

## Design and Analysis of Compressor based Dadda tree Multiplication

Jayakodi<sup>1</sup>(Jayakodimuhil@Gmail.Com) M.Kamalanathan<sup>2</sup>

<sup>1</sup>(Student, Gnanamani College Of Technology,Namakkal )

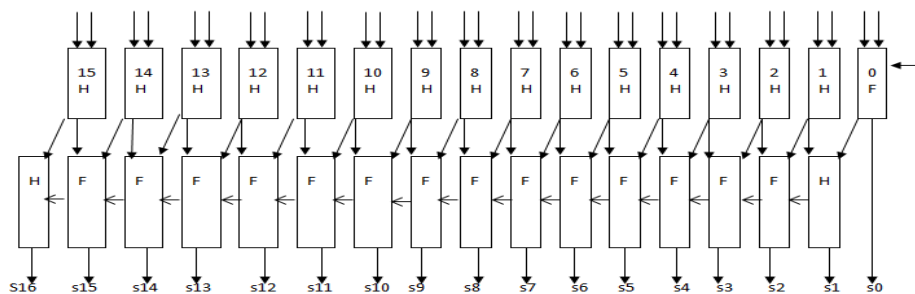
<sup>2</sup>(Ap/Ece, Gnanamani College Of Technology, Namakkal)

**Abstract :** A multiplier is one of the key hardware blocks in most digital signal processing (DSP) systems. Typical DSP applications where a multiplier plays an important role include digital filtering, digital communications and spectral analysis. Many current DSP applications are targeted at portable, battery-operated systems, so that power dissipation becomes one of the primary design constraints. There are several multipliers available to increase the performance level in the design field. Row bypassing multiplier with adaptive hold logic is used to reduce the power and area. The multiplier is able to provide higher throughput through the variable latency and can adjust the AHL circuit to mitigate performance degradation that is due to the aging effect. Moreover, this architecture can be applied to a column- or row-bypassing multiplier. This architecture increased delay. To overcome An appropriate design of an approximate compressor, multipliers can be designed for DSP applications. These multipliers offer significant advantages in terms of both circuit-level and error figures of merit. The Dadda tree multiplier designed using 4:2 compressor in the proposed design to be useful in other arithmetic circuits for applications in which inexact computing can be used.

**Keywords:** DSP, AHL , COMPRESSOR

### I. Introduction

Most of the VLSI circuits such as application specific DSP architectures and microprocessors used multiplier as a crucial portion, since they form the base element of all arithmetic functions. Generally as we know multiplication goes in three basic steps. Partial product generation, reduction and final stage are addition. For the reduction and the addition process, we can use the adder circuits. To minimize this logic required, it is assumed that ripple-carry adders and carry save adders are used. A ripple-carry adder is a set of single-bit adders connected serially. Notice that when we compute addition by hand, we only need to compute one sum digit and one carry at a time (a single-bit adder computes this in binary). Each carry is passed from right to left, hence the name ripple-carry adder.



**Fig. 1: basic multiplier using Ripple carry adder**

But it is very complex design and it consumes more power and area. But the digital signal processing requires high speed multipliers for real-time applications. To achieve this multiplication architecture, we reduce the adder depth i.e., number of adders. The adder depth is defined as the maximum number of adders that we pass through along any path from the input to any of the outputs in the constant multiplication logic circuit. The adder depth is an estimate of the longest path through the logic circuit, which is known as the critical path. As more logic gates are placed serially between the input and the output, the critical path becomes longer and the logic circuit must be clocked at a slower speed, which results in a lower computational throughput.

			a3	a2	a1	a0	
	X		b3	b2	b1	b0	
			a3b0	a2b0	a1b0	a0b0	pp0
			a3b1	a2b1	a1b1	a0b1	pp1
			a3b2	a2b2	a1b2	a0b2	pp2
			a3b3	a2b3	a1b3	a0b3	pp3
y6	y5	y4	y3	y2	y1	y0	

Fig. 2 Basic 4X4 multiplication process

### II. Existing System

The variable latency multiplier architecture for m bit is shown in the fig, which includes a m bit row-bypassing multiplier, normally it will operate at the in one clock cycle (minimum 50% of multiplicand is at logic "0" if less than that output of the multiplier is unstable because of less clock period), 2m 1-bit Razor flip-flops and an AHL circuit. The two inputs are given to the row bypass multiplier. The inputs are multiplicand and the multiplier. Row bypassing technique is based on number of zeros in the multiplier bits. If the corresponding multiplier bit is 0, the multiplexer select  $a_{ij}$  as the sum bit and zero as the carry bit. Razor flip-flops can be used to detect whether timing violations occur before the next input pattern arrives..

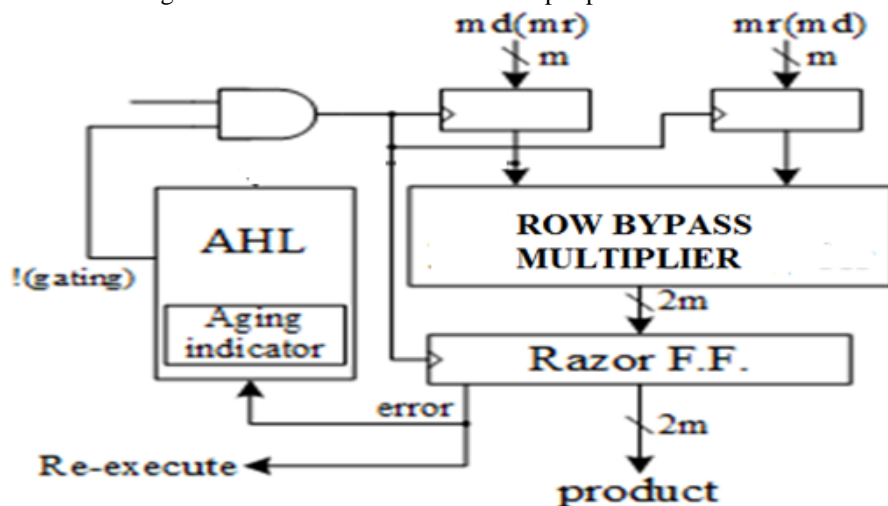


Fig3. Row Bypass Multiplier with AHL

### III. Proposed COMPRESSOR

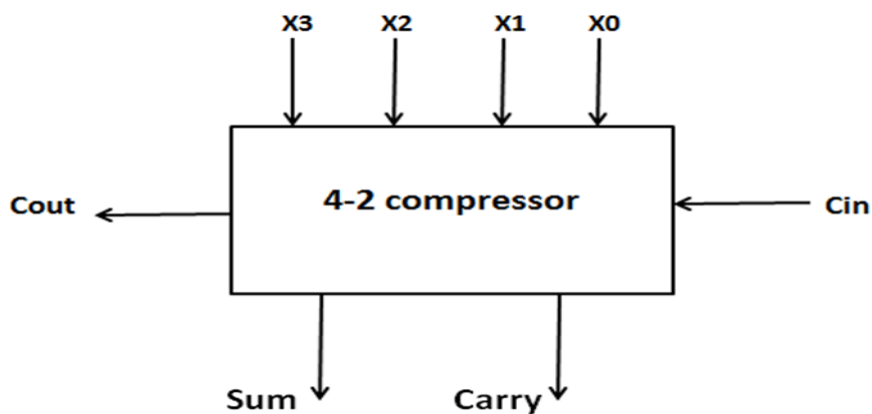
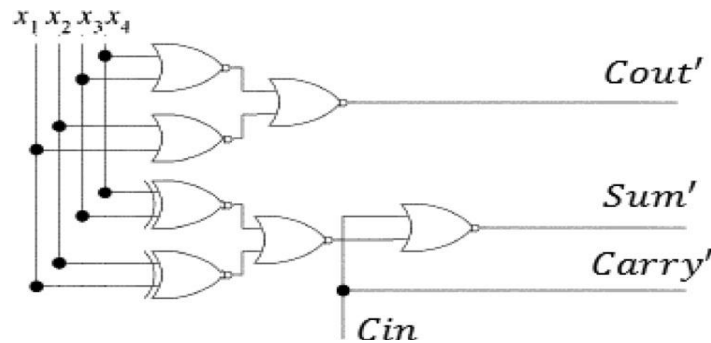


Fig.3 compressor diagram

Two designs of an approximate compressor are presented. Intuitively to design an approximate 4-2 compressor, it is possible to substitute the exact full-adder cells in Figure by an approximate full-adder cell. However, this is not very efficient, because it produces at least 17 incorrect results out of 32 possible outputs, i.e. the error rate of this inexact compressor is more than 53% (where the error rate is given by the ratio of the number of erroneous outputs over the total number of outputs). Two different designs are presented to reduce the error rate; these designs offer significant performance improvement compared to an exact compressor with respect to delay, number of transistors and power consumption. The approximate compressor design drives the inputs as  $x_1, x_2, x_3, x_4, cin$  and the outputs are  $sum, carry,$  and  $cout$ . And in the existing system, the implement the multiplier by the compressor using dadda's tree. The output equations are given below,

$$\begin{aligned}
 \text{Carry} &= cin \\
 \text{Sum} &= \overline{cin} (x_1 \oplus x_2 + x_3 \oplus x_4) \\
 \text{Cout} &= (\overline{x_1 x_2} + x_3 x_4)
 \end{aligned}$$

The gate level implementation of the existing design is,



**Fig. 4 gate level implementation of proposed compressor design**

The existing compressor gives the output carry as the input cin. So it leads to error in the output. The error rate of the existing system is 53 percentages.

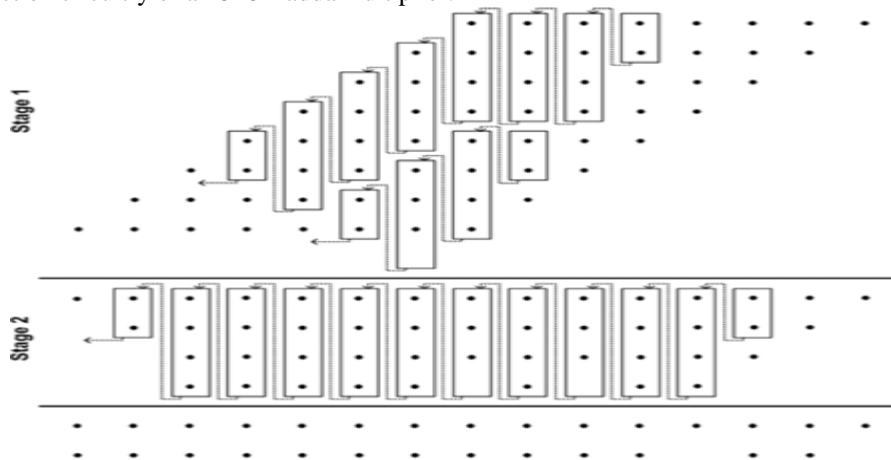
**Table: 1 truth table for proposed compressor design**

cin	x4	x3	x2	x1	cout	carry	sum	difference
0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	1	0
0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	1	-1
0	0	1	0	0	0	0	1	0
0	0	1	0	1	1	0	0	0
0	0	1	1	0	1	0	0	0
0	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	1	0
0	1	0	0	1	1	0	0	0
0	1	0	1	0	1	0	0	0
0	1	0	1	1	1	0	1	0
0	1	1	0	0	0	0	1	0
0	1	1	0	1	1	0	0	0
0	1	1	1	0	1	0	1	0
0	1	1	1	1	1	0	1	-1
1	0	0	0	0	0	1	0	1
1	0	0	0	1	0	1	0	0
1	0	0	1	0	0	1	0	0
1	0	0	1	1	0	1	0	-1
1	0	1	0	0	0	1	0	0
1	0	1	0	1	1	0	0	1
1	0	1	1	0	1	1	0	1
1	0	1	1	1	1	1	0	0
1	1	0	0	0	0	1	0	0
1	1	0	0	1	1	1	0	1
1	1	0	1	0	1	1	0	1
1	1	0	1	1	1	1	0	0
1	1	1	0	0	0	1	0	-1
1	1	1	0	1	1	1	0	0
1	1	1	1	0	1	1	0	0
1	1	1	1	1	1	1	0	-1

The table expresses the truth table for the proposed design of approximate compressor. In this table shows 12 incorrect outputs out of 32 outputs. The column difference shows the difference value between actual values to the output from the designed compressor.

**DADDA TREE MULTIPLICATION**

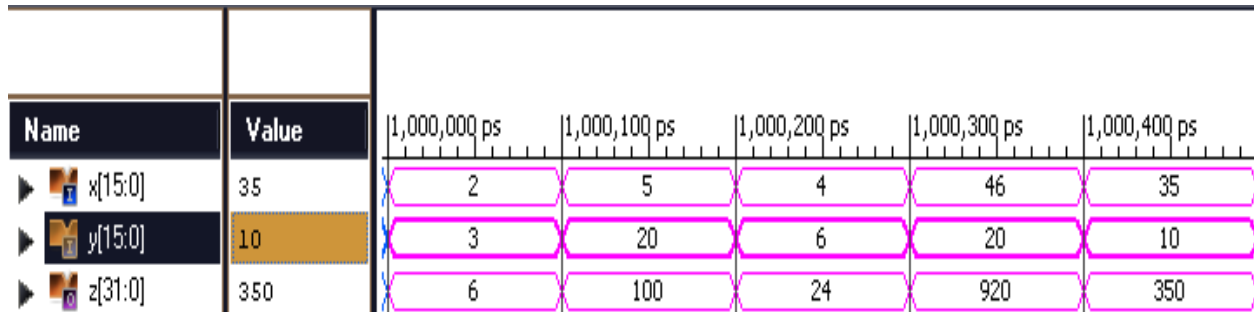
A 8x8 unsigned Dadda tree multiplier is considered to assess the impact of using the presented compressors in approximate multipliers. The multiplier uses in the first part AND gates to generate all partial products. In the second part, the approximate compressors proposed in the previous section are utilized in the CSA tree to reduce the partial products. The last part is an exact CPA to compute the final binary result. Figure shows the reduction circuitry of an exact multiplier for n=8. In this figure, the reduction part uses half-adders, full-adders and 4-2 compressors; each partial product bit is represented by a dot. In the first stage, 2 half-adders, 2 full-adders and 8 compressors are utilized to reduce the partial products into at most four rows. In the second or final stage, 1 half-adder, 1 full-adder and 10 compressors are used to compute the two final rows of partial products. Therefore, two stages of reduction and 3 half-adders, 3 full-adders and 18 compressors are needed in the reduction circuitry of an 8x8 Dadda multiplier.



**Fig. 5 Reduction circuitry of an 8x8Dadda multiplier, (a) using Design 1 compressors, (b) using Design 2 compressors**

In the first case (Multiplier 1), Design 1 is used for all 4-2 compressors in Figure. In the second case (Multiplier 2), Design 2 is used for the 4-2 compressors. Since Design 2 does not have cin and cout, the reduction circuitry of this multiplier requires a number of compressors (Figure 5(b)). Multiplier 2 uses 6 half adders, 1 full-adder and 17 compressors. In the third case (Multiplier 3), Design 1 is used for the compressors in then-1 least significant columns. The other n most significant columns in the reduction circuitry use exact 4-2 compressors. In the fourth case (Multiplier 4), Design 2 and exact 4-2 compressors are used in then-1 least significant columns and then most significant columns in the reduction circuitry respectively.

**IV. Results**



**Fig 6: Dadda tree simulation output**

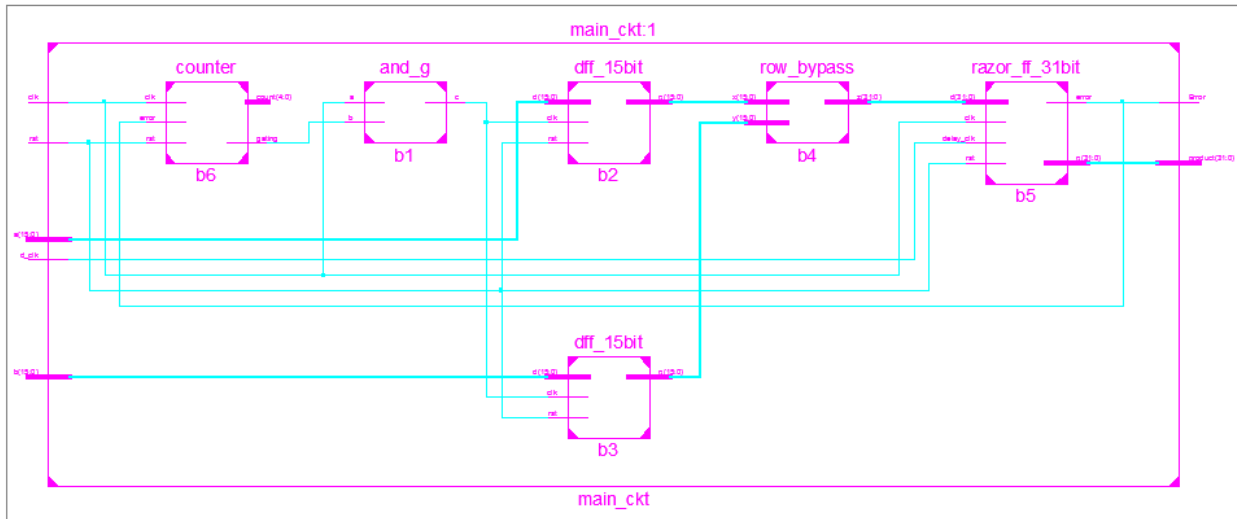


Fig 7: Synthesis RTL View output

### V. Conclusion

It has shown that by an appropriate design of an approximate compressor, multipliers can be designed for inexact computing. These multipliers offer significant advantages in terms of both circuit-level and error figures of merit. The proposed designs may also be useful in other arithmetic circuits for applications in which inexact computing can be used. The approximate compressors will show a significant reduction in transistor count, power consumption and delay compared with an exact design in further implementation. The approximate compressors have been utilized in the reduction module of a Dadda multiplier.

### References

- [1]. C. Chang, J. Gu, M. Zhang, "Ultra Low-Voltage Low- PoIr CMOS 4-2 and 5- Compressors for Fast Arithmetic Circuits," IEEE Transactions on Circuits & Systems, Vol. 51, No. 10, pp. 1985-1997, Oct. 2004.
- [2]. D. Radhakrishnan and A. P. Preethy, "Low-poIr CMOS pass logic 4-2 compressor for high-speed multiplication," in Proc. 43rd IEEE MidIst Symp. Circuits Syst., vol. 3, 2000, pp. 1296-1298
- [3]. B. Parhami, "Computer Arithmetic: Algorithms and Hardware Designs," 2nd edition, Oxford University Press, New York, 2010.
- [4]. K. Prasad and K. K. Parhi, "Low-poIr 4-2 and 5-2 compressors," in Proc. of the 35th Asilomar Conf. on Signals, Systems and Computers, vol. 1, 2001, pp. 129-133.
- [5]. Ercegovac, Miloš D., and Tomas Lang. Digital arithmetic. Elsevier, 2003.