Dadda Multiplier

12x12 bit Multiplier Implemented using Dadda architecture

Athmaram Swaminathan

School of Electrical and Computer Engineering

Purdue University

West Lafayette, Indiana, 47906

Email: aswamina@purdue.edu

Vaibhav Ramachandran

School of Electrical and Computer Engineering

Purdue University

West Lafayette, Indiana, 47906

Email: ramachav@purdue.edu

***Abstract*—**The 12x12 bit multiplier was implemented using the Dadda architecture and 0.2μm technology. It consisted of 5 stages of reduction of the 144 product terms, using 120 full adders and 12 half adders total. The full and half adders were implemented using Manchester carry generation logic to improve speed. A combination of pass transistor logic and CMOS logic was used to create the individual gates to minimize transistors and improve performance. The number of transistors thus used in this architecture was 3936. Through Cadence simulation the average power consumed by this architecture was found to be 788.2635μW when clocked at 33MHz. Through simulations in Cadence, the operation time was estimated to be approximately 15ns per multiplication. If the clock signal is faster, the results become undetermined with floating voltages and errors appear. Through theoretical analysis of the propagation delays of full adders and half adders, the estimated operation time was found to be 12.376ns.

# Introduction

This project requires implementing a 12x12 bit digital multiplier to be used in the ALU of a high performance future generation RISC microprocessor chip.

An efficient multiplier should have the following characteristics:

1. Accuracy – A good multiplier should give correct result.
2. Speed – Multiplier should perform operation in high speed.
3. Area – a multiplier should occupy less number of slices and LUTs.
4. Power - Multiplier should consume less power

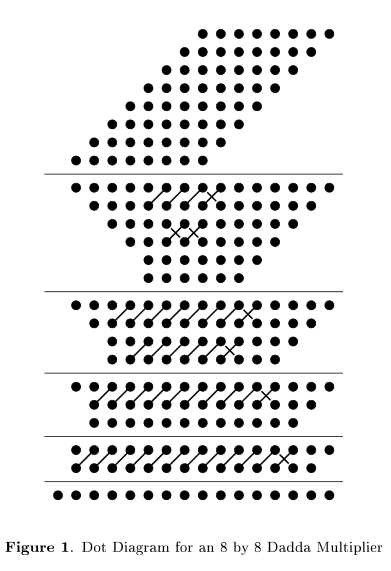
Several types of multipliers are available which are implemented in different ways. Some of these multipliers include Dadda, Wallace-Tree, Booth, Sequential, Array etc. A brief background to the above listed multipliers as well as a comparison between them is shown below.

## Dadda Multiplier

A Dadda multiplier is a hardware multiplier design invented by computer scientist Luigi Dadda. The Dadda multiplier has three steps for two bit strings w1 and w2 of lengths l1 and l2 respectively:

1. Multiply (logical AND) each bit of w1, by each bit of w2, giving l1\*l2 results, that are grouped by weight in columns.
2. Reduce the number of partial products by stages of full adders and half adders by (3/2) at each stage until we are left with at most two bits of each weight.
3. Add the final result with the conventional adder.

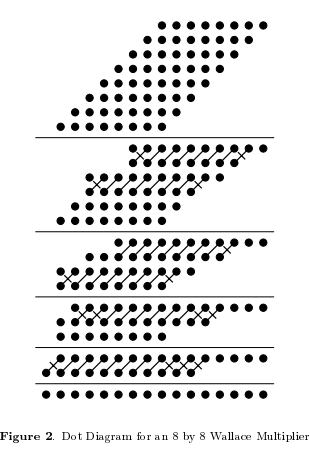
The dot diagram for an 8x8 bit Dadda multiplier is shown below:



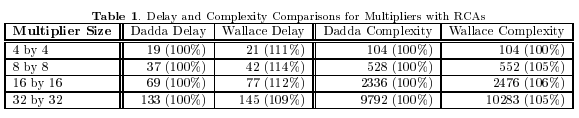
## Wallace-Tree Multiplier

A Wallace multiplier works similarly to the Dadda multiplier and follows the same 3 steps to perform the calculation. Although Wallace and Dadda multipliers contain nearly identical number of full adders, more of the Wallace full adders are applied during the reduction of the matrix. This and the additional half adders used in the Wallace reduction results in the shorter final carry propagating adder.

The dot diagram for an 8x8 bit Wallace multiplier is shown below:



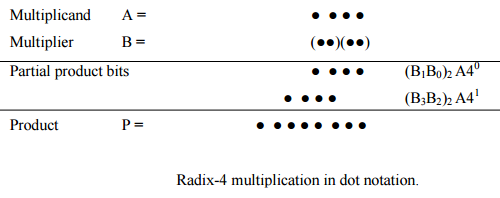
Comparing Dadda and Wallace multiplier:



It is clear from the table that Dadda is both faster and smaller than the corresponding Wallace multiplier.

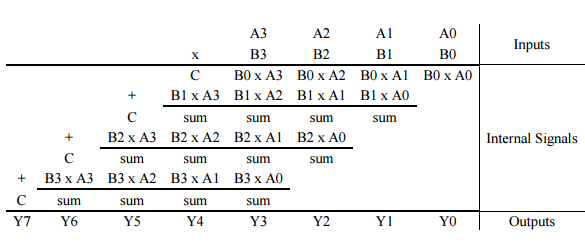
*C. Booth multiplier*

Booth multiplication algorithm gives a procedure for multiplying binary integers in signed -2’s complement representation, including an implicit bit below the least significant bit, y-1=0. For each bit yi, for i running from 0 to N-1, the bits yi and yi-1 are considered. When these two bits are equal, the product accumulator P is left unchanged. Where yi=0 and yi-1=1, the multiplicand times 2i is the added to P, and when yi=1 and yi-1=0, the multiplicand time 2i is subtracted from P. The final value of P is the signed product.



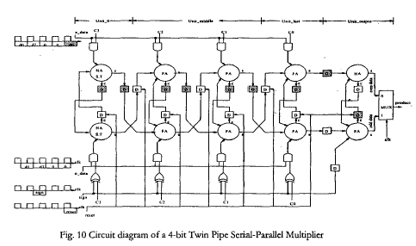
*D. Array Multiplier*

The array multiplier is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial products are shifted according to their bit orders and then added. The addition can be performed with normal carry propagate adder. N-1 adders are required where N is the multiplier length.

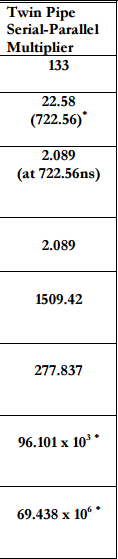
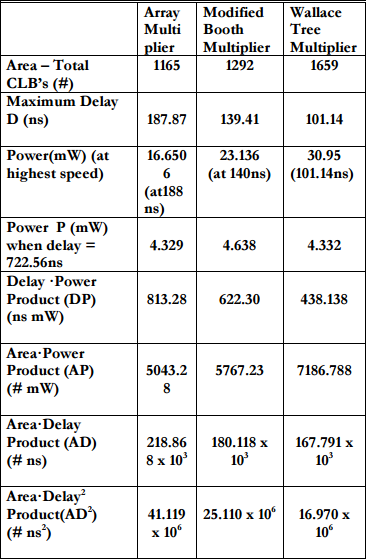


*E . Twin-Piped Serial-Parallel Multiplier*

The odd-indexed data bits and even-indexed data bits are processed in different clock phase and different circuits. It results in double throughput. The multiplier consists of 4 units. The multiplicand is fed in parallel and the multiplier is fed in serial. The product is shifted out in series. These are used where area and power consumption is restricted and speed is not important.



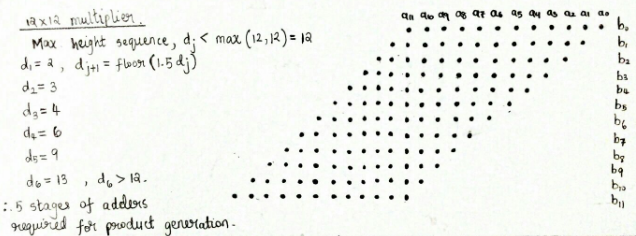
Comparing these multipliers:

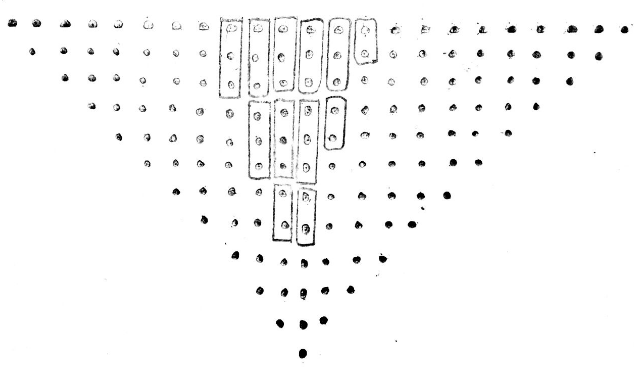
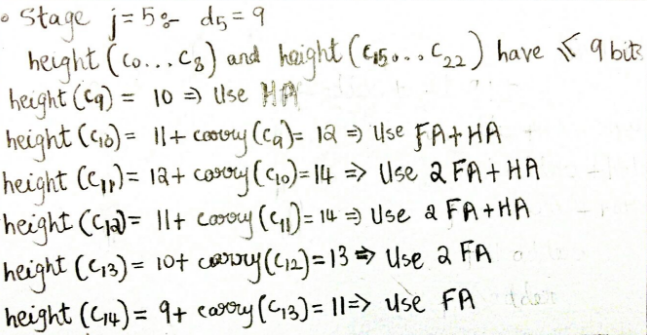


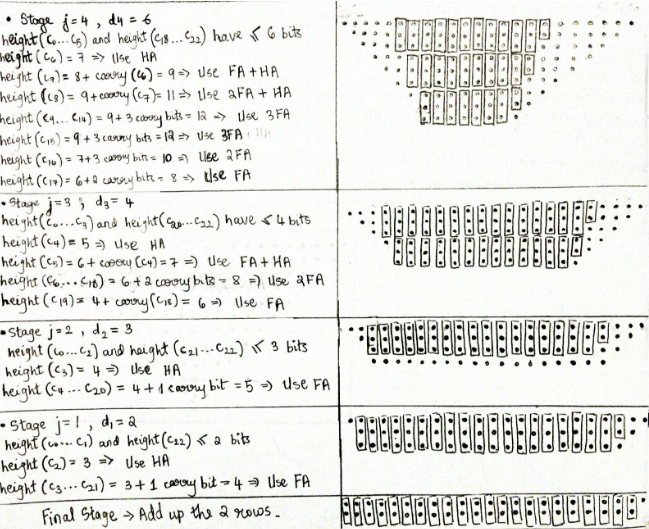
# Proposed Technique

Our proposed technique was to implement a 12x12 bit Dadda multiplier in order to satisfy the high-performance requirements requested by Letni Inc.

* Our first step was to design the 12x12 bit Dadda multiplier on paper as shown below for easy replication on CADENCE:

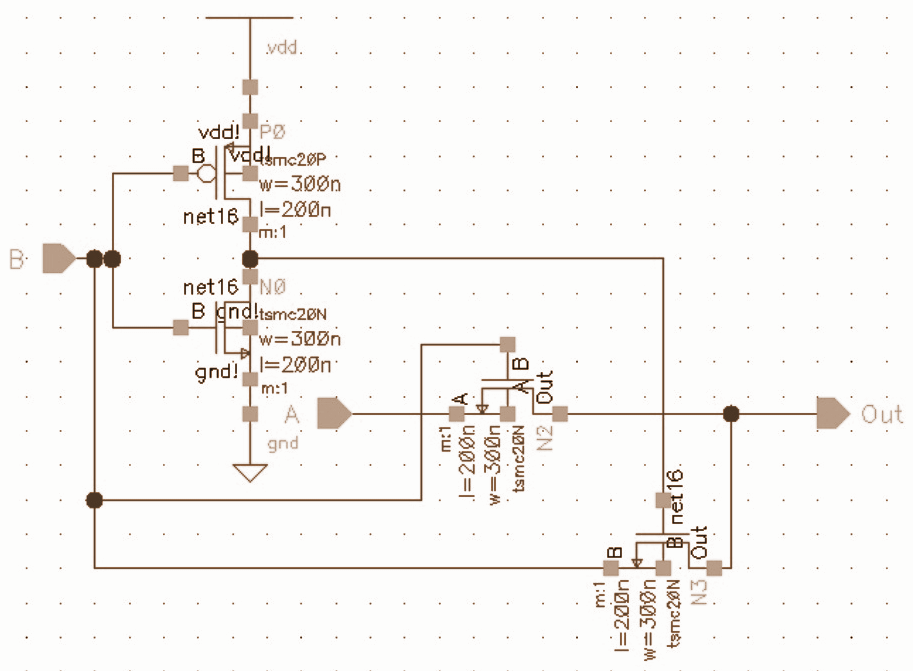




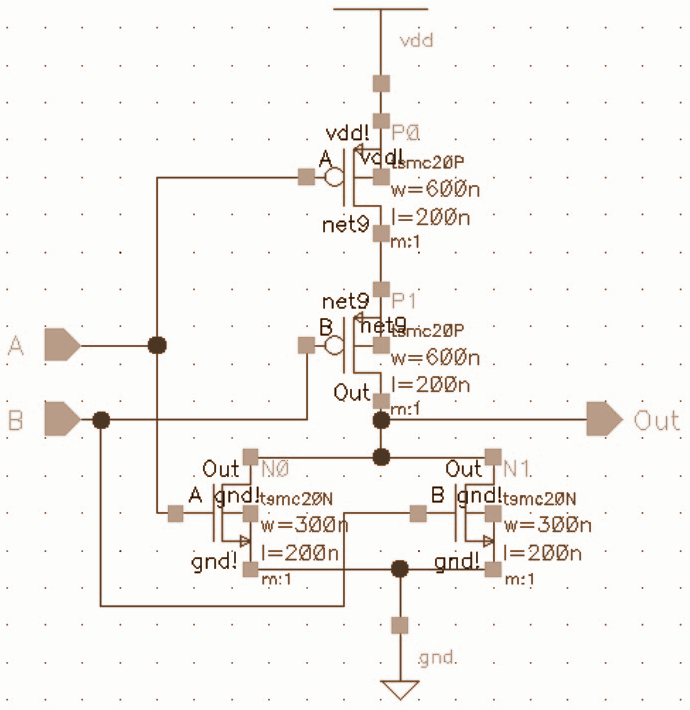


* We then mapped out the individual product terms.
* We started CADENCE by first designing the small sub-blocks which build up to the final circuit based on particular logic in order to minimize the number of transistors used and also to reduce delay.
* We decided to build the manchester-carry adder in order to implement the Full and Half adders required for the Dadda multiplier because it had a much shorter propagation delay compared to the ripple carry adder.
* The Full and Half adders have been further divided into the following:

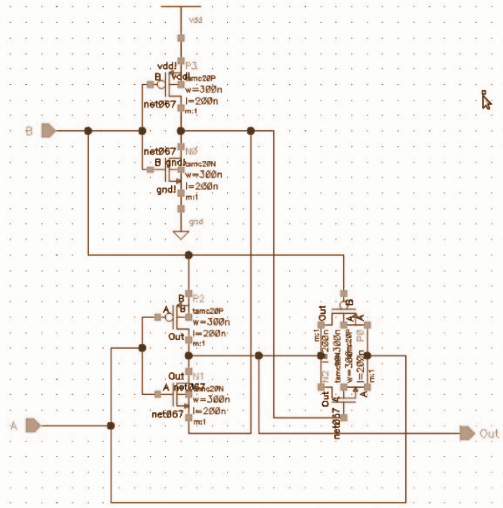
1. AND gate using pass transistor logic (Generate Function) (Uses minimum transistors)



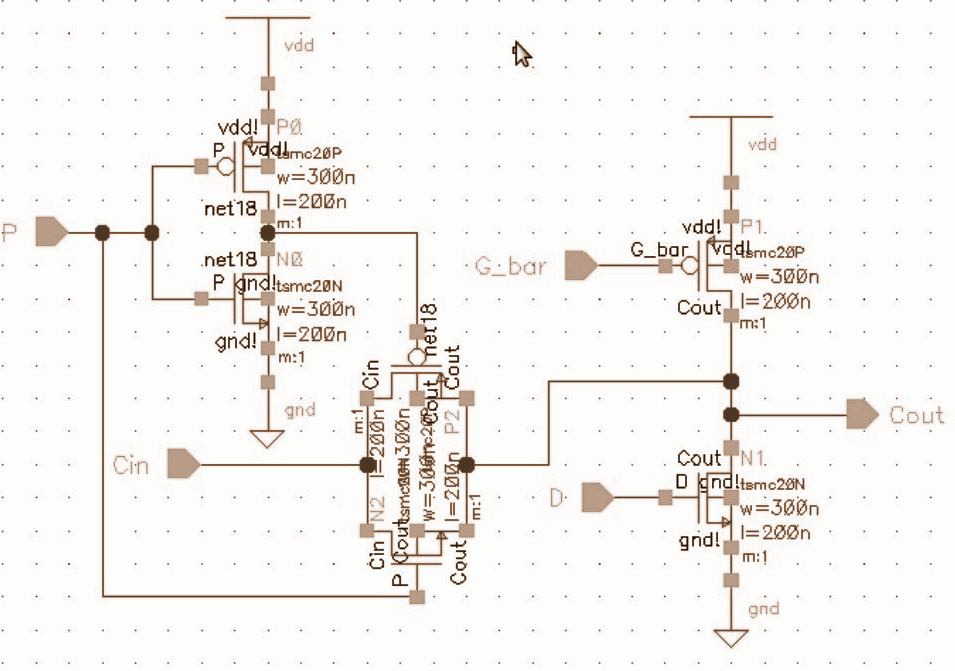
1. NOR gate using complementary CMOS logic (Delete Function) (Uses minimum transistors)



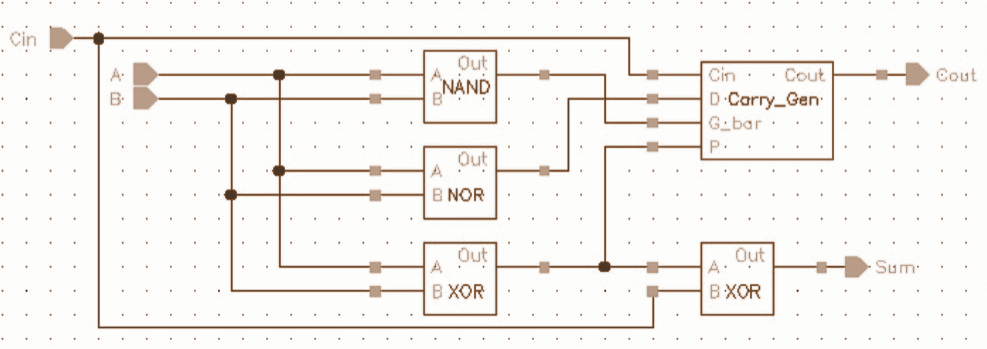
1. XOR gate with Transmission gate logic (Propagate Function) (Uses minimum transistors, min. delay)



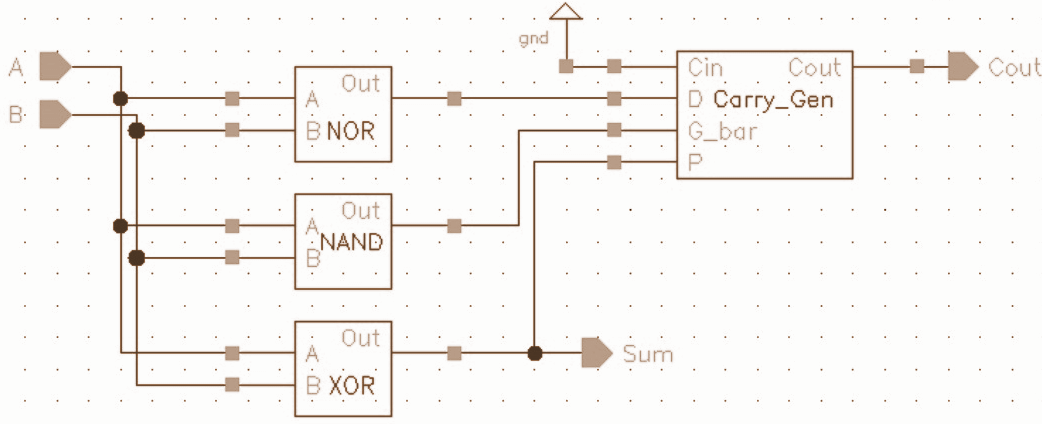
1. Manchester Carry using Transmission gate logic



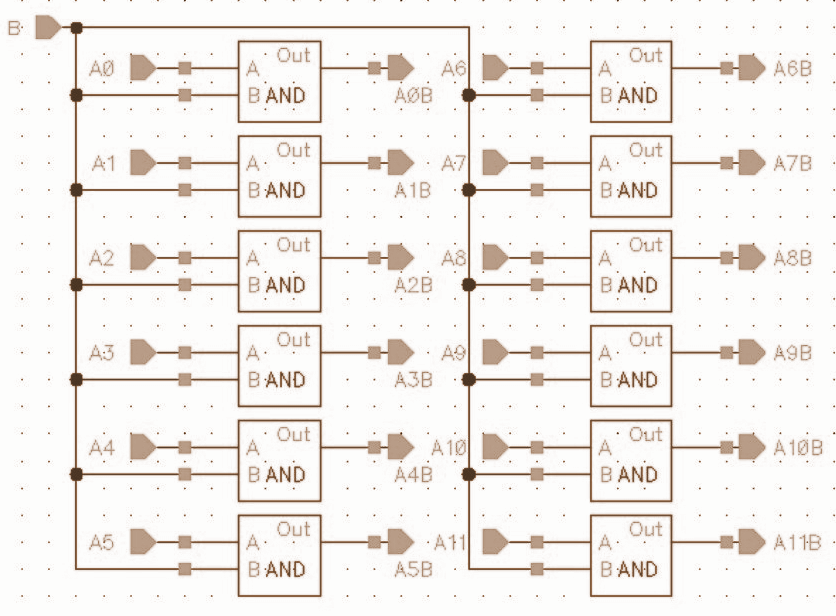
* Full Adder Circuit making use of a NAND gate instead of an AND gate and an inverter, to reduce transmission time. (Since G\_bar is used in the Manchester adder circuit)



* Half Adder Circuit

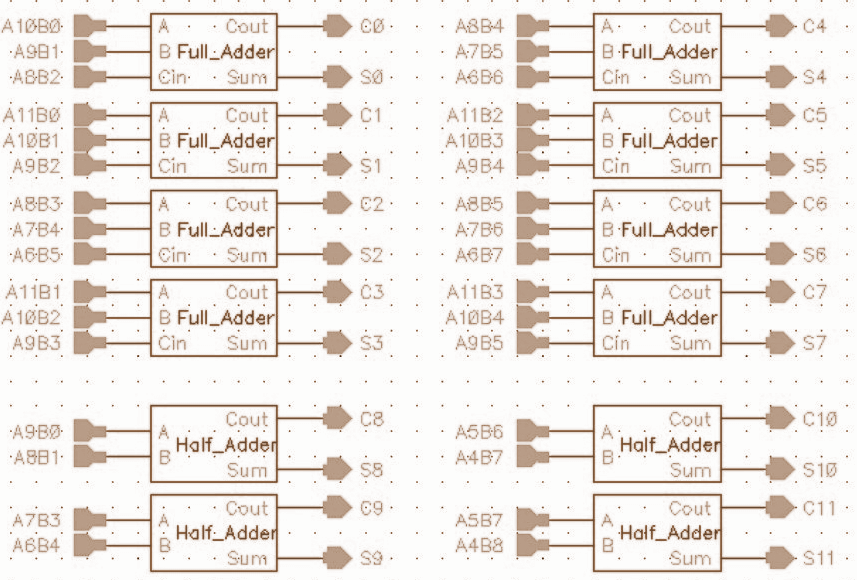


* In order to produce the product terms, we compacted multiple AND gates into a single circuit to make the final circuit more presentable. The circuit shown below shows 12 AND gates compacted into one circuit which was then replicated 12 times in order to produce 144 product terms.

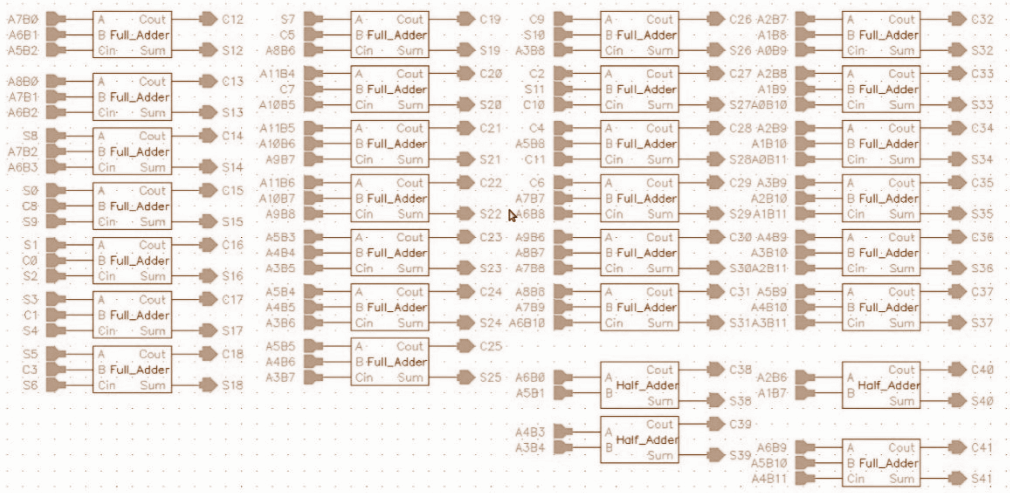


* Based on our analytical calculations on page 3, 5 stages of adders were needed to be designed in order to build our 12x12 bit Dadda multiplier. These stages have been implemented as shown below. Details about the procedure to calculate the number of adders have been provided in the analytical calculations.

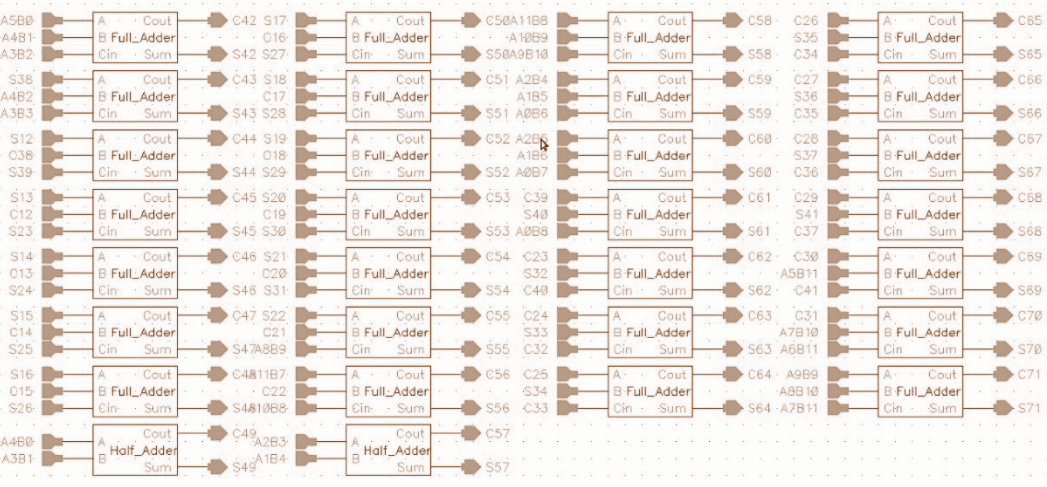
1. Stage 1 Reduction



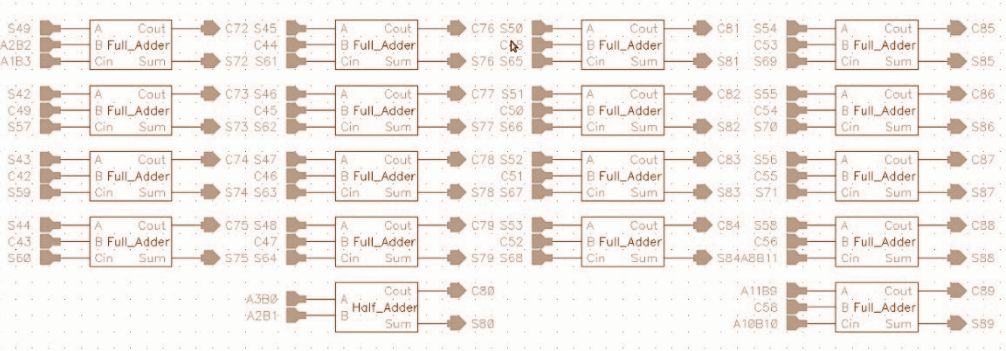
1. Stage 2 Reduction



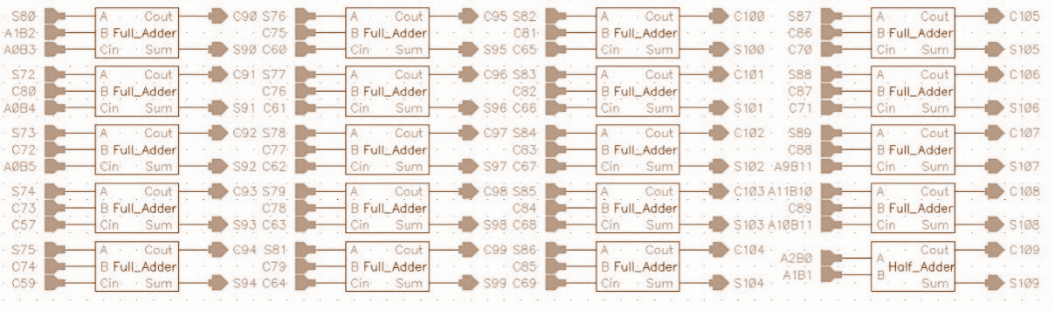
1. Stage 3 Reduction



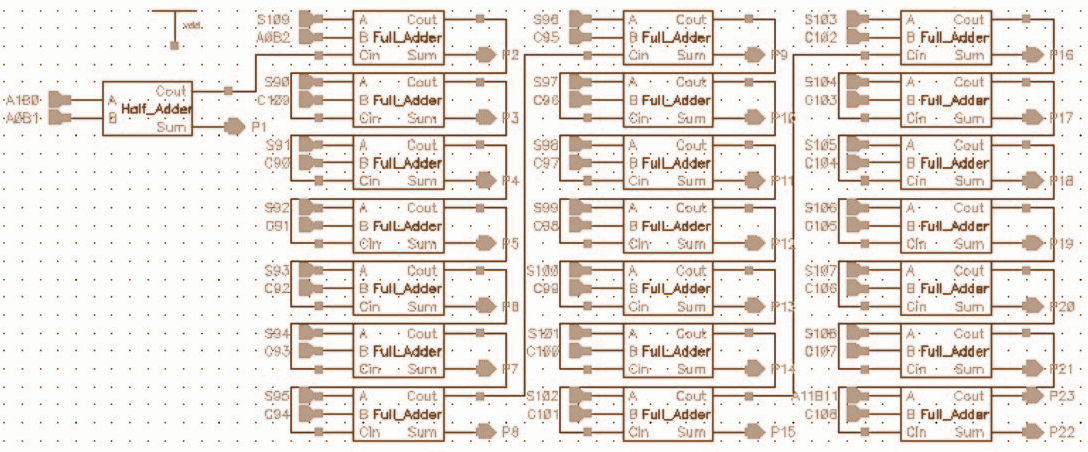
1. Stage 4 Reduction



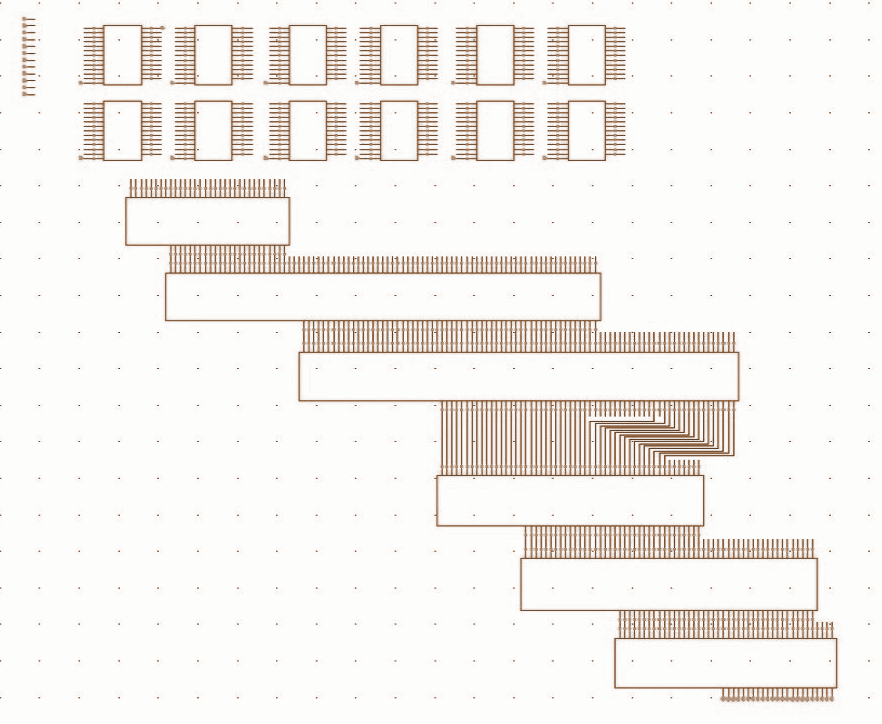
1. Stage 5 Reduction



1. Final Adder



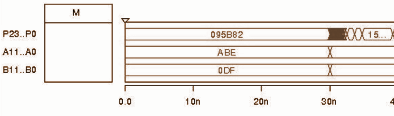
* Putting together all the sub-blocks, our final 12x12 bit Dadda multiplier was created as shown below.

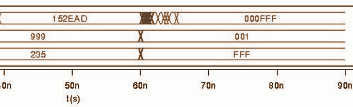


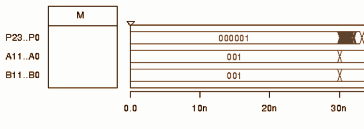
# 

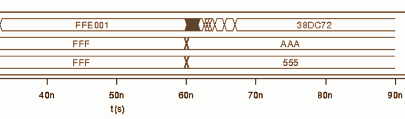
# Proving Functionality

A random set of inputs were written into the .vec file and the nanosim command was run. The multiplier was clocked at a frequency of 33.33MHz. The .cfg file was modified to plot out the input logic levels. The input voltage was also plotted for debugging purposes. The command used was “nanosim -nspice Project\_2\_Netlist -nvec Project\_2.vec -c Project\_2.cfg -out fsdb -z tt -t 90”. As can be seen below, the outputs are exactly as expected showing that the circuit works as required.









*Nanosim Results*

Number of transistors used: 3936

Average Power Dissipated: 788.2635μW

Estimated Operation Time: 12.376ns

Simulated Operation Time : 15ns

*Our Design vs other Multipliers*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Dadda Multiplier | Array Multiplier | Booth Multiplier | Wallace Multiplier | Twin-Pipe Serial-Parallel Multiplier |
| Max Delay | 15ns | 187.87ns | 139.41ns | 101.14ns | 22.58ns |
| Average Power | 788.2635μW | 16.65mW | 23.136mW | 30.95mW | 2.089mW |

##### References

1. Dadda multiplier. (2017, April 17). Retrieved April 26, 2017, from <https://en.wikipedia.org/wiki/Dadda_multiplier>
2. (n.d.). Wikipedia, the free encyclopedia. *Dadda multiplier - Wikipedia.* Retrieved from <http://en.wikipedia.org/wiki/Dadda_multiplier>
3. Bickerstaff, K. C., Schulte, M., & Swartzlander, E. E. (n.d.). Reduced Area Multipliers. Retrieved April 26, 2017, from <http://ieeexplore.ieee.org/document/397168/>
4. file:///U:/Personal/Downloads/Multipliers.pdf [PDF file]
5. file:///U:/Personal/Downloads/STUDY\_ON\_COMPARISON\_OF\_VARIOUS\_MULTIPLIE.pdf [PDF file]
6. <http://www.cerc.utexas.edu/~whitney/pdfs/spie03.pdf> [PDF file]
7. <http://www.ijettcs.org/Volume2Issue4/IJETTCS-2013-08-25-115.pdf> [PDF file