Design of an Optimized 8×8 Dadda Multiplier Using 4:2 Compressors

Amit Vardhan Suryadevara
Department of Electrical Engineering
Purdue Univeristy
Indianapolis, United States
avsuryad@purdue.edu

***Abstract* - In this paper the design of an 8×8 Dadda multiplier implemented using the 45nm CMOS technology optimized for high speed and low-area applications. The multiplier design used 4:2 compressors in the initial reduction stages to efficiently handle partial products, minimizing delay. The final stage employs a series of FA (Full Adders) and HA (Half Adders) to compute the sum of the last two rows, further improving performance. Transistor level design and simulations validate the architecture, demonstrating an average power dissipation of 12uW and at 500GHz. Analytical calculations and cadence simulations confirm the functionality, reliability, readiness of the multiplier for tape-out in Letni's next generation chip.**

***Keywords* –Dadda Multiplier, Compressor, Power Consumption.**

I.INTRODCUTION

Multipliers are crucial parts of contemporary digital systems, utilized extensively in image processing, digital signal processing (DSP), cryptographic methods, and arithmetic operations. Designing multipliers is frequently a difficult task, particularly when considering area usage, power efficiency, and speed. In applications where performance optimization is essential, such as microprocessors and field-programmable gate arrays (FPGAs), multipliers are especially important. Multipliers work by generating partial products and summing them to compute the result. An array multiplier, Wallace tree multiplier Booth multiplier, and Dadda multiplier are some of the important designs that have been created over the years to optimize this process. This report focuses on the design of an 8×8 Dadda multiplier, a hardware-efficient architecture known for its efficient and accurate balance between speed and area.

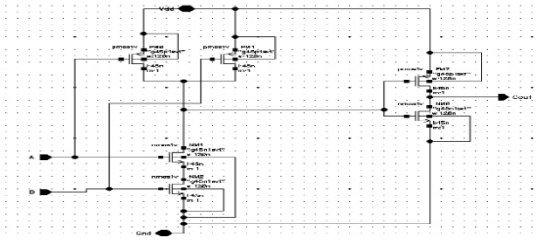
Due to their importance in processors, digital signal processing (DSP) units, and embedded systems, multipliers of modest size, such 8x8 multipliers, are extensively researched. 8*8 Wallace Tree multipliers actively reduce partial products at every level and a quick adder is used to sum the two rows that remain after each stage of reducing three partial product rows to two rows, hence why they are known for their high-speed performance, whereas the 8*8 Dadda Multiplier has specified height limits, the Dadda multiplier takes an organized and careful approach, reducing partial products only when required. As a result, fewer adders are needed, which lowers power and space usage. According to studies [3], the Dadda multiplier outperforms the Wallace multiplier in terms of area and power efficiency for 8x8 multiplication while retaining competitive speed. This makes it especially appropriate for low-power applications and embedded CPUs, where resource efficiency is essential. The challenges of dealing with products improve with increasing bit-width, as in 16x16 multipliers. The increased complexity also translates into higher power consumption and design challenges. The Dadda multiplier is more area-efficient and power-efficient than the Wallace multiplier for 16x16 designs, achieving a 50-70% reduction in area and power overhead by using fewer adders and stages, while sacrificing only marginal speed, making it ideal for applications requiring a balance of performance and resource efficiency [2]. For 4x4 multipliers, simpler designs like array and Booth multipliers are commonly used in systems where power and size are less critical. Array multipliers, with their straightforward grid-like structure, are easier to implement but are larger and slower compared to more optimized designs like Wallace or Dadda multipliers. Despite these drawbacks, their reduced delay and area impact in 4x4 configurations make them suitable for low-performance applications. Booth multipliers, on the other hand, are well-suited for signed multiplications, offering moderate speed and area efficiency through Booth encoding, and are widely used in applications like signal processing that require signed operations.

II. PROPOSED TECHNIQUE

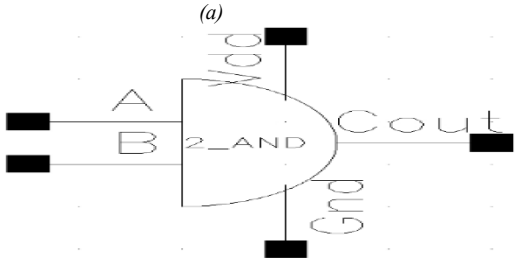
This section is about the architecture proposed in the 8*8 Dadda multiplier. To generate 16-bit output we need to construct

- The Circuit is constructed using static CMOS logic, the Dadda architecture, combined with CMOS's fast switching capabilities, ensures high-speed multiplication. The two 8-bit operands A and B, a total of 64 partial products is generated, with each partial product created by multiplying one bit of A with one bit of B using an AND operation. These partial products are arranged in an 8×8 matrix, forming the basis for further reduction and addition in the multiplication process.
- There are three reduction stages to compress to the final stage and the reduction phase uses 4:2 compressors, full adders (FAs), and half adders (HAs) to minimize the number of rows step by step.
- A 4:2 compressor efficiently combines four input bits and an input carry (Cin) into two outputs (Sum and Carry) and output Cout, reducing critical path delay and minimizing the need for additional adders, making it ideal for high-speed operations in large matrices. While Half Adders (HA) are simple and effective for two-bit additions, they are less

suited for complex reductions. Full Adders (FA), on the other hand, handle three-bit additions with greater versatility, making them a staple in advanced addition circuits. The provided block diagrams are the required components to build the 8*8 Dadda Multiplier.

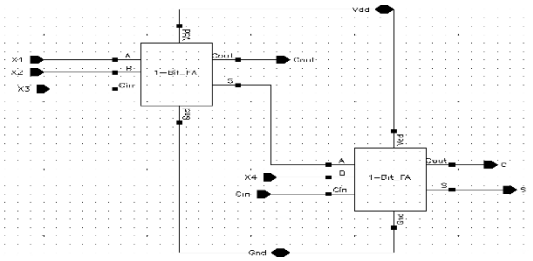


(a)

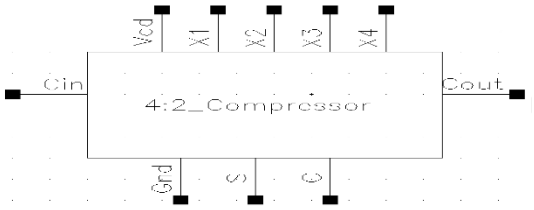


(b)

Fig 1. 2-Bit AND gate (a) Schematic (b) Symbol



(a)



(b)

Fig 2. 4:2 Compressor (a) Schematic (b) Symbol

*Fig 3. 8*8 Dadda Multiplier Architecture*

Fig 3. illustrates the working principle of the Dadda Multipliers.

- Stage 1 uses the reduction elements (Compressors, FA, HA) when the partial products in a column are greater than 4.
- Half Adder generates the carry(C) to the compressor in the next column as Cin, in the same stage and generates the sum(S) to the same column in stage 2 as the partial product.
- The Compressor receives Cin from HA in the previous column, it generates Carry(C) as the partial product to the next stage (Stage 2) and Cout to the Compressor in next column as Cin and the sum is generated in the same column. This process is iterated throughout the last column of stage 1 and a total of 8 compressors; 2 HA and 2 FA are implemented in stage 1.
- The same method is implemented in stage 2 when the partial products in each column are greater than 2, where the Sum(S), Carry(C) is generated to stage 3 and a total of 10 Compressors, 1 FA, 1HA are implemented in stage 1.
- Stage 3 generates to the final stage with same method, if the partial products in each column are greater than 1 and a total of 13 FA and 2 HA are implemented in this stage. The final schematic contains 18 Compressors, 15 FA, 5 HA which gives a total of 2874 transistors.

With fewer components and efficient stages, architecture offers fast speed and lower power consumption, making it scalable for higher bit-width multipliers. Its implementation and reduction stages require careful planning, since its design is more complex than that of simpler alternatives such as array multipliers.

III. SIMULATION RESULT

The final schematic is generated with Compressors, Half Adders and Full Adders in Cadence virtuoso (Fig 4). The result is simulated with transient analysis of 100ns. The series of inputs along with the time, rise and fall time are given in the vector file and the waveform is generated.

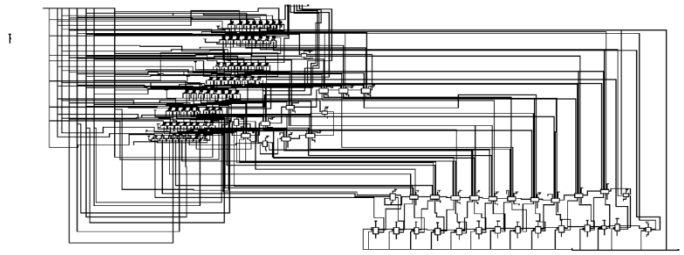


*Fig 4. Schematic of 8*8 Dadda Multiplier*

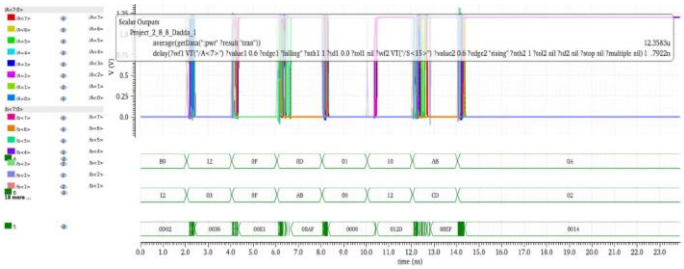


*Fig 5. Waveform of 8*8 Dadda Multiplier*

Fig 5. illustrates the waveform of multiplier with the period of 2ns which is 500 MHz and the 8-input series, the output obtained is efficient but the transient disturbance in the waveform is due to the

delay which is 1.7ns which is negligible, this disturbance increases when the delay increases to further. The power dissipation is 12.35uW, since fewer adders are used, and the overall structure is simpler, there is typically less switching activity, resulting in lower dynamic power consumption. When the number of inputs increases with frequency, the power dissipation goes high and as it consumes more power from inputs. In Fig 6 the output obtained is efficient and the power dissipation is 16.46uW as the number of inputs increases to 11 and the delay also increases to 4.033ns with increased inputs.

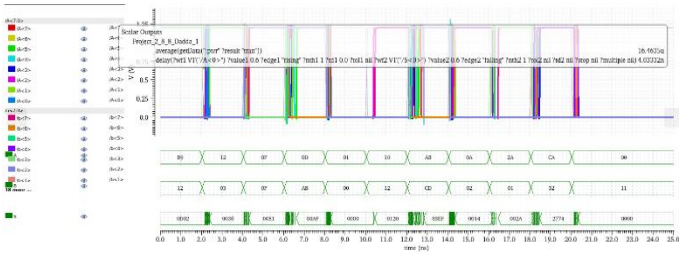


*Fig 6. 8*8 Dadda Multiplier with 11 inputs*

Fig 7. shows the worst-case period of 0.5 ns which is 2GHz, the waveform will have more disturbances with the inefficient output. As the period decreases further the output will be less efficient. This is because the maximum frequency at which the output will be generated is 2 GHz and after this frequency there will be a transient disturbance along with the inefficient output.

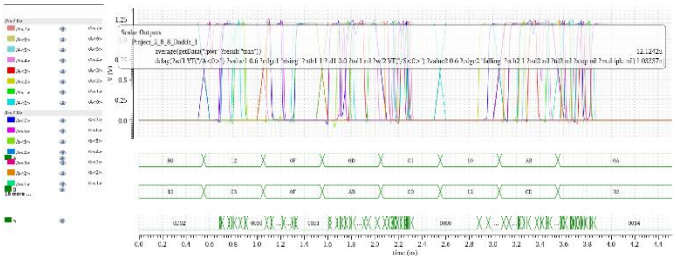


Fig 7. Worst Case_1- delay of a 11-bit

The second worst case in Fig 8. is the inefficiency of the cadence virtuoso in not differentiating the monotonous inputs as it generates all the inputs in the same period of 2ns which is making the power and delay higher. This is because the static power doesn't have the clock frequency to differentiate the monotonous inputs, and it generates the output with the same period. This is one of the drawbacks of this architecture using static power design. The clock frequency helps to differentiate the inputs as it represents the number of clock cycles per second. The higher the clock frequency, the faster the circuit performs computations. Higher clock frequencies increase dynamic power consumption because power depends on the frequency ($P_{Dynamic} \propto f$). This can lead to greater heat generation and potentially reduced efficiency.

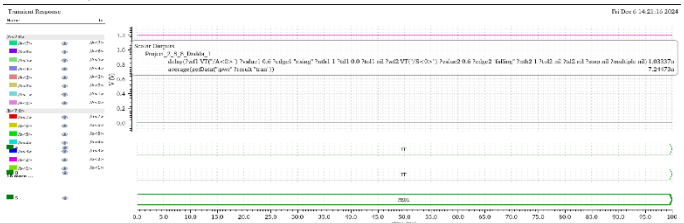


Fig 8. Worst case2- Monotonous Inputs

The theoretical calculation of power gives a value of 17.24µW and the propagation delay of 1.20ns, assuming activation factor and Current leakage for the frequency of 500MHz. However, the Cadence simulations revealed significantly optimized results, with a power dissipation of only 12 µW and a delay of 1.7 ns, demonstrating the efficiency of the transistor-level implementation. The simulation results show a significant latency reduction due to efficient layout, optimized connections, and the effective use of 4:2 compressors, half adders, and full adders. The lower power dissipation in the simulated circuit can be credited to reduced switching activity and improved sizing.

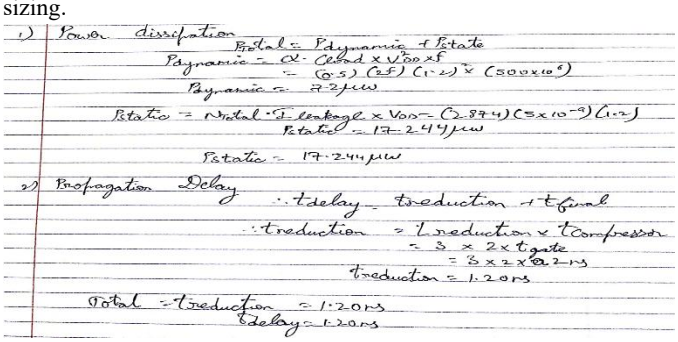


Fig 9. Theoretical Calculation

Multiplier Type	Technology type	Number of bits	Power consumption (µW)	Remarks
Array Multiplier	FinFET 10nm	8*8	120	Reduced power and area compared to CMOS;
Dadda Multiplier	CMOS 45nm	8*8	50	Optimized for CMOS implementation with reduced transistor count and critical path delay.
Wallace Multiplier	SOI 22nm	8*8	90	SOI technology lowers power.

Table 1. Comparison Table

V. REFERENCES

[1] S. Chanda, K. Guha, S. Patra, L. M. Singh, K. L. Baishnab, and P. K Paul, "An energy efficient 32 bit approximate Dadda multiplier," 2020 IEEE Calcutta Conference (CALCON), vol. 10, pp. 162–165, Feb. 2020.

[2] V. Manu, A. M. V. Prakash, and M. U. Chandra, "Design and implementation of sixteen-bit low power and area efficient DADDA multiplier," Design and Implementation of Sixteen-bit Low Power and Area Efficient DADDA Multiplier, vol. 5, pp. 631–636, May 2019.

[3] G. Prithi, G. Rithik, and K. Gurjit, "A design technique for delay and power efficient Dadda-Multiplier," *IEEE ICICCS 2021 Proceedings*, Oct. 2021.