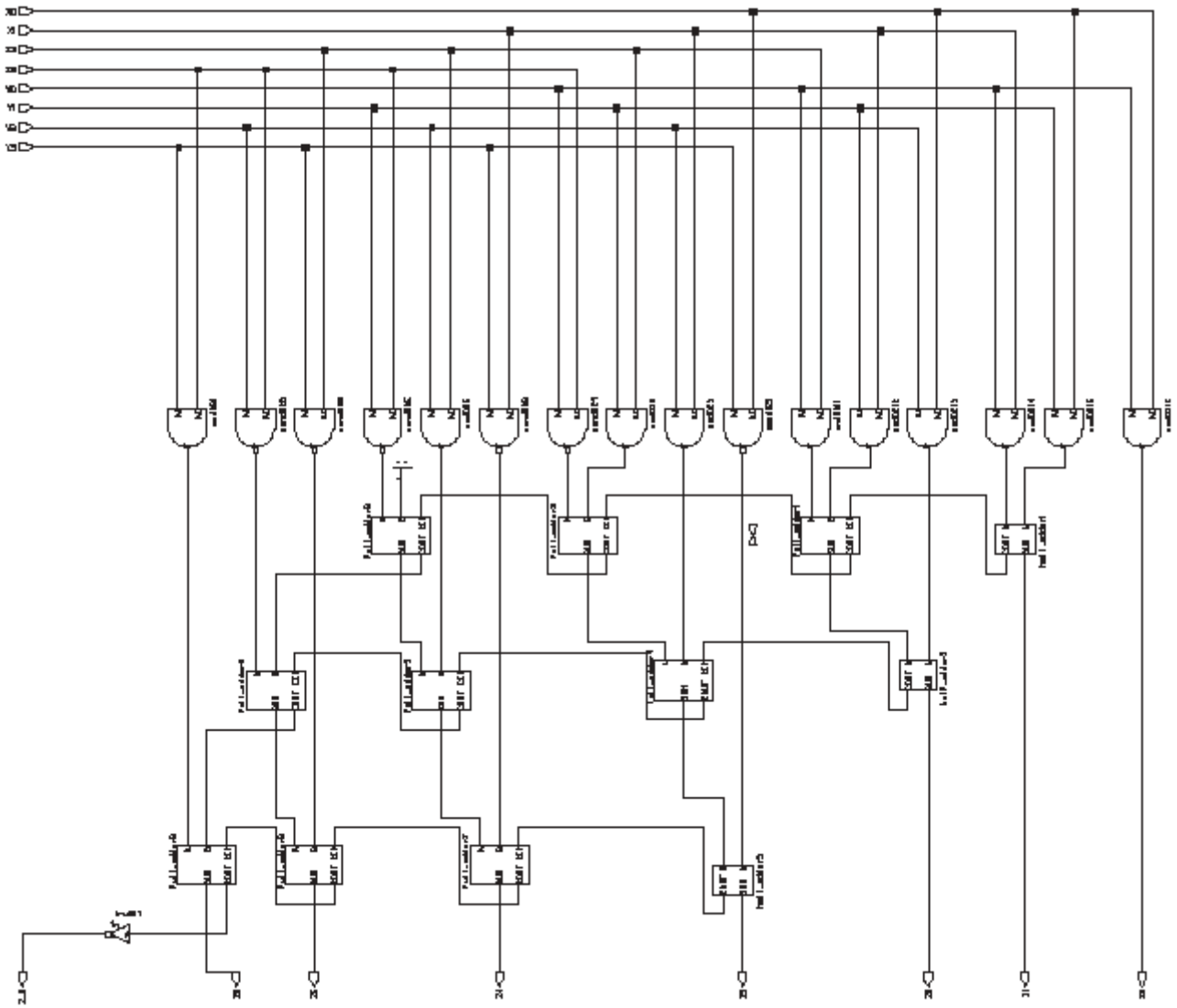

Figure 6: 4-bit by 4-bit Signed Two's Compliment Multiplier

sample the input and hold it, until the complex multiplier is done calculating the answer. A set of FIFOs are also used at the output. There is a delay before the complex multiplier has computed the correct answer. The output FIFOs will sample and hold the output after this delay. This way the output is a discrete waveform, where all outputs change at the same time. This can be seen in the comparison of Fig.'s 15 and 16. A timing diagram is provided below that illustrates the timing of the final circuit. [7]

In order to fulfill the specifications of the design, everything was wired according to Fig. 2. The inputs are labeled A, B, C and D each with 4 bits. The outputs are 7-bits and are labeled *J* and *R* for imaginary and real, the full schematic is as follows:

3. TESTING OF DESIGN

After completing the schematic, various simulations were run to verify the functionality of each design. Simulations were also used to determine the delays occurring within the circuit. Testing all input combinations is not always possible. Determining the most relevant test cases is important to confirm functionality and performance. In the simulation test runs pulsed waveforms were used to discover circuit delays. [8]

$$\begin{array}{r} {\scriptstyle 1\,1\,1\,1\,1\,1\,1} \\ 10101011 \\ +01010101 \\ \hline 100000000 \end{array}$$

The full adder, even though it is included in the ADK library and should work properly, is tested for functionality and to see delays. Each possible combination is discovered using pulsed waveforms with periods that double. The delays come from the time it takes the circuit to produce the correct answer. There are narrow spikes that indicate there are delays before the correct answer is computed. This delay will grow as more full adders are used. [8]

The case used to simulate the ripple carry adder is shown below. This case has the maximum number of carries. As seen in Fig.11, there is a delay propagating through the circuit

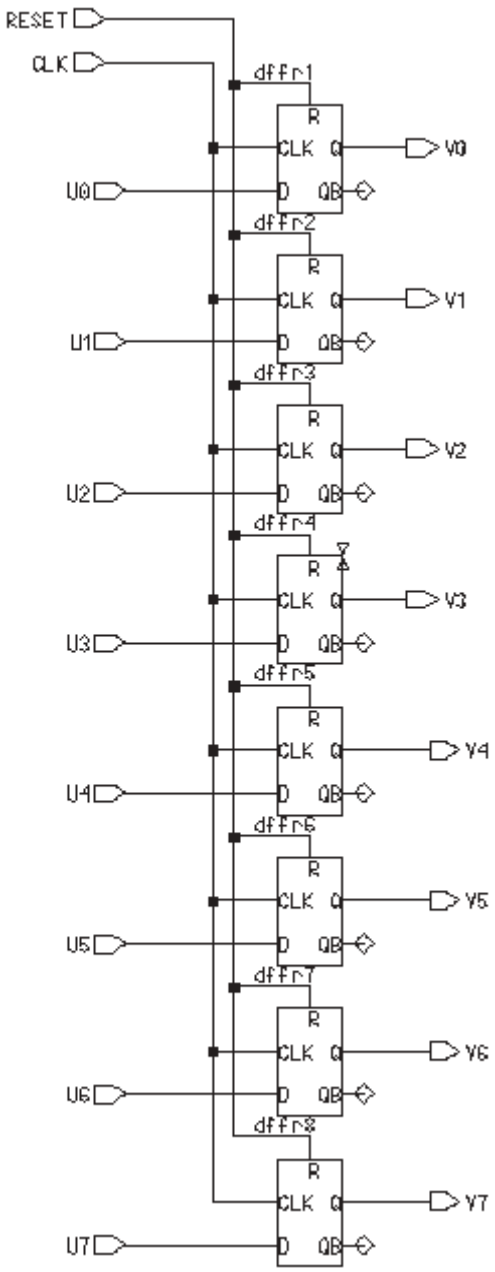

Figure 7: FIFO using D Latches with Reset

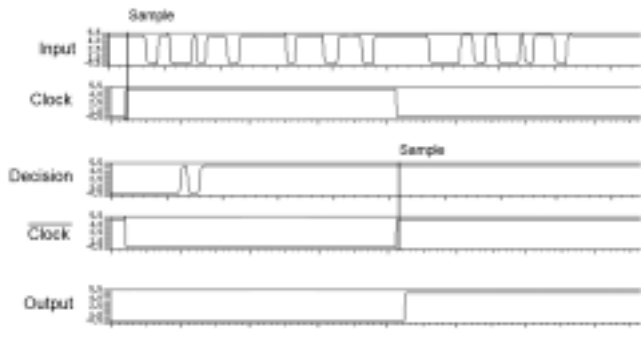

Figure 8: Timing Diagram

as each carry takes place. The total delay time for the ripple carry adder in which the maximum number of carries is involved was seen to be less than 5ns. This is considered the worst case delay [6]

The case used to simulate the ripple carry subtractor involves pushing all inputs from high to low. If the inputs are all high, the difference is zero thus the output would result would be all low. If the circuit is pushed from all high, to all low there will be a borrow across every full adder. This case would cause the maximum delay. The maximum delay for this worst case simulation is about 5.5ns. [5, 6]

Fig. 13 is just one of the many simulations run on the multiplier. Many cases that involve positive-positive, positive-negative and negative-negative sets of inputs were used. The simulation results are for the product of -3 and -6. The inputs are pulsed in at 0 seconds and return to zero at 10ns. The result, 18, is properly calculated with the correct sign bit (Z_S). The delay in this particular simulation is less than 4ns. For all simulations the maximum delay found was also under 4ns. [4, 8]

Simulations were then run on the final schematic design without using FIFOs (un-clocked). Many simulations were run and the simulation shown in Fig. 14 is just one of them. The delay in the final circuit is less than 8ns, much like the delay in this simulation, on output J6. The answer is for the imaginary part is 91(01011011) which come from adding the two products of 7*7 and 7*6. The real part is 7(00000111) which come from the difference of 7*7 and 7*6. The output of the un-clocked system is very choppy and it is hard to read. The clocked version has a set of FIFOs on the output. The FIFOs will sample and hold after the correct amount of time to create an answer that is discrete, and easily readable. The same simulation is shown below in Fig. 15. Both the imaginary and real components are easy to read. [5, 8]

4. LAYOUT

With the final design completed and simulated the schematic is used to generate the transistor level layout. This is known as schematic driven layout or SDL. All of the blocks used (full adder, dff, etc.) have a layout associated with them. As the full layout is automatically generated, each individual block layout is placed with connectivity. Connectivity information is used to place routing. The routing on the final layout was done using the automatic route command. This produces many errors in the DRC, and cannot route all paths. These errors can be fixed, and the other paths are routed by hand. VB_{DDB} and ground rails run along the sides of the layout, which connect ground and VB_{DDB} to each string of instances. The instances are not grouped as they are in the schematic. To save space, the blocks are placed so connectivity is simple, thus requiring less or shorter routes. This makes the circuit hard to decipher however, due to time and area constraints, it was the best option

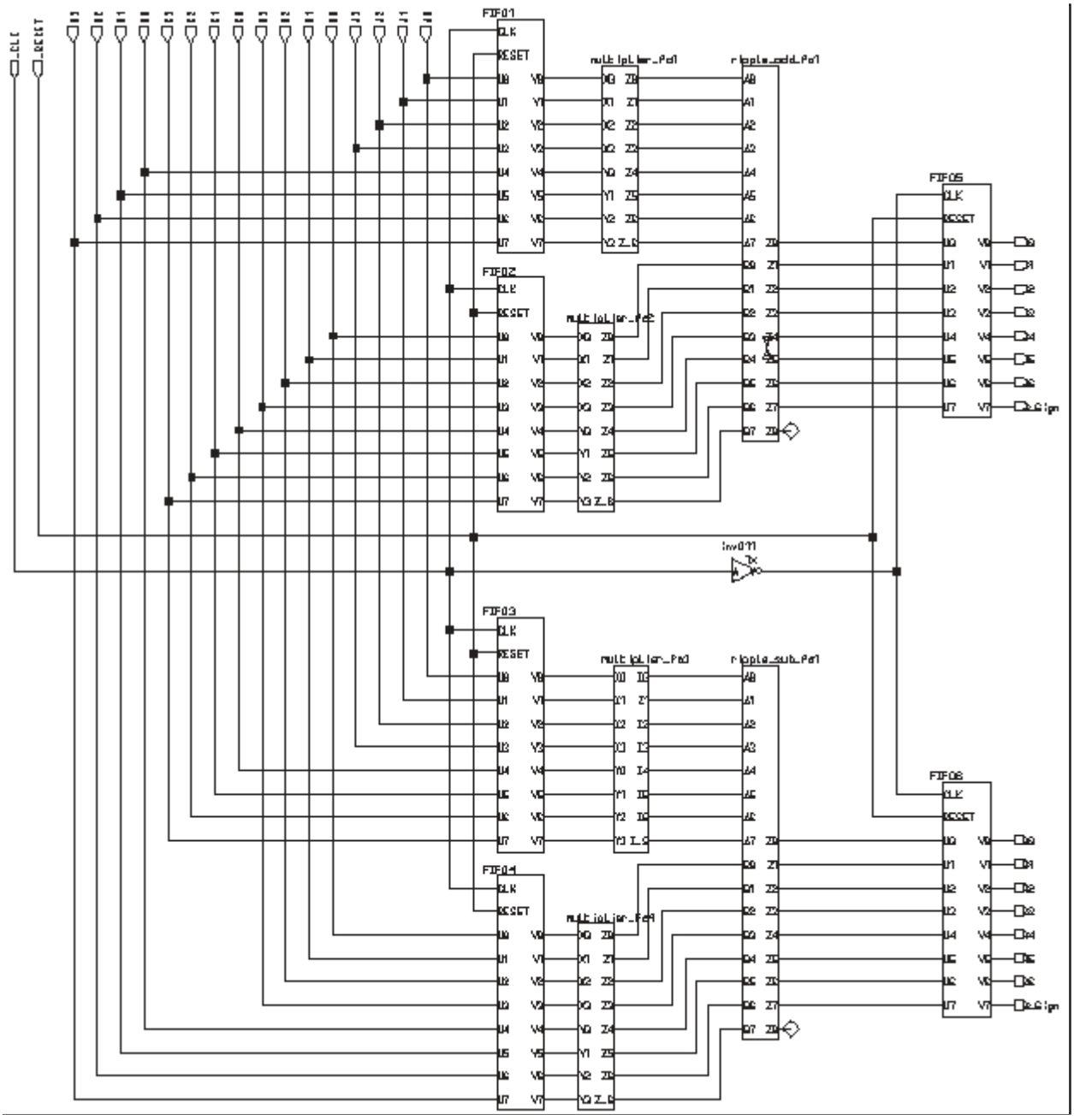

Figure 9: Full Complex Multiplier

The blue, purple and grey represent different layers of metals in Fig.'s 16 and 17. When metals on different layers cross they do not connect. This allows the crossing of paths, without shorting the design. In order to connect two layers, a via must be used. What are not shown in the layout are the poly, n-diff or p-diff areas. These are essential in forming transistors and are included in the layout.

The final layout was 2048λ by 2056λ. For a 0.5 micron process the design is 1.024 mm by 1.028 mm. This fits well into the 2 mm squared specification and will allow plenty of room for routing. The final transistor count is 1802 CMOS transistors.

The final step is to wire the layout into the pad. The pad is used to connect the actual transistors to the pins of the final chip. This connection is made with fine gold wires that are soldered onto each pad. The routing between the layout and each pad was performed by hand. All of the routing goes around the outside of the layout to help prevent shorts. The date and initials are written in metal 3, or the outermost layer. They should be visible on the final chip. The final design is checked with the DRC and LVS to ensure that it has no layout errors and that the layout matches the schematic.

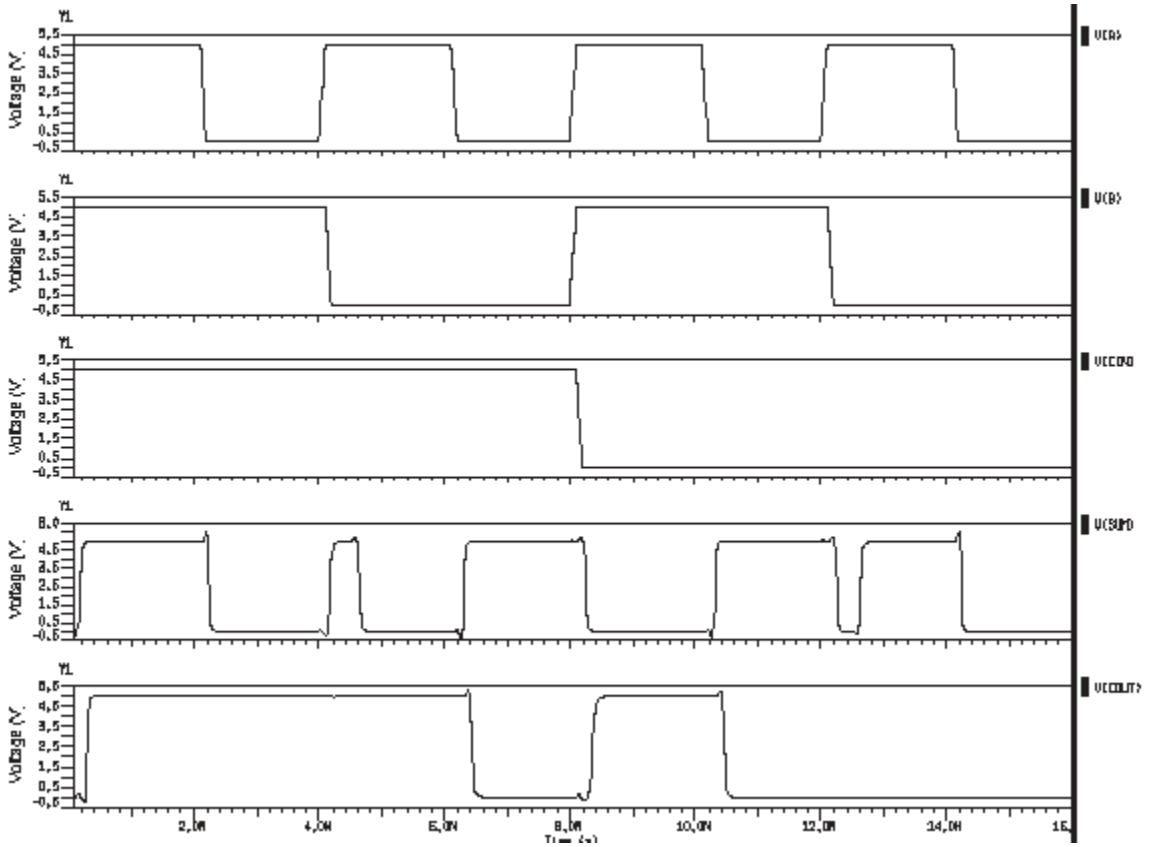

Figure 10: Full Adder Simulations

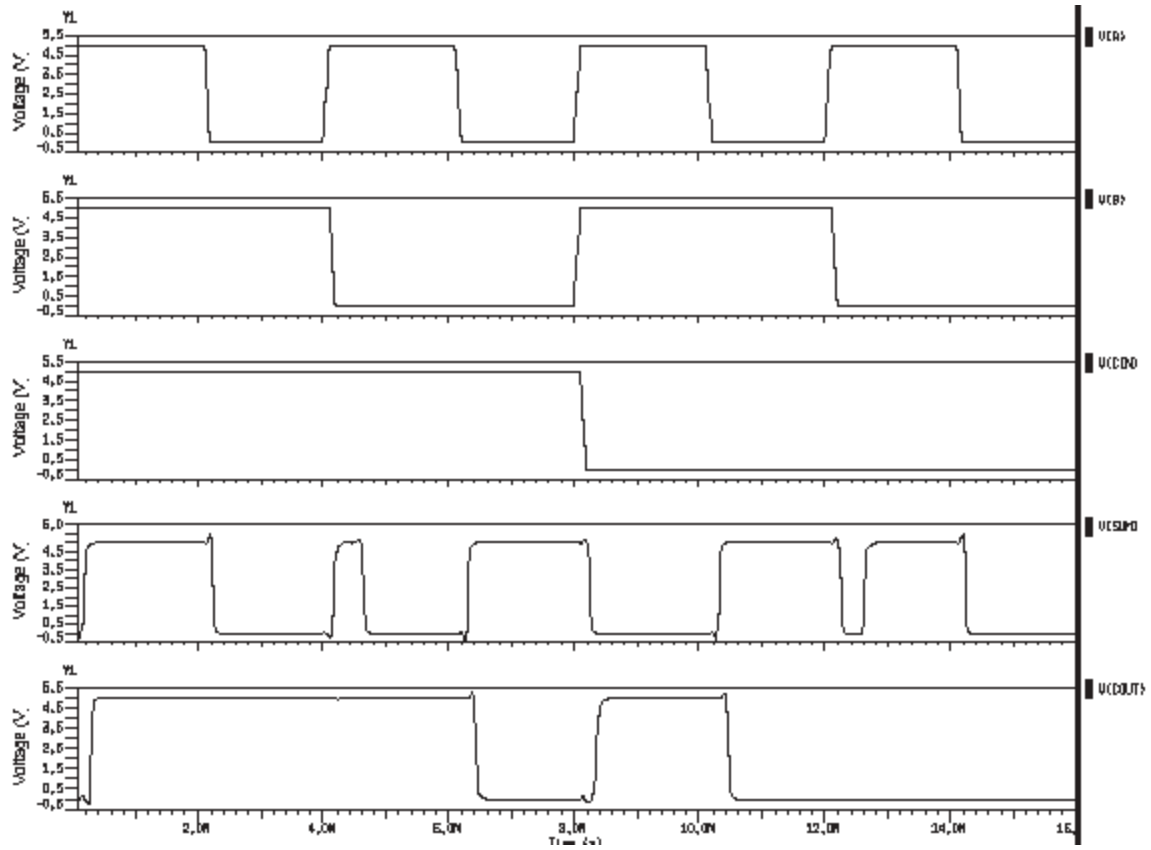

Figure 11: Ripple Carry Adder Simulation

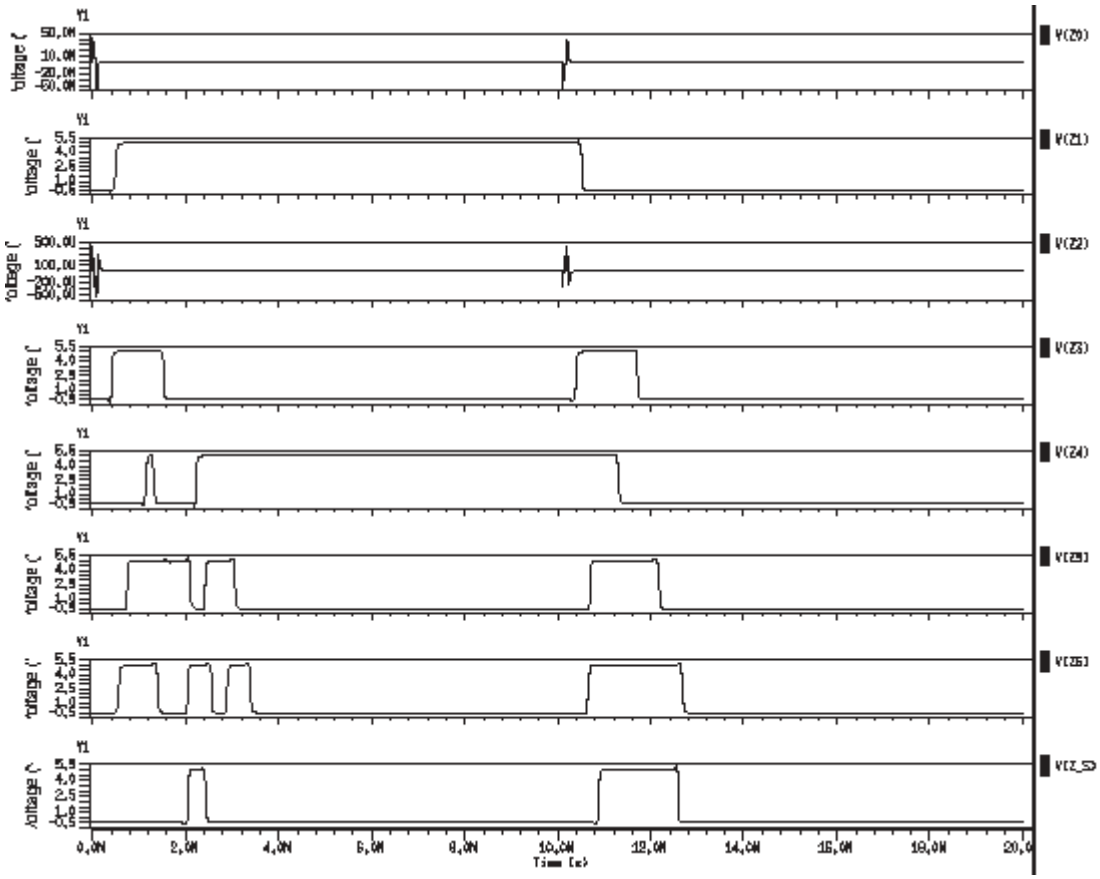

Figure 12: Ripple Carry Subtractor Simulation

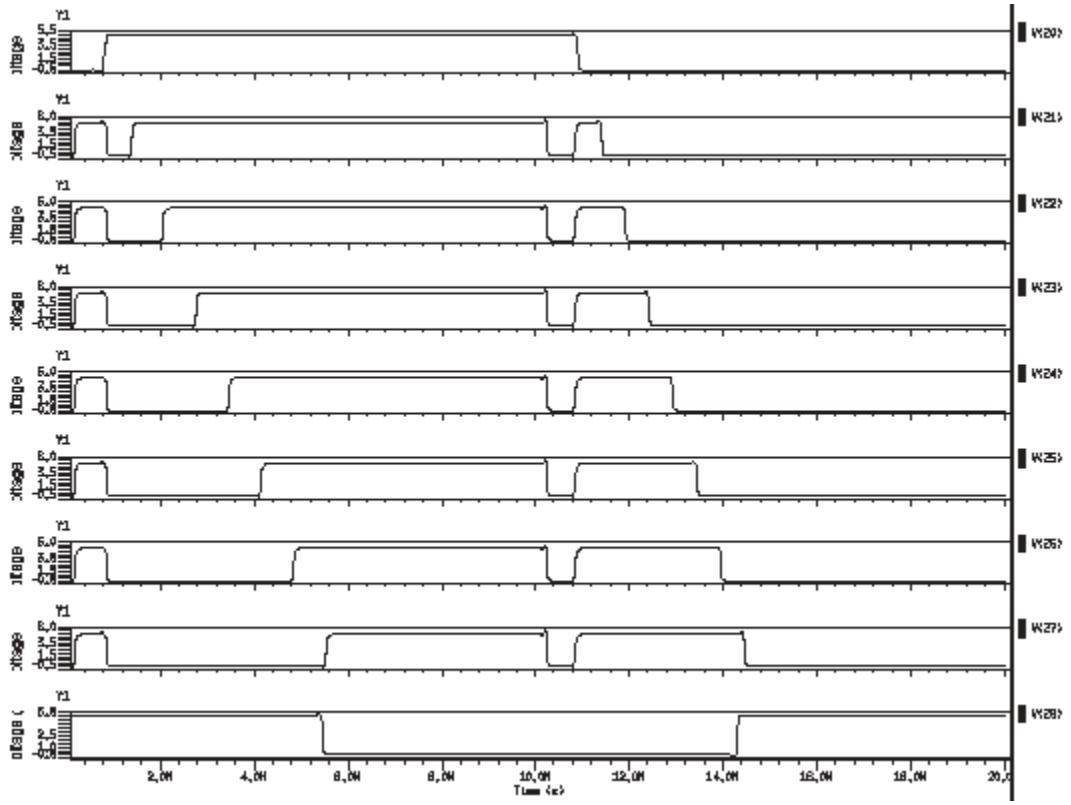

Figure 13: 4-bit by 4-bit Multiplier Simulation

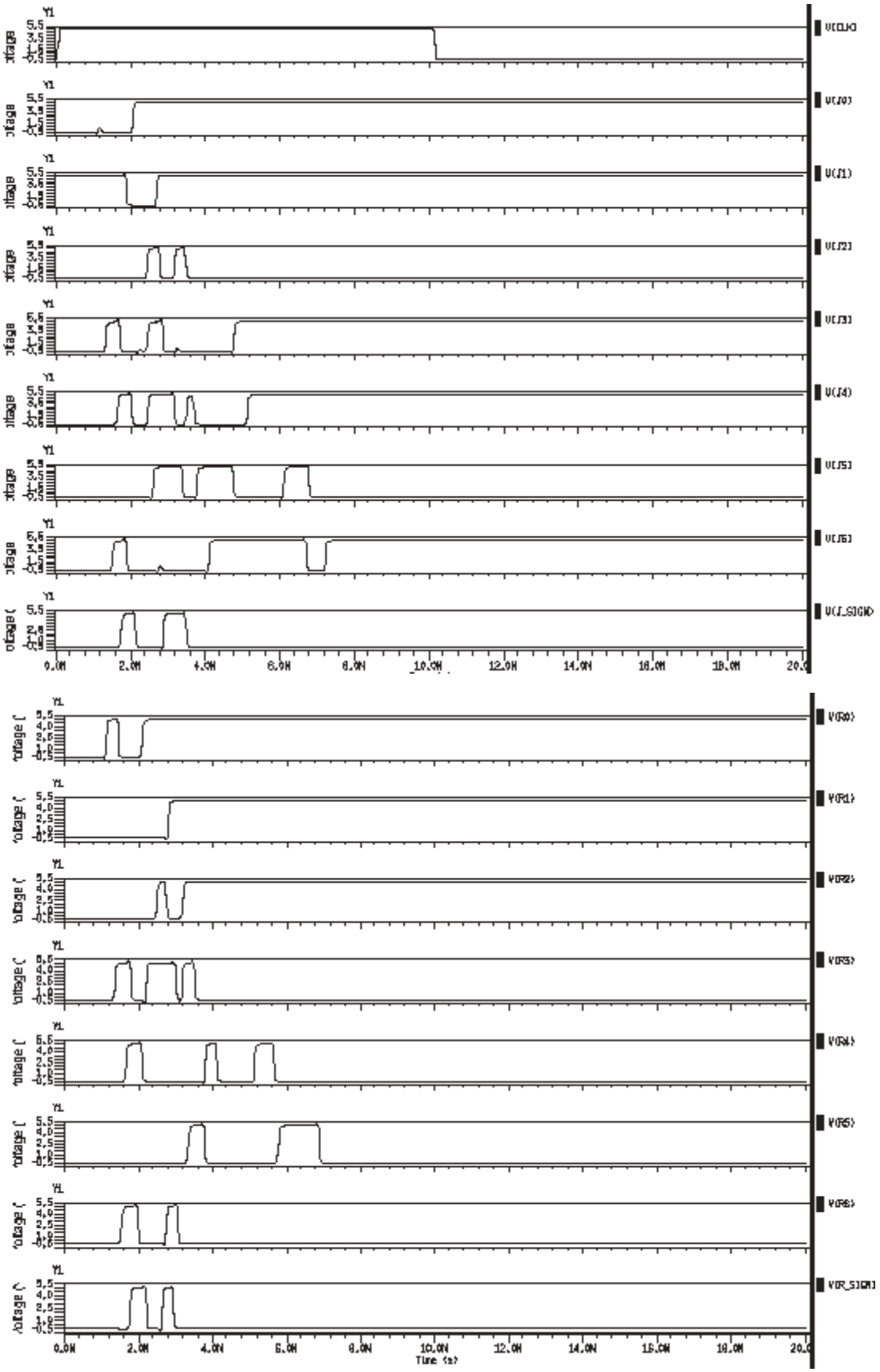

Figure 14: Un-clocked Complex Multiplier Simulation (no FIFO's)

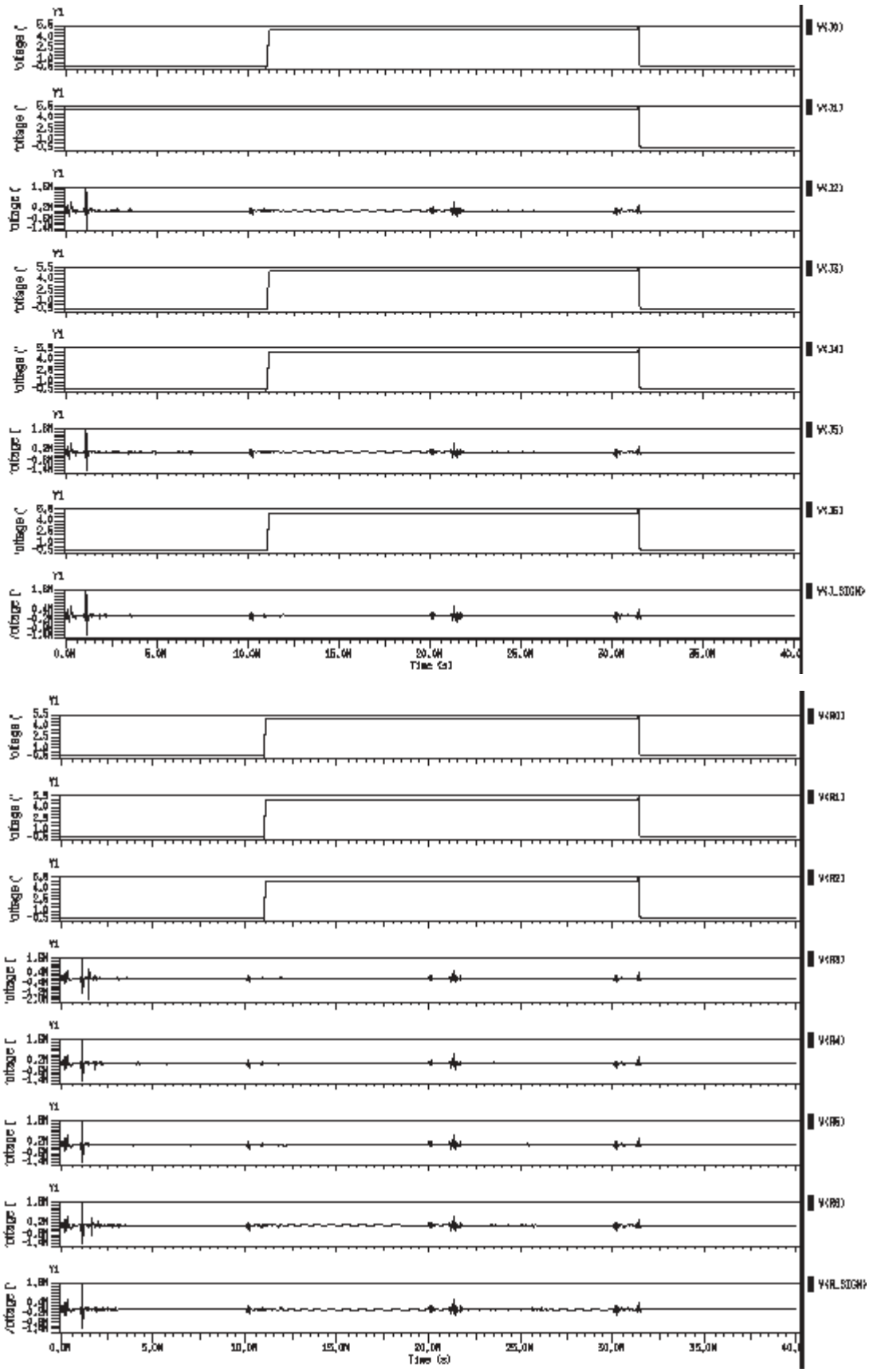

Figure 15: Clocked Complex Multiplier Simulation

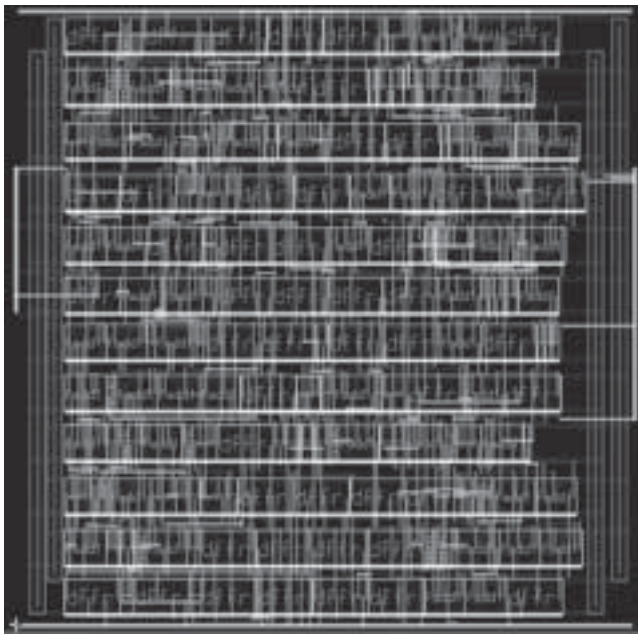

Figure 16: Automatic Layout and Routing Results

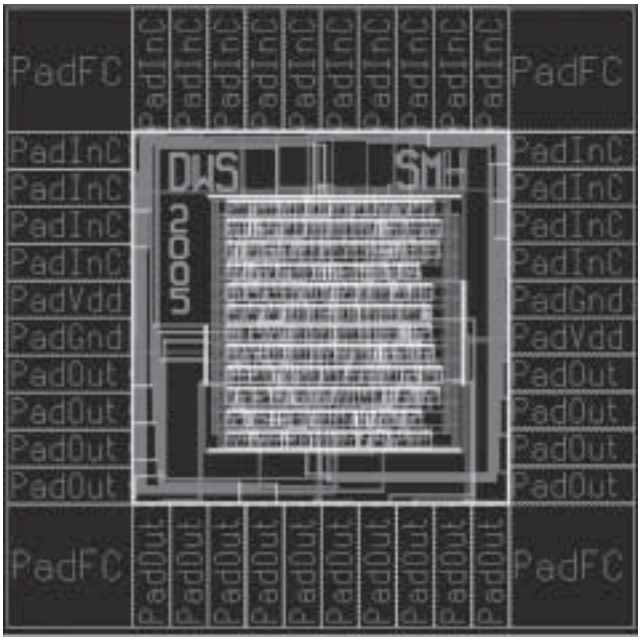

Figure 17: Final Layout, Hand Routed into the Pad

5. CONCLUSION

Comparison between theoretical simulations and actual performance results. This will be a discussion on the characteristics of the actual chip versus the simulations. Perhaps this section will include what should have been done to get the chip to work properly or within the goals. The experience in the lab produced many lessons that were learned in early stages of the project. For example, when testing a schematic composed of many layers of abstraction, it is important to start at the lowest level and work up. Making the correct decisions at each progress point is a very beneficial strategy. If a decision is incorrect early on, the whole system must be fixed. Going back and fixing errors is very time consuming.

In order to make sure that each step or level of abstraction was correct before moving on, it was rigorously simulated. Also different designs were examined to see which had the best performance. With this said, we had to go back from the layout stage, and fix almost the entire system. This set us back several days. If the error was discovered early on, the layout would have been completed a week earlier.

Working with new software is difficult. With little time to learn how to do each operation needed, it was difficult to get each stage of the design completed. Many hours were spent trying to Fig. out how to get things accomplished in the program. With the experience gained, the schematic and layout of a design could be finished in a matter of weeks, not months.

Even now with the final design completed, there are little things that could have been changed to help weak the circuit. For example a better clock timing mechanism would help the circuit go faster, without the use of a duty cycled clock.

REFERENCES

[1] M.M. Mano, C.R. Kime. Logic and Computer Design Fundamentals: 2P^{ndP} Edition Updated. New Jersey. Prentice-Hall Inc., (2001).

[2] N.H.E. Weste, D. Harris. CMOS VLSI Design: A Circuits and Systems Perspective. Boston. Pearson Education Inc., (2005).

[3] A.M. Despain, "Very fast Fourier Transform Algorithms Hardware for Implementation," *IEEE transactions on computers*, **C-28**, (5), May 1979, 333-341.

[4] Deepak Kumar, Tala. Asic-World., http://www.asic-world.com/digital/index.html, Oct 7, 2005.

[5] http://www.seas.upeen.edu/~ese201/lab/CarryLookAhead/CarryLookAh adF01.html,Oct 7, 2005.

[6] "Multipliers.", Tima Laboratory., http://timacmp.imag fr/~guyot/cours/oparithm/englisg/ Multip.htm, Nov 14, 2005.

[7] Bigelow, Ken. "Digital Logic." Play-Hookey.com. <http://www.playhookey.com/digital/> Oct 12, 2005.

[8] "Project Assignment: Fast Fourier Transform Components." *Introduction to VLSI*. <http://www.ece.unh.edu/courses/ece715/proj_options.htm> Sept 28, 2005.