ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization Vol.5, Special Issue 2, April 2017

An International Conference on Recent Trends in IT Innovations - Tec'afe 2017

Organized by

Dept. of Computer Science, Garden City University, Bangalore-560049, India

Computing Performance Enhancement through Fast Multiplier Using Karnaugh Map in Cloud

Vikash Kumar

Assistant Professor, Dept. of Computer Science & IT, JAIN University, Bangalore, India

ABSTRACT: A multiplier is one of the key hardware blocks in most digital and high performance systems such as FIR filters, digital signal processors and microprocessors etc. Multiplication represents one of the major bottlenecks in the digital processing systems. Generally, multiplication goes in two basic steps, generating partial product and then addition. Depending on the word size, partial products are formed and are added to evaluate the final product. The well-known Shift-and-add algorithm uses minimum hardware but has unacceptable performance for most applications. Several fast multiplication schemes have been suggested using several levels of blocks containing full adders. Now days, many digital signal processing systems are targeted at portable, battery-operated systems, so power dissipation is one of the primary design constraints. Since multipliers are rather complex circuits and typically must operate at a high system clock rate, reducing logic transitions and thus the power dissipation of multipliers is a key to satisfying the overall power budget. This paper presents performance of Fast Multiplier using Karnaugh Map.

KEYWORDS*:* DSP, Latency, FIR, CPA, CSA.

I. INTRODUCTION

With the rapid development of handheld electronics, the need for high performance and low power signal processing systems with very high data throughput has grown in recent years. To obtain the necessary throughput and a high power-performance ratio, efficient multiplier design is necessary for signal processing applications. Digital multipliers are needed in many system applications, including digital filters, correlates and neural networks and used in various multimedia processing. Multiplication represents a fundamental building block in all DSP tasks. Due to large latency inherent in multiplication, schemes have been devised to minimize the delay.

High speed multiplication is a critical function in a range of very large scale integration (VLSI) applications. Multiplications are expensive and slow operations as many partial products are added to produce the product. The performances of many computational problems are often dominated by the speed at which a multiplication operation could be executed.

Wallace [1] suggested the idea of pseudo-adders, which *are* essentially arrays of full adders without rippling carries, which take three inputs and reduce them to two outputs. Wallace uses pseudo-adders at several levels in summation of the partial
products. Doing so finally left with two partial products. At this point, a fast carry propagate adder (CPA) such as carry look ahead adder, carry skip adder, carry select adder etc. is used to determine final product.

Dadda [2], [3] generalized the idea of full adder to reduce the partial product matrix by the concept of (n, m) parallel counter. An (n, m) parallel counter is a combinational network with n inputs and m outputs where the output shows the count of number of inputs that are ONEs. Thus, full adder is a (3, 2)
Counter. Dadda postulated that, at each stage, only minimum amount of reduction should be done in order to reduce the partial product matrix to the lower number in the sequence. This paper describes, multiplication scheme using (2, 2) counter an (3, 2) counter.

1.2 MULTILICATION

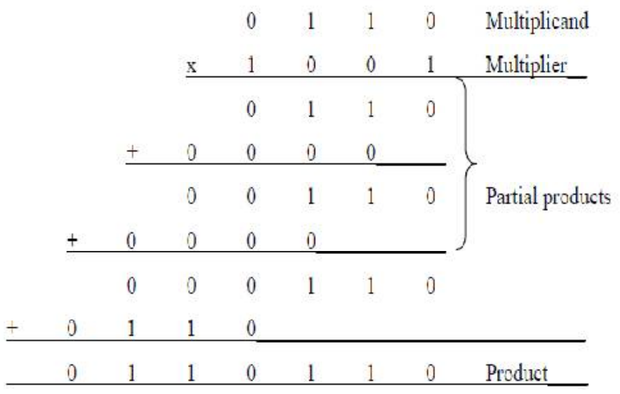
Multiplication is an important basic arithmetic operation which is less common operation than addition, but is still essential for microprocessors, digital signal processors and graphic engines [4]. Multiplication algorithms are used to illustrate methods of designing different cells so that they fit into a larger structure. Multiplication is logically carried out by a sequence of addition, subtraction and shift operations. It is an expensive and slow operation. VLSI designers have recognized this by dedicating significant area to integer and floating point multipliers. Therefore, high speed multiplication can be achieved by having a high speed multiplier.

1.3 MULTIPLIER

Multipliers are in effect complex adder arrays [6]. From the Wikipedia encyclopedia [10] in digital design, a multiplier is a hardware circuit dedicated to multiply two binary values. A variety of computer arithmetic techniques can be used to implement a digital multiplier. Most techniques involve computing a set of partial products, and then summing the partial products together. This process is similar to the method taught to primary school children for conducting long multiplication on base-10 integers, but has been modified here for application to a base-2 (binary) numeral system. At each step we multiply one digit of the multiplier by the full multiplicand then we add the result shifted by the proper number of bits to get the partial product. When we run out of multiplier digits, we stop. Single-digit multiplication is easy for binary number i.e., multiplication of two bits is performed by the AND function. The computation of partial products and their accumulation into the complete product can be optimized in many ways, but an understanding of the basic steps in multiplication is important to a full appreciation of those improvements.

A high speed multiplier is an electronic computing unit used to provide multiplication processes at a very high speed by multiplier; a partial product is obtained from the initial addition of the multiplicand and the multiplier. Then, the partial product is added together in order to obtain the final product.

Consider the simple example below:



1.3.1 MULTIPLIER TOPOLOGIES

There are two types of multiplier topologies which are regular or serial topology and irregular or parallel topology. The summation of the partial products is done using some variation of a carry-save adder, referred to generally as counters [4]. These counters can be connected by several different methods. As used here, topology refers to implementation differences in the way the counters are connected, the allowable number of wires per wiring channel, and the length of the wires required to connect the counters [4].

1.3.1.1 REGULAR TOPOLOGIES

In a regular or serial topology, the counters are connected in a regular pattern that is replicated. The regular connections make the design of the partial product array a hierarchical design. A serial multiplier works in a manner similar to manual multiplication of two decimal numbers, although two binary numbers are multiplied in this case. The only difference between this serial multiplier and manual multiplication is the repeated addition of each multiplicand-multiple, instead of one-time addition of all multiplicand-multiples at the end.

The regular topologies are the topologies most commonly used in custom design, since they provide a compromise between optimization and design effort [4]. The regularity allows designers to build a small group of building blocks that contain connected counters and compressors and then connect these blocks to form the topology. For the regular topologies, the maximum number of counters in series defines the delay of the topology. Regular topologies can be classified as either array topologies or tree topologies [4]. In array topologies, the counters are connected mostly serially [4]. The array topology is a two-dimensional (2D) structure that fits nicely on the VLSI planar process.

There are several possible array topologies, including simple, double, and higher-order arrays [4]. Tree topologies are very fast structure for summing partial products. In a tree, counters are connected mostly in parallel [4]. Although trees are faster than arrays, they both use the same number of counters to reduce the partial products. The difference is in the interconnections between the counters. Trees create a 3D structure. However, integrated circuits are planar these trees must be flattened to fit in the 2D plane. The flattening is achieved by placing the counters and compressors linearly [4].

1.3.1.2 IRREGULAR TOPOLOGIES

In an irregular or parallel topology, the counters are connected in order to minimize the total delay disregarding the ease of laying out the multiplier [4]. They do not have a regular pattern for connecting the counters. There are several types of parallel multipliers in digital design with different speeds, areas, and configurations such as Bough-Wooley array multiplier, Braun array multiplier, Modified Booth multiplier, Wallace Tree multiplier, Dadda multiplier and Ferrari Stefanelli multiplier. The Bough-Wooley array multiplier was developed to perform two’s complement multiplication [7]. It implements with a full adder tree. The Modified Booth algorithm has been extensively used in multipliers with long operands (> 16 bits). Its principle is based on recoding the two’s complement operand to reduce the number of partial products to be added. The reduction of partial products improves the performance of the multiplier [9].

The Wallace Tree multiplier is a high speed multiplier. It is a full adder structured specially for a quick addition of the partial products. The Dadda multiplier operation is similar to that of Wallace Tree multiplier. The Ferrari – Stefanelli multiplier is also called “nests” multiplier as the 2-bit sub-multipliers in the main multiplier reduce the number of partial products [8]. Although there are many types of parallel multiplier this project will mainly focusing on the study of the Wallace Tree multiplier as one of the high speed multiplier of all it kinds.

II. FAST MULTIPLICATION

Fast multiplication is done using (2, 2) counters and (3, 2) counter by reducing the partial products to two. Ripple carry adder is used to obtain the final product.

2.1 COUNTERS

A Half adder is an implementation of (2, 2) counter which takes two inputs A, B and the output are given by sum and carryout. Figure 1 shows the block diagram of (2, 2) counter. This (2, 2) counter is implemented using XOR and AND gates. The inputs to the XOR gate are also the inputs to the AND gate. Figure 1(a) shows the (2, 2) counter implemented using XOR and AND gates.

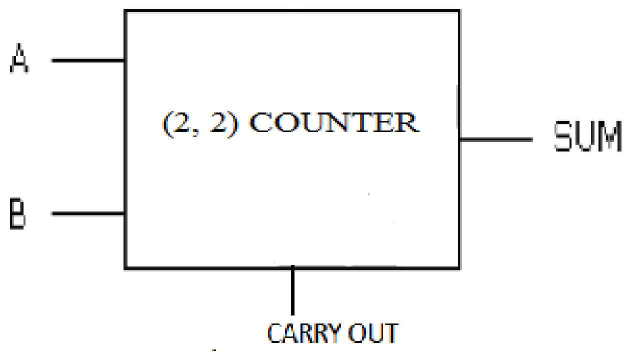


Figure 1. (2, 2) Counter

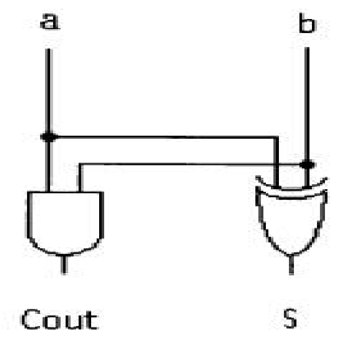


Figure 1(a). (2, 2) Counter using XOR and AND gates

A full adder implementation is similar to (3, 2) counter or carry save adder (CSA) which takes three inputs A, B and Cin, the outputs are given by sum and carry out. Figure 2 shows the block diagram of (3, 2) counter. This (3, 2) counter is implemented using XOR and AND. The inputs to the XOR gate are also the inputs to the AND gate. Figure 2(a) shows (3, 2) counter implemented using XOR and AND gates.

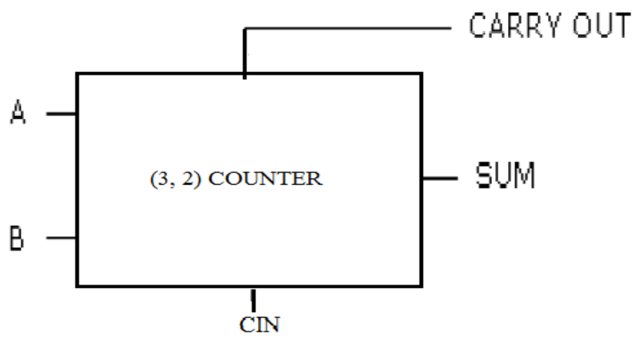


Figure 2. (3, 2) Counter

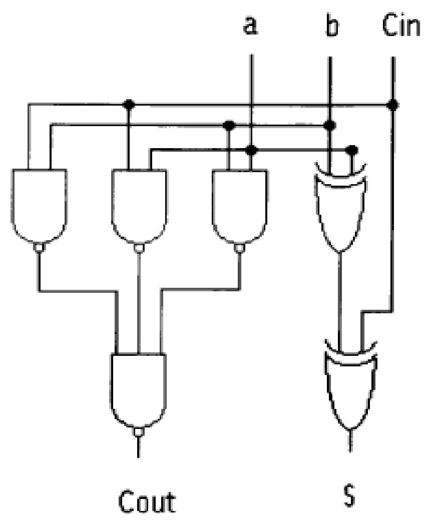


Figure 2(a). (3, 2) Counter using XOR and AND gates.

2.2 RIPPLE CARRY ADDER

Figure 3 shows the block diagram of Ripple carry adder. The Ripple carry adder is implemented by means of half adder and multiple full adders to have n-bit addition. It is a type of carry propagate adder where the carry is propagated through each stage in order to get the final sum and carryout. This adds the gate delays between each gate for the carry propagation.

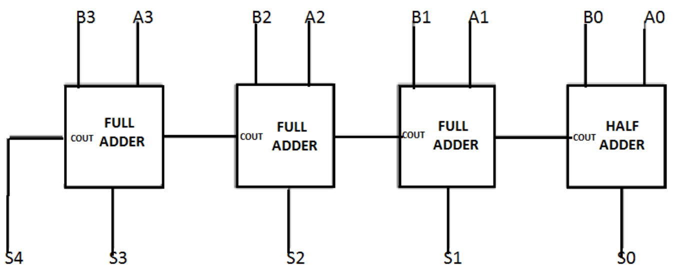
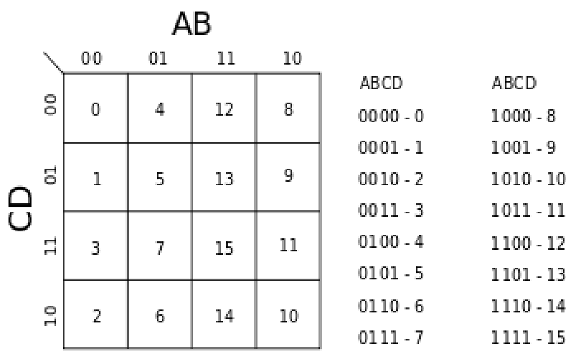


Figure 3. Block diagram of Ripple carry adder

2.3KARNAUGH MAP

The Karnaugh map (K-map for short), Maurice Karnaugh's 1953 refinement of Edward Veitch's 1952 Veitch diagram, is a method to simplify Boolean algebra expressions. The Karnaugh map reduces the need for extensive calculations by taking advantage of humans' pattern-recognition capability, also permitting the rapid identification and elimination of potential race conditions.

In a Karnaugh map the boolean variables are transferred (generally from a truth table) and ordered according to the principles of Gray code in which only one variable changes in between adjacent squares.

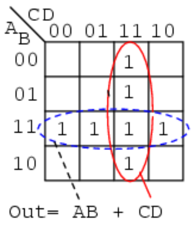


CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

ABCD	ABCD
0000 - 0	1000 - 8
0001 - 1	1001 - 9
0010 - 2	1010 - 10
0011 - 3	1011 - 11
0100 - 4	1100 - 12
0101 - 5	1101 - 13
0110 - 6	1110 - 14
0111 - 7	1111 - 15

Table 1. Four variables representation using K-Map.

$$\text{Out}= \bar{A}\bar{B}CD+\bar{A}BCD+ABCD+A\bar{B}CD+AB\bar{C}\bar{D}+AB\bar{C}D+ABC\bar{D}$$



AB \ CD	00	01	11	10
00			1	
01			1	
11	1	1	1	1
10			1	

$$\text{Out}= AB + CD$$

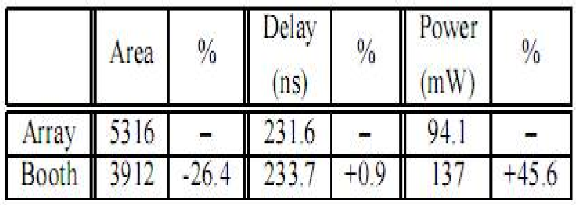
Table 2. Solving Boolean expression using K-Map.

The approach used in Table 2. for solving the boolean expression can be implemented with some modification to reduce the number of partial products in multiplication. By using this approach fastest multiplication circuit can be achieved. I propose to implement this logic in cloud environment to increase computing efficiency.

III. RESULTS

For a fast multiplication of 8 bit operands, we had constraints in reducing partial products. The complexity of an 8 by 8 bit multiplier is, it uses thirty eight (3, 2) counters, fifteen (2, 2) counters and a 10-bit carry propagate adder. The objective of the project is to focus on the speed of the multiplier, so the increased area of the multiplier is not taken into count. This is achieved by designing Wallace multiplier for an 8 by 8 bit using (2, 2) counters and (3, 2) counters where partial products were reduced in five stages. Designed schematic and verilog source code is simulated and checked for its functionality.

In the Hybrid radix- 2^m multiplier, it is possible to obtain significant power reduction mainly due to the less amount of switching activity and glitching transitions in the circuit. Table 2 presents area, delay and power results for radix-4 Booth multiplier and the 2's complement m=2 Hybrid arraymultiplier [12].



	Area	%	Delay (ns)	%	Power (mW)	%
Array	5316	-	231.6	-	94.1	-
Booth	3912	-26.4	233.7	+0.9	137	+45.6

Table 3. AREA, DELAY AND POWER FOR 2'S COMPLEMENT PARALLEL MULTIPLIERS (M=2).

Here the hybrid architecture is significantly more efficient, with no delay penalties and 45% less power consumption. Whereas fast multiplier using counter produces faster results even for 8 bits. But with increase in hardware.

III. CONCLUSION

A scheme for fast multiplication using (2, 2) counter and(3, 2) counter is designed and discussed. This indicates that the multipliers implemented using counters have better performance and fast as the partial products were reduced to two in few stages but by using K-Map approach it will reduce more number of steps and the fastest multiplication can be achieved.

IV. FUTURE SCOPE

Hererepresenting multiplicand and multiplier in term of boolean expression is the most critical and complex task. For future development of this project, this complexity can be reduced to achieve greater performance in cloud computing environment.

REFERENCES

[1] C. S Wallace, "A suggestion for a Fast Multiplication," *IEEE Transactions on Electronic Computers Vol* EC-13, pp. 14-17, 1969
[2] L. Dadda, "Some Schemes for Parallel Multipliers," *Alta Frequenza*, Vol.34, pp. 349-356, 1965.
[3] L. Dadda, "On Parallel Digital Multipliers," *Alta Frequenza*, Vol. 34, pp.574-580, 1976.
[4] Michael J. Flynn, and Stuart F. oberman, (2001). *Advanced Computer Arithmetic Design*, John Wiley & Sons Inc., USA.
[5] Vikash Kumar& Chanda Mishra,"Performance Based Comparative Study of Fast Multiplier Using Counter with 2's Complement Gray Encoded Multiplier" NAAC-2012, MVJCE, Bangalore.
[6] Jan M. rabaey, Anantha Chanderkasan, and Borivoje Nikolic, (2004). Digital *Integrated Circuits. A Design Perspective*, 2nd Edition, Prentice Hall, USA.
[7] John P. Uyemura, (2002). *Introduction to VLSI Circuits Systems*, John Wiley & Sons Inc., USA.
[8] Michael John Sebartian Smith, (2003). *Application-Specific Integrated Circuits*, Addison Weslay, USA.
[9] M. Michael Vai. (2002), *VLSI Design*, CRC Press, USA.
[10] Wikipedia Encyclopedia, (2006), Multiplier http://en.wikipedia.org/wiki/multiplier.
[11] Whitney J. Townsend, *Earl E. Swartzlander, Jr. , ** and Jacob A. Abraham*, "*A Comparison of Dadda & Wallace Multipliers Delays*"
[12] E. da Costa, J. Monteiro, and S. Bampi." A New Architecture for 2's Complement Gray Encoded Array Multiplier