ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

FFT Design Using Reliable Multiplier with Adaptive Hold Logic

A V V Hanuman Sai Krishna
PG Scholar in VLSI Design,
Department of ECE,
Dhanekula Institute of Engineering & Technology,
Ganguru, Krishna Dist., Andhra Pradesh, India.

A Sivannarayana
Assistant Professor,
Department of ECE,
Dhanekula Institute of Engineering & Technology,
Ganguru, Krishna Dist., Andhra Pradesh, India.

Abstract:
FFT and IFFT algorithm plays an important role in design of digital signal processing. Thispaper describes the design of Decimation in Time-Fast Fourier Transform (DIT-FFT). The proposed design is implemented with radix-2, based 4 point FFT. Whereas digital multipliers are among the most critical arithmetic functional units.The overall performance of these systems depends on the throughput of the multiplier.Here a reliable multiplier with adaptive hold logic is used. This approach reduces the multiplicative complexity which exists in conventional FFT implementation. For the number representation of FFT fixed point arithmetic has been used. The design is implemented using Verilog HDL language.

Keywords:DIT-FFT, Complex multiplication, Verilog, Radix-2, Adaptive Hold Logic.

INTRODUCTION

FFT and IFFT commonly used algorithm for processing signals. It can be used for WLAN, image process, spectrum measurements, radar and multimedia communication services. Now a days, FFT processors were using in wireless communication systemsthat are having fast execution and low power consumption. These are some most important constraints of FFT processor. Complex multiplication is main arithmetic operation used in FFT/IFFT blocks. This is the main issue in processor. It is time consuming and it consumes a large chip area and power. When large point FFT is to be designed, it increases the complexity. To reduce the complexity of the multiplication, there are two methods one simple method is to real and constant multiplications take the place of complex multiplication. The other method is non-trivial complex multiplication is wipe out by the twiddle factors and fulfils the processing with no complex multiplication.In our work FFT algorithm is implemented in radix 2. The basic idea of these algorithms is to divide the N-point FFT into smaller ones until two point FFT is obtained. Hence the algorithm is called radix-2 algorithm. Here the multiplier used is the reliable multiplier design using adaptive hold logic. In this again we use Row/Column Bypassing Techniques to reduce the Dynamic power and delay for the multiplication process.The paper is organized as follows. Section II describes radix 2 DIT FFT algorithm, and fixed point number representation, Section III describes the complex multiplication technique, section IV explains implementation of 4 point FFT, Section V presents ASIC Synthesis results and Layout and the Last section concludes the design.

Discrete Fourier Transform (DFT)

The Fourier transform is mathematical method of changing time representation of signal into frequency representation. It transforms one function from time domain to frequency domain. The DFT of a input sequence x[n] can be computed using the formula given:

$$X(K) = \sum_{n=0}^{N-1} x(n)\,.W_N^{nk} \qquad 0 \leq k \leq N-1 \qquad (1)$$

Where $W_N = e^{-j2\pi/N}$

Radix 2 DIT – FFT Algorithm

The basic module for implementation is butterfly module which is shown in the fig 1.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

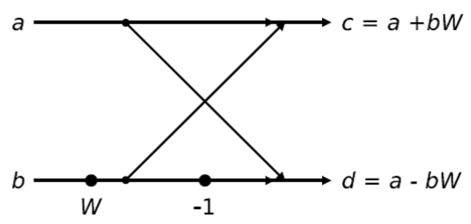

Fig. 1 Radix-2 Structure

There are two inputs called a, b and two outputs c, d and twiddle factor W. Output is as follows:

$$C = a + bW \quad (3)$$

$$D = a - bW \quad (4)$$

With these butterfly units we can built whole FFT structure. If N is the input for FFT then stages are required, each stage requires N/2 butterflies. As we can see from above fig that one butterfly unit requires 1 complex multiplier and 2 adders for executing single butterfly. For every DIT-FFT radix – 2 algorithms with N input sequence requires N/2 multipliers and N adders.

Number representation

For number representation of both real as well as imaginary fixed point scheme is followed so that we can reduce the complexity of using floating point arithmetic. The twiddle factor used is in complex form real and imaginary. To represent this number we are multiplying these numbers by scaling factor which is where s N. So that twiddle factor is rounded up in integer number. For complex multiplication we require twiddle factor magnitude and sign bit so s+1 bit are required to represent twiddle factor. As the input given to the design can also be in floating form then we can apply the same scheme of rounding up input in the integer. As we are scaling up the input or twiddle factor we have to scale down the signals at the output and we have to accept some rounding errors. Simple way for scaling down is by multiplying or using shifting operation. It is as simple as we are multiplying one no with something and dividing the same number we will get the original number.

COMPLEX MULTIPLIER

Here FFT is implemented by using the proposed multiplier.An Aging-aware reliable multiplier design with adaptive hold logic (AHL) is used for multiplication. This multiplier is based on the variable-latency technique and can adjust the AHL circuit to achieve reliable operation under the influence of NBTI and PBTI effects. To be specific, the contributions of this multiplier are summarized as follows:

1) variable-latency multiplier architecture with an AHLcircuit [4].

2) A reliable multiplier design method that is suitable for large multipliers. The experiment is performed in 4, 16, 32 and 64-bit multiplications.

3) An FFT was designed by using this multiplier.

Column bypassing and Row Bypassing Multiplier

The column bypassing multiplier [5] is constructed asfollows. First, the modified FA cell will be designed. If *aj*= 0, the FA will be disabled. For an Array multiplier, there are only two inputs for each FA in the firstrow (i.e., row 0). Therefore, when *aj*= 0, the two input ofFA0,*j*are disabled, and thus it output carry bit will not bechanged. Therefore, all three inputs of FA1,*j*are fixed, whichprohibit its output from changing.

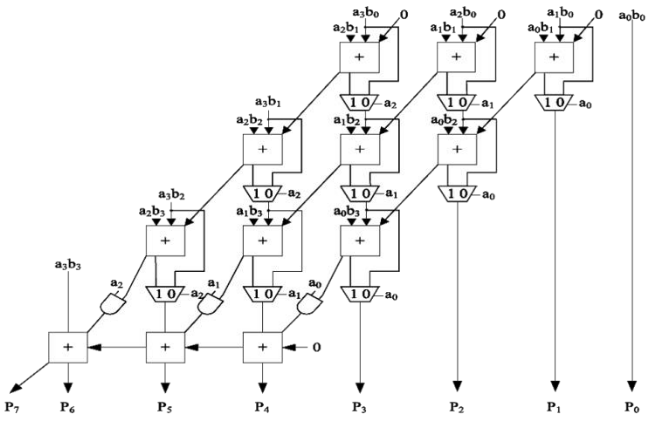

Fig.2Column Bypassing Multiplier

Figure 2 shows the 4×4 array structure of the proposed column-bypassing multiplier. In the bottom of the CSA array,we need to set the carry outputs to be 0. Otherwise, thecorresponding FA's may not produce the correct outputssince their inputs are disabled. This is done by adding anAND gate at the outputs of the last-row CSA adders.

The Row bypassing multiplier [6] reduces the switching activity by bypassing the row in which the multiplicand bit is zero. That means in the multiplier if a bit is zero then that row of adders will get disabled. For example, consider the multiplication of 1011 x 1010. Here the multiplier consists of zero in first and third positions. During multiplication the first and third row of adders get disabled and previous sum is taken as the present sum. In this adding cell the three state gate will enabled only when Xj =1 and then the adder will get input. If Xj =0 then the previous sum and carry only will be taken as the present sum and carry. Thus row bypassing can be done by this adding cell (AC).

In this way the switching activity can be reduced if the multiplicand bit is zero. Thus switching activity in row bypassing multiplier is less than that of Braun multiplier.

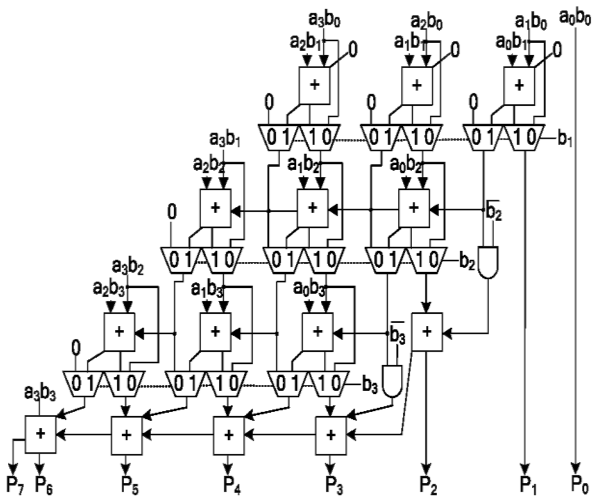

Fig.3 Row bypassing Multiplier

But the only disadvantage of this row bypassing multiplier is that it needs extra circuitry than Braun multiplier. This limitation can be overcome by the column bypass multiplier.

Razor Flip Flop

The key idea of Razor [7] is to purposely operate the circuit at sub-critical voltage and tune the operating voltage by monitoring the error rate. This eliminates the need for conservative voltage margins. In order to detect an error at the circuit level, each flip-flop is augmented by a shadow flip-flop, which is clocked by a delayed clock. If the combinational logic meets the setup time of the main flip-flop, then the main and delayed flip-flops will latch the same value. In this case, the error signal remains low. If the setup time of the main flip-flop is not met, then the main flip-flop will latch a value that is different from the shadow flip-flop.

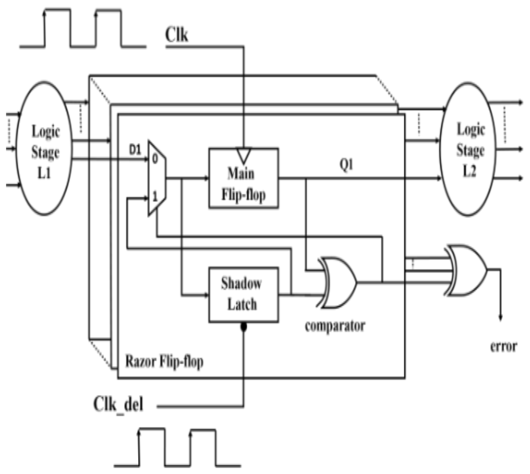

Fig.4 Razer flip-flop

To guarantee that the shadow flip-flop always latches the input data correctly, the input voltage is constrained such that under the worst-case condition, the logic delay does not exceed the shadow flip-flop's setup time. The circuit is shown in Figure 4.

Adaptive Hold Logic

The Adaptive Hold Logic (AHL) circuit is the main concept in this reliable multiplier. Fig 5 shows the components in the AHL circuit.The key components contained in the AHL circuit includes

a. Judging Blocks
b. D Flip-Flop
c. 2x1 Multiplexer
d. Aging Indicator

The Adaptive Hold Logic (AHL) circuit will decide whether the input patterns require one or two cycles to compute the operation with minimum performance degradation after considerable aging effect occurs.

Aging Indicator

The Aging Indicator indicates that whether the circuit has suffered any performance degradation due to aging effects. The aging indicator is implemented by using a counter that counts the total number of errors per certain amount of operations and it reset to zero at the end of those operations. The operations may be including multiplication or addition. The aging effect is not significant in the beginning, so the aging indicator produces output as 0. If errors happen frequently and exceed a predefined threshold, it means the circuit has suffered significant timing degradation due to the aging effect, and the aging indicator will output signal 1; otherwise, it will output 0 to indicate the aging effect is still not significant, and no actions are needed.

Judging Blocks

There are two judging blocks in Adaptive Hold Logic (AHL) circuits. The 1st judging block will generate an output as 1 if the number of zeros in the input sequence is larger than n. If the number of zeros in the input sequence is larger than n+1 then the output of the 2nd judging block is 1.but only one of them will be chosen at a time. The value of n is defined by the user [2], [3]. In the beginning, the aging indicator produces 0, so the first judging block is used. After a period of time when the aging effect becomes significant, the second judging block is chosen. Compared with the first judging block, the second judging block allows a smaller number of patterns to become one-cycle patterns because it requires more zeros in the multiplicand (multiplicator).

The operation of Adaptive Hold Logic (AHL) circuit are as follows: when an input sequence is given, both the judging blocks will decide whether the sequence requires one cycle or two cycle to complete their operation and pass both results to the multiplexer.

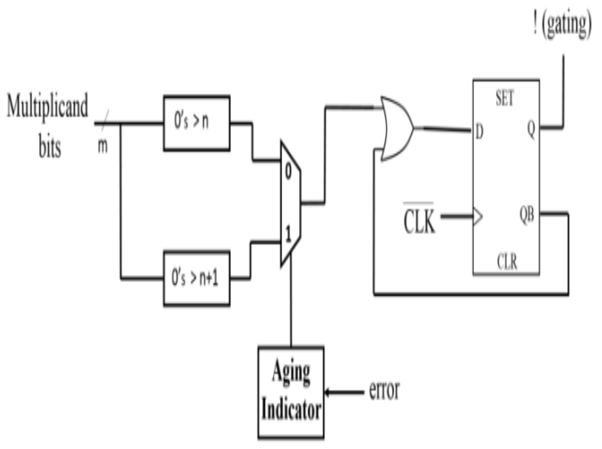

Fig.5 Adaptive Hold Logic

The multiplexer selects one of either result based on the output of the Aging Indicator. The result of the multiplexer and the output signal to the D flip-flop is ORed. This output of the OR operation is used to determine as the input of the D flip-flop. When the input sequence requires one cycle, the output of the multiplexer is 1. The! (gating) signal will become 1 and the input flip-flops will latch new data to perform operation in the next cycle. If the output of the multiplexer is 0 which means the input sequence requires two cycles to complete its operation. The OR gate will output 0 to the D flip-flop. Thus, to disabled the clock signal of the input flip-flops in the next cycle the !(gating) signal will become 0. Only one cycle of the input flip flop will be disabled because the D flip-flop will latch 1 in the next cycle.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

PROPOSED ARCHITECTURE

Fig. 6 shows our proposed aging-aware multiplier architecture [1], which includes two *m*-bit inputs (*m* is a positive number), one 2*m*-bit output, one column- or row-bypassing multiplier, 2*m* 1-bit Razor flip-flops, and an AHL circuit. The inputs of the row-bypassing multiplier are the symbols in the parentheses.

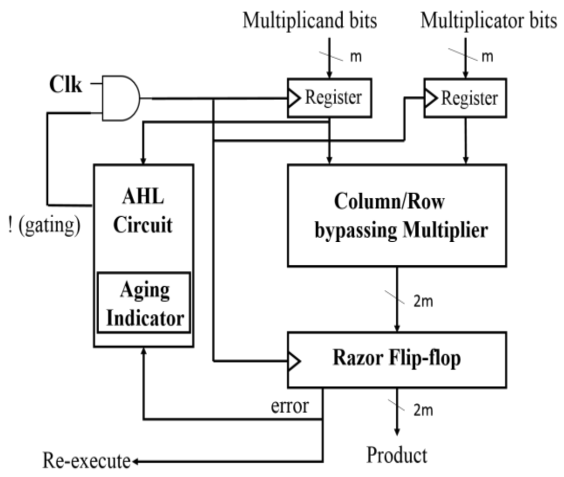

Fig. 6 Proposed Architecture

In the proposed architecture, the column- and row-bypassing multipliers can be examined by the number of zeros in either the multiplicand or multiplicator to predict whether the operation requires one cycle or two cycles to complete. Hence, the two aging-aware multipliers can be implemented using similar architecture, and the difference between the two bypassing multipliers lies in the input signals of the AHL. According to the bypassing selection in the column or row-bypassing multiplier, the input signal of the AHL in the architecture with the column-bypassing multiplier is the multiplicand, whereas that of the row-bypassing multiplier is the multiplicator. Razor flip-flops can be used to detect whether timing violations occur before the next input pattern arrives. The overall flow of our proposed architecture is as follows: when input patterns arrive, the column- or row-bypassing multiplier, and the AHL circuit execute simultaneously. According to the number of zeros in the multiplicand (multiplicator), the AHL circuit decides if the input patterns require one or two cycles. If the input pattern requires two cycles to complete, the AHL will output 0 to disable the clock signal of the flip-flops. Otherwise, the AHL will output 1 for normal operations. When the column- or row-bypassing multiplier finishes the operation, the result will be passed to the Razor flip-flops. The Razor flip-flops check whether there is the path delay timing violation. If timing violations occur, it means the cycle period is not long enough for the current operation to complete and that the execution result of the multiplier is incorrect. Thus, the Razor flip-flops will output an error to inform the system that the current operation needs to be reexecuted using two cycles to ensure the operation is correct. In this situation, the extra reexecution cycles caused by timing violation incurs a penalty to overall average latency. However, our proposed AHL circuit can accurately predict whether the input patterns require one or two cycles in most cases. Only a few input patterns may cause a timing variation when the AHL circuit judges incorrectly. In this case, the extra reexecution cycles did not produce significant timing degradation.Most tedious part in FFT is the complex multiplication. Therefore, we need correct solution for executing complex multiplications. Complex numbers are divided into two parts real and imaginary. Say $ar + jb$ is a complex number which is again multiplied by complex number cr +jd.

$$ar + jb \qquad (5)$$

$$cr + jd \qquad (6)$$

By multiplying these equations, we will get

$$(ac - bd) + j\,(bc + ad) \qquad (7)$$

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Another method for complex multiplication is shift and add for nontrivial twiddle factor multiplication. In radix-2 4 point FFT algorithm, the twiddle factor multiplication with W_4^2=-j and factors is trivial, multiplication with easily can be done by exchanging real to imaginary part and vice versa,by changing the sign of real and imaginary numbers.

4-POINT RADIX-2 DIT-FFT

The 4-point decimation-in-time (DIT) FFT algorithm computes the final output in two stages [8]. The four input time samples are first divided (or *decimated*) into two groups of 2-point DFTs. The two 2-point DFTs are then combined into 4-point DFT. This was the final output X(k). The detailed process is shown in Figure 7, where all the multiplications and additions are shown. Note that the basic two-point DFT butterfly operation forms the basis for all computation. The computation is done in two stages. After the first stage computation is complete, there is no need to store any previous results. The first stage outputs can be stored in the same registers which originally held the time samples x(n). Similarly, when the second stage computation is completed, the results of the first stage computation can be deleted. In this way, in-placecomputation proceeds to the final stage.

The Flow graph for DIT- FFT decomposition for 4 point is shown in the below figure

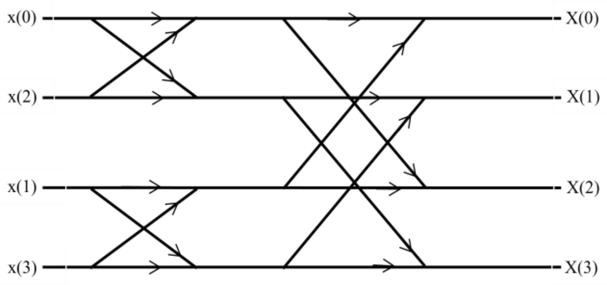

Fig 7: 4 point FFT Structure

The simulation output for the FFT is shown below

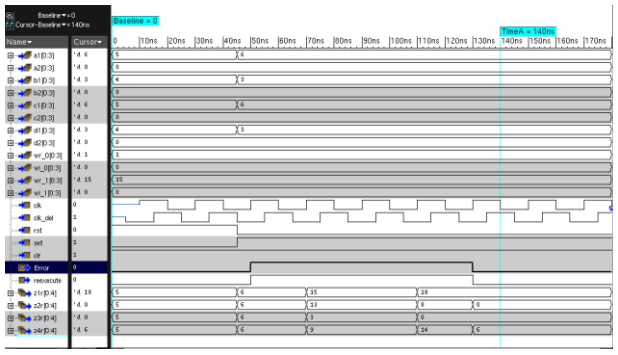

Fig 8: Simulation result of 4 point FFT

ASIC SYNTHESIS RESULTS AND LAYOUT

We have designed4 x 4, 16× 16, 32×32 and 64x64 for Array Multiplier, Fixed Latency Column bypassing Multiplier, Fixed Latency Row bypassing Multiplier, Variable latency Column bypassing Multiplier and Variable Latency Column bypassing Multiplier.

A 4 point FFT was also implemented by using these multipliers.All the designs are synthesized in the Cadence RTL Compiler (RTL) using 180-nm CMOS library. The netlist file was extracted from RTL Compiler.

Area Comparison

Fig. 9 compares the normalized area of the AM, FLCB,A-VLCB, FLRB, and A-VLRB in 4x4, 16× 16, 32×32 and 64x64 multipliers. The data are normalized to the area of the AM. Inthe 16×16 multiplier, the area of the A-VLCB and A-VLRB is26% and 22.9% higher than FLCB and FLRB. In the 32×32multiplier, the area of the A-VLCB and A-VLRB is 14.5% and12.8% higher than that of the FLCB and FLRB.In the 64 × 64 multiplier, the area of the A-VLCB and A-VLRB is 0.08/% and 0.06% higher than FLCB and FLRB, respectively. This is because when a fixed-latency bypassing multiplier ischanged to a variable-latencybypassing multiplier, additionalcircuits are needed for AHL and Razor flip-flops to ensure thecorrect operations of the multiplier after degradation. Note thatthe increased area overhead ratio of the 64 × 64 A-VLCB andA-VLRB is much smaller than that of the 32×32 A-VLCB andA-VLRB and is smaller than that of the 16×16 A-VLCBand A-VLRB. This is becauseAHL and Razor flip-flops bothoccupy a smaller area ratio in larger multipliers.

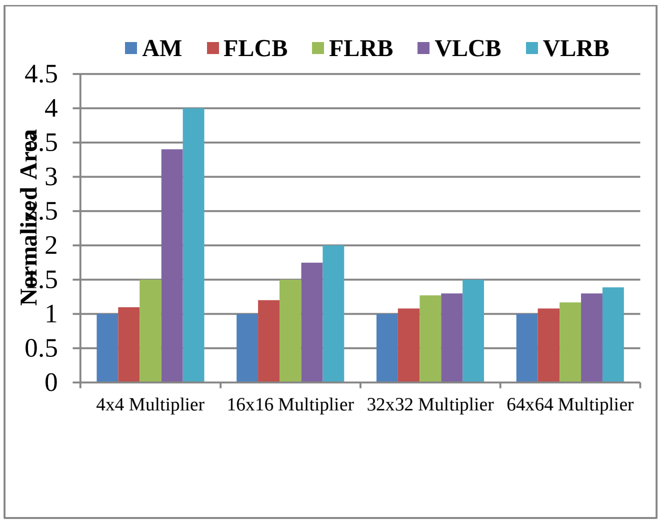

Fig 9 Normalized Area Comparison

Power Comparison

This is because the fixed- latency multiplier uses the bypassing techniques to reduce power consumption. Compared with the fixed-latency multiplier, the variable-latency multiplier has higher power due to more complicated circuits. However, the variable-latency multiplier still has less power than that of the AM because it uses both the clocking gating and a bypassing power reduction technique. Moreover, the power of the 16 × 16 A-VLRB is larger than that of the 16 × 16 A-VLCB. This is because the row-bypassing multiplier is more complicated than the column-bypassing multiplier and because the area overhead of the row-bypassing multipliers is larger than that of the column-bypassing multipliers, which results in more power consumption.

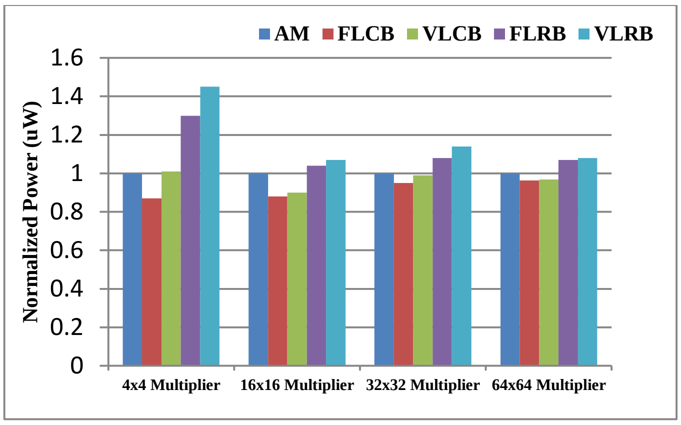

Fig 10: Normalized Power Comparison

Layout

The below figure shows the layout of 64x64 Variable Latency Column bypassing Multiplier. The layout was generated by using Cadence Encounter Tool.

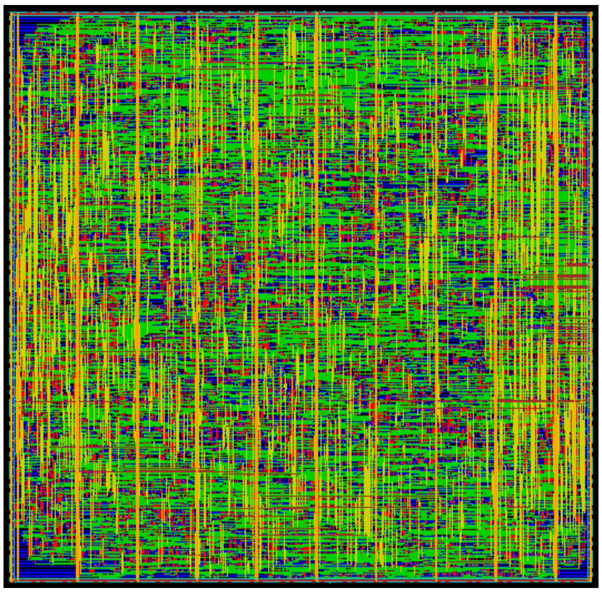

Fig 11: Layout for 64 x 64 Variable Latency Column bypassing Multiplier

CONCLUSION

In this paper, a 4-point DIT-FFT processor is implemented using radix-2. This helps in reducing the complex multiplications. This paper also describes how to avoid floating point arithmetic for implementation of FFT.The complex multiplication is implemented by using a reliable multiplier with AHL Circuit. In future, the work can be extended to the N bit variable input signals. The implemented design can be used as a basic block for further computation. The pipelined architecture can also be added to FFT for providing fast and better performance. The proposed processor can be integrated with other components which can be used as a stand-alone processor for many applications.

REFERENCES

1. Ing-Chao Lin, Yu-Hung Cho , Yi-Ming Yang "Aging-Aware Reliable Multiplier Design With Adaptive Hold Logic" in IEEE Transactions on Very Large Scale Integration (VLSI) SYSTEMS 1063-8210 2014

2. K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," inProc. DATE, 2012,pp. 1257–1262.

3. A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," inProc. DATE,2008, pp. 1250–1255.

4. Y.-S. Su, D.-C. Wang, S.-C. Chang, and M. Marek-Sadowska, "Performance optimization using variable-latency design style," IEEETrans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 10,pp. 1874–1883, Oct. 2011.

5. M.-C. Wen, S.-J. Wang, and Y.-N. Lin, "Low power parallel multiplier with column bypassing," in *Proc. IEEE ISCAS*, May 2005, pp. 1638–1641.

6. J. Ohban, V. G. Moshnyaga, and K. Inoue, "Multiplier energy reduction through bypassing of partial products," inProc. APCCAS, 2002,pp. 13–17.

7. D. Ernstet al., "Razor: A low-power pipeline based on circuit-leveltiming speculation," inProc. 36th Annu.IEEE/ACM MICRO, Dec. 2003,pp. 7–18.

8. Neha V. Mahajan, Dr. J. S. Chitode "Simple Computation of DIT FFT" in ijarcsse, Volume 4, Issue 5, May 2014