Research Journal of Applied Sciences, Engineering and Technology 9(1): 53-57, 2015
DOI:10.19026/rjaset.9.1376
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: November 13, 2014 Accepted: November 13, 2014 Published: January 05, 2015

Research Article

Low Power and Efficient Dadda Multiplier

S. Ravi, Govind Shaji Nair, Rajeev Narayan and Harish M. Kittur
School of Electronics Engineering, VIT University, Vellore, Tamilnadu 632014, India

Abstract: In this study an area optimized Dadda multiplier with a data aware Brent Kung adder in the final addition stage of the Dadda algorithm for improved efficiency has been described in 45 nm technology. Currently the trend is to shift towards low area designs due to the increasing cost of scaled CMOS. An area reduced full adder is the key component in our design. It uses lesser number of gates than conventional design and hence lesser area and delay. The data aware Brent Kung adder in the final addition stage helps in reducing dynamic power as it reduces switching activity depending on the inputs. We have compared the results to the existing benchmark designs and our experimental results show that we have been capable of reducing the area by 13.011% and total power by 26.1% with only a slight increase in the delay.

Keywords: Brent Kung adder, dadda multiplier, data aware, low area

INTRODUCTION

Column compression multipliers are gaining popularity in high performance digital systems due to their higher speeds (Parhami, 2000; Swartzlander Jr. and Goto, 2002). Wallace and Dadda are the two main types of the column compression multipliers. A close consideration of the Wallace and Dadda multipliers has been carried out since the 2000's and it's been proven that the Dadda multiplier is faster and hardware requirement is lesser in comparison to Wallace multiplier (Bickerstaff *et al*., 2001; Townsend *et al*., 2003). Hence we have chosen the Dadda multiplier for our design.

The increasing cost of scaled CMOS and the current shift towards smaller chip sizes has increased the demand for area reduction in digital systems. The main components of the Dadda multiplier is the full adder and half adder modules that are used for performing the reductions of the partial products (Arunachalam Kirubaveni, 2013). Therefore significant reductions in area can be made by optimizing the conventional full adder and half adder modules.

MATERIALS AND METHODS

Generalized architecture: The Dadda multiplier (Parhami, 2000) is a hardware multiplier design invented by computer scientist (Dadda, 1965).

It is similar to the Wallace multiplier, but it is slightly faster (for all operand sizes) and requires fewer gates (for all but the smallest operand sizes) (Bickerstaff *et al*., 2001; Townsend *et al*., 2003). Dadda partial product reduction scheme is shown in Fig. 1.

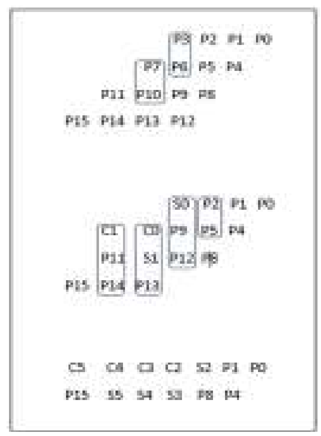

Fig. 1: Reduction of partial products of 4×4 dadda approach

The three main steps in DADDA are as follows (Ramkumar *et al*., 2011; Dadda, 1965):

- Generation of the partial products in parallel using an array of AND gates.
- Reduction of the partial products using exact placement of the (3, 2) counters and (2, 2) counters in the maximum critical path delay of the multiplier.
- The final set of partial products remaining after the reduction phase are added using a conventional adder.

Dadda multipliers do as few reductions as possible. Because of this, Dadda multipliers have a less expensive reduction phase, but the numbers may be a

Corresponding Author: S. Ravi, School of Electronics Engineering, VIT University, Vellore, Tamilnadu 632014, India

few bits longer, thus requiring slightly bigger adders in the final stage.

Optimized proposed architecture:

Low area full adder: As mentioned earlier, the full adder module is one of the main components of the Dadda multiplier. The conventional full adder shown in Fig. 2 has 2 EXOR gates, 2 AND gates and 1 OR gate.

The Sum and Carry formula in conventional full adder is:

Sum = (A^B^Cin)

Carry = ((Cin. (A^B)) + (A.B))

After careful evaluation of the 10 Transistor Full adder design mentioned in Rani *et al*. (2011), we were able to design a full adder using just 2 EXNOR gates and 1 MUX.

The Sum and Carry formula for proposed full adder is:

X1 = ~(A^B)

Sum = ~(X1^Cin)

Carry = X1? A: Cin ----> {Describes a MUX}

Thus the proposed full adder shown in Fig. 3 not only reduces area and number of components but delay as well since propagation delay is reduced due to less number of components.

Data aware brent kung adder: The Brent-Kung adder is a parallel prefix adder. Parallel prefix adders are special class of adders that are based on the use of generate and propagate signals. Simpler Brent-Kung adders have been proposed to solve the disadvantages of Kogge-Stone adders. The cost and wiring complexity is greatly reduced. But the logic depth of Brent-Kung adders increases to 2log (2n-1), so the speed is lower (Pudi and Sridharan, 2012). We propose a method to reduce delay and power consumed by the Brent Kung adder by analyzing and dividing the inputted data into blocks that will only be added if it holds any value at all. Hence the inputted values are initially compared before deciding up to how many of theadder blocks should be activated. This approach can easily be integrated to the existing design of the Brent Kung adder, thus making it more efficient. The block diagram of 4-bit Brent-Kung adder is shown in Fig. 4.

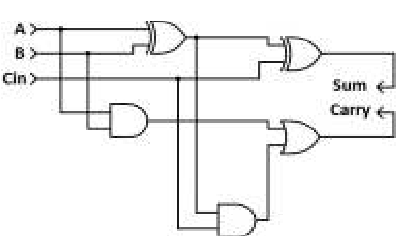

Fig. 2: Conventional full adder circuit diagram

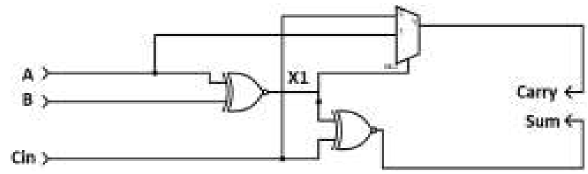

Fig. 3: Optimized full adder circuit diagram

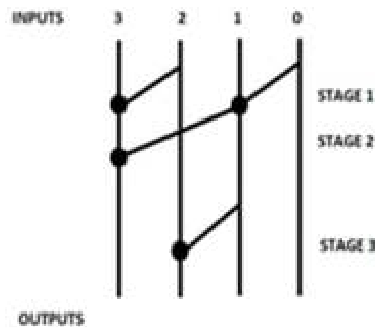

Fig. 4: Carry generation of 4-bit Brent Kung adder

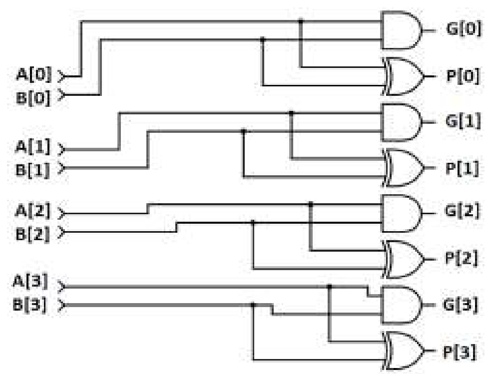

Fig. 5: Preprocess circuit diagram for 4-bit Brent Kung adder

The 4-Bit Brent Kung adder module has 3 main stages of operation. They are explained in detail below.

Preprocess-generate and propagate:

Propagate: A XOR B

Generate: A and B

The preprocess stage shown in Fig. 5 calculates the initial propagate and generate signals directly from the inputs using the formulae mentioned above. Since this is 4-bit module, 4 initial propagate and generate signals will be calculated as shown in the circuit diagram.

Black dot functionality:

Propagate: P0 and P1

Generate: G0 and P1+G1

The second stage calculates both the internal as well as final carry using the initial generate and propagate signals and using the black dot operation as shown in Fig. 4. The Black dot operation takes in 2 pairs of generate and propagate signals and gives a single pair of generate and propagate signal as output as

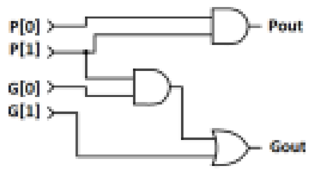

Fig. 6: Circuit diagram depicting black dot functionality

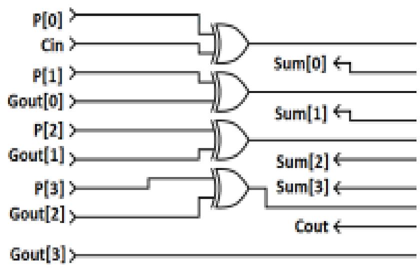

Fig. 7: Postprocess circuit diagram for 4-bit Brent Kung adder

per the formulae mentioned above is shown in Fig. 6. Hence at the end of second stage, we would have obtained 4 pairs of final propagate and generate signals. The final generate signals outputted represent the carry of that stage.

Postprocess-sum and carry:

SUM [i] = A [i] ^B [i] ^C [i-1]
= P [i] ^C [i-1]
CARRY = FINAL GENERATE SIGNAL CALCULATED ON THE MSB

The post process shown in Fig. 7 uses the generate (carry) signals outputted at the end of the second stage in calculating the sum and final output carry using the formulae that are mentioned above.

The final stage of the 64-bit Dadda multiplier is the addition operation which is usually performed by conventional adders. Here we have implemented the final stage addition using two 64-bitdata aware Brent Kung adders connected in a cascading manner. For the case of 64-bit bit data aware Brent Kung adder, 16 four bit BrentKung adders, each with its own enable line were made and joined such that they could perform a 64 bit addition operation to generate a 64 bit output.

Here the adder is made data aware by first analyzing the input to find the highest MSB bit in the two operands and also to compare as to which operand is bigger using the algorithm depicted above. Based on the results drawn from it, the 64-bit adder only activates up to that many 4-bit Brent Kung adder modules using the enable lines (Fig. 8).

The remaining 4-bit modules are deactivated and hence switching activity is reduced leading to reduction in dynamic power consumption (Zhou and Guo, 2008) making the Dadda multiplier more efficient.

For example, consider 2 inputs (63:0) A and (63:0) B are fed into the Brent Kung adder. A has a higher MSB than B and the MSB of A is at the 52^{nd} bit. In this case after checking the inputs, the enable lines 0 to 13 will be activated while 14 and 15 will be deactivated. If the MSB bit was at 10^{th} bit, then only enable lines 0 to 2 will be activated while the remaining enable lines 3 to

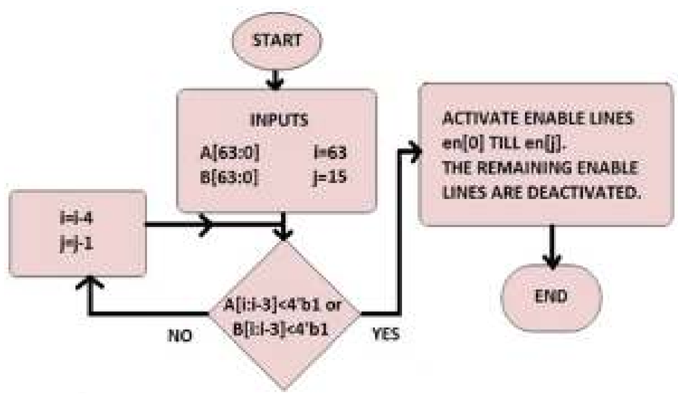

Fig. 8: Algorithm used for selection of enable lines based on input (data aware)

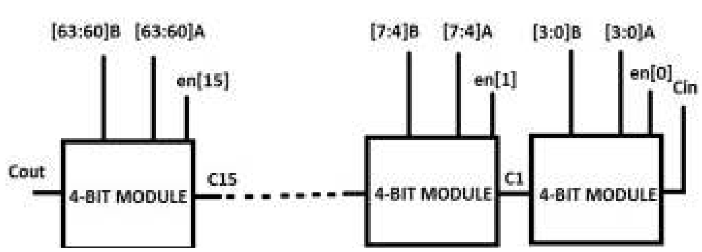

Fig. 9: Sixty four bit Brent Kung adder made of 4-bit Brent Kung modules with specific enable lines for each module

15 will be deactivated. Hence depending on the data in the inputs, the switching activity varies resulting in more efficient dynamic power management.

The diagram Fig. 9 above depicts a 64-bit data aware Brent Kung adder formed by the cascaded network of 16 4-bit Brent Kung adder modules, each with a specific enable line.

RESULTS AND DISCUSSION

For the RTL synthesis we have used the CADENCE RTL compiler. For the process we first created a setup file for 45 nm technology using the slow library functions for finding the specifications for the worst case scenario. In the setup files we called the top module of each architecture and giving a virtual clock we synthesized to find various stipulations like Time Slack, Leakage and Dynamic Power, Total Area and The number of cells. The various readings obtained from the RTL synthesis of all the architectures are:

Dadda with conventional Full Adder shown in Table 1, Dadda with proposed full adder shown in Table 2. Proposed architecture Table 2 values are positive to move the architecture further.

After estimating and analyzing the various parameters of the design shown in Table 3 and 4 after synthesis, we can come to the conclusion that area optimized Dadda multiplier performs better than the conventional Dadda multiplier. The percentage decrease in the parameters between the conventional Dadda and area optimized Dadda multipliers are shown Table 5.

Table 1: Dadda with conventional full adder

Size	Area (um^2)	Delay (nsec)	Power (mW)
8×8	517	3.800	0.194
16×16	2183	6.310	0.708
32×32	8641	11.440	2.068
64×64	38412	21.673	5.933

Table 2: Dadda with proposed full adder

Size	Area (um^2)	Delay (nsec)	Power (mW)
8×8	449	3.700	0.181
16×16	1954	6.278	0.679
32×32	8110	11.386	2.048
64×64	33029	21.606	5.743

Table 3: Area optimized 64-bit dadda with conventional adder in final stage

Parameters	64-bit dadda
Area (um^2)	33029
Delay (nsec)	21.606
Power (mW)	5.743

Table 4: Area optimized 64-bit dadda with proposed adder in final stage

Parameters	64-bit dadda
Area (um^2)	33414
Delay (nsec)	30.400
Power (mW)	4.444

Table 5: Comparison between conventional dadda and area optimized dadda multipliers

Size	Area (%)	Delay (%)	Power (%)
8×8	-13.15	-2.60	-6.70
16×16	-10.40	-0.51	-4.10
32×32	-6.14	-0.47	-0.96
64×64	-14.01	-0.31	-3.16

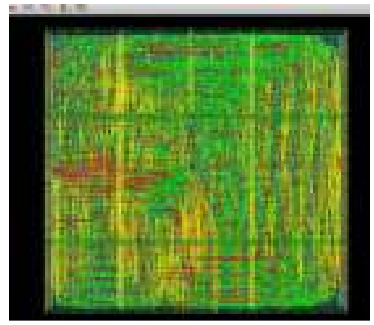

Fig. 10: Physical view of area optimized dadda multiplier with data aware Brent Kung adder in final stage

CONCLUSION

By using the proposed data aware Brent Kung Adder in the final stage of the area optimized 64 bit Dadda multiplier, we were able to reduce total power by 26.1 and 22.6% when compared to conventional 64 bit Dadda multiplier and area optimized 64 bit Dadda multiplier. Hence the proposed final Dadda multiplier is more efficient in terms of power management and area when compared to the conventional Dadda multiplier.

The proposed area efficient Dadda multiplier with data aware Bret Kung adder in its final addition stage was taken to the backend process and the final physical view just before the generation of the GDSII file was obtained.

The backend process of power plan, floor plan, optimization and placement were all done in cadence encounter tool using 45 nm technology. The total die size was found to be 47733.97 um^2. Figure 10 shown above depicts the physical view of the area optimized Dadda multiplier with Data aware Brent Kung adder in final stage obtained using the Cadence encounter software.

REFERENCES

Arunachalam, T. and S. Kirubaveni, 2013. Analysis of high speed multipliers. Proceeding of International Conference on Communications and Signal Processing (ICCSP, 2013), pp: 211-214.

Bickerstaff, K.C., E.E. Swartzlander and M.J. Schulte, 2001. Analysis of column compression multipliers. Proceeding of 15th IEEE Symposium on Computer Arithmetic, pp: 33-39.

Dadda, L., 1965. Some schemes for parallel multipliers. Alta Freq., 34: 349-356.

Parhami, B., 2000. Computer Arithmetic. Oxford University Press, New York.

Pudi, V. and K. Sridharan, 2012. Low complexity design of ripple carry and Brent-kung adders in QCA. IEEE T. Nanotechnol., 11(1): 105-119.

Ramkumar, B., V. Sreedeep and H.M. Kittur, 2011. A Design Technique for Faster Dadda Multiplier. Retrieved form: http://arxiv.org/ftp/arxiv/papers/1110/1110.3281.pdf.

Rani, T.E., M.A. Rani and R. Rao, 2011. Area optimized low power arithmetic and logic unit. Proceeding of 3rd International Conference on Electronics Computer Technology (ICECT, 2011), pp: 224-228.

Swartzlander Jr., E. and G. Goto, 2002. Computer Arithmetic. In: Oklobdzija, V.G. (ed.), the Computer Engineering Handbook. CRC Press, Boca Raton, FL.

Townsend, W.J., E.E. Swartzlander and J.A. Abraham, 2003. A comparison of Dadda and Wallace multiplier delays. Proceeding of SPIE Advanced Signal Processing Algorithms, Architectures and Implementations XIII, 5205: 552-560.

Zhou, Y. and H. Guo, 2008. Application specific low power ALU design. Proceeding of IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC'08), pp: 214-220.