Structural VHDL Implementation of Wallace Multiplier

Jasbir Kaur, Kavita

Abstract— Scheming multipliers that are of high-speed, low power, and standard in design are of substantial research interest. By reducing the generated partial products speed of the multiplier can be increased. Several attempts have been made to decrease the number of partial products generated in a multiplication process. One of the attempt is Wallace tree multiplier. This paper aims at designing and implementation of Wallace tree multiplier. Speed of Wallace tree multiplier can be enhanced by using compressor techniques. By minimizing the number of half adders and full adders used in a multiplier reduction will reduce the complexity.

Index Terms— Adder, Full adder, Half adder, Multiplier, Ripple Carry adder, Structural, Wallace Tree.

1 INTRODUCTION

THIS digital signal processing (DSP) is one of the core technologies in multimedia and communication systems. Many application systems based on DSP, especially the recent next-generation optical communication systems, require extremely fast processing of a huge amount of digital data [2]. Multiplication operation is essential in DSP Applications. Multiplication requires large processing time than the addition and subtraction. The multipliers play a key role in arithmetic operations in digital signal processing applications [6].Hence the need of low power multipliers has increased. C.S.Wallace suggested a fast multiplier during 1964 with the combination of half adders and full adders. During that period the need for low power designs was not up to the mark, but in coming years the need for compatible and advanced systems made a demand for the new designs of basic circuits with low power, delay and fast working [4].
This paper basically describes a method of implementation of Wallace Tree Multiplier explaining the use of half and full adders for addition of intermediate product terms obtained after the multiplication of two nibbles (4 bits). In this paper multiplication of 4x4 numbers and 8x8 numbers are presented. The structure of Wallace Tree Multiplier is explained for 4x4 multiplier and 8x8 multiplier and their simulation results are also shown.

The simulation of this Wallace multiplier is done using Modelsim simulator. The waveforms of the results are shown along with the product bits obtained after multiplication.

- *Jasbir Kaur is currently working as Assistant Professor in Electronics and Electrical Communication Engineering in PEC University of Technology, India, PH-9876232454. E-mail: jasbirkaur70@yahoo.com*
- *Kavita is currently pursuing masters degree program in electronics engineering in PEC University of Technology, India, PH-9467248606. E-mail: kavitasaharan@hotmail.com*

2 MULTIPLIER

A basic multiplier consists of three parts (i) partial product generation (ii) partial product addition and (iii) final addition. A multiplier essentially consists of two operands, a multiplicand "A" and a multiplier "B" and produces a product "P". In the first stage, the multiplicand and the multiplier are multiplied bit by bit to generate the partial product terms. The second stage is the most important, as it is the most complicated and determines the overall speed of the multiplier. This stage includes addition of these partial product terms to generate the product "P". This paper will be more focused on the optimization of this stage, which consists of the addition of all the partial products [5]. If speed is not an issue, the partial products can be added serially, reducing the design complexity. However, in high-speed design, the Wallace tree construction method is usually used to add the partial products in a tree-like fashion in order to produce two rows of partial products that can be added in the last stage. Although fast, since its critical path delay is proportional to the logarithm of the number of bits in the multiplier. In the last stage, the two-row outputs of the tree are added using any high-speed adder such as carry save adder to generate the output result.

3 HALF ADDER

Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit (S) and carry bit (C) as the output. If A and B are the input bits, then sum bit (S) is the X-OR of A and B and the carry bit (C) will be the AND of A and B. From this it is clear that a half adder circuit can be easily constructed using one X-OR gate and one AND gate.

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig.1. Truth table of half adder

Fig. 1 shows the truth table of half adder circuit which shows that the sum is 1 when exactly one of the two inputs is one otherwise zero and carry is 1 when both the inputs are 1.

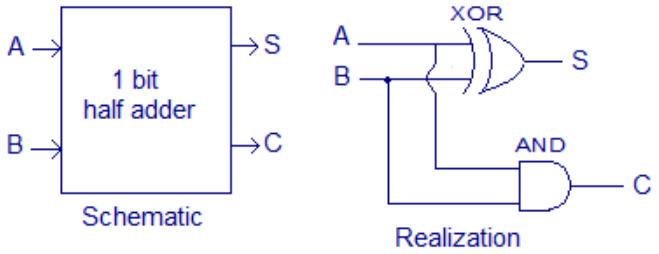

Fig. 2 Schematic and realization of half adder

Fig. 2 shows the schematic of half adder and the logic circuit that shows how to realize half adder.

4 FULL ADDER

A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full adder adds three one-bit numbers, often written as A,B, and C_{in}; A and B are the operands, and C_{in} is a bit carried in from the next less significant stage.

Inputs			Outputs	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Fig. 3 Truth table of full adder

Fig. 3 shows the truth table of full adder circuit in which there are three inputs A,B and C_{in} and two outputs C_{out} and Sum (S).

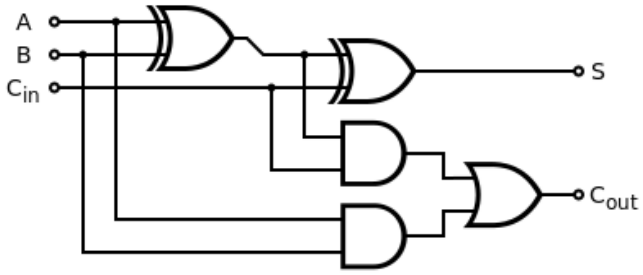

Fig. 4 Logic diagram of full adder

Fig. 4 shows the basic logic involves in a full adder. Two XOR gates, two AND gates and one OR gate is used.

5 RIPPLE CARRY ADDER

A simple ripple carry adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the carry output from each full adder connected to the carry input of the next full adder in the chain. Fig 5 shows the interconnection of four full adder (FA) circuits to provide a 4-bit ripple carry adder. Notice that from Fig 5 the input is from the right side because the first cell traditionally represents the least significant bit (LSB). Bits a_0 and b_0 in the figure represent the least significant bits of the numbers to be added. The sum output is represented by the s_0-s_3. The main problem with this type of adder is the delays needed to produce the carry out signal and the most significant bits. These delays increase with the increase in the number of bits to be added.

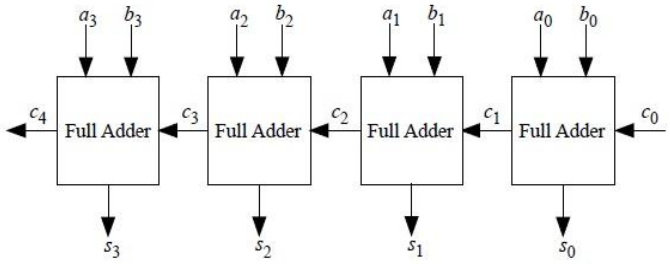

Fig. 5 4-bit Full Adder

6 CARRY LOOKAHEAD ADDER (CLA)

The carry look ahead adder (CLA) solves the carry delay problem by calculating the carry signals in advance, based on the input signals. It is based on the fact that a carry signal will be generated in two cases: (1) when both a_i and b_i are 1, or (2) when one of the two bits is 1 and the carry-in is 1. Thus, one can write,

$c_{i+1} = a_i.b_i + (a_i \oplus b_i).c_i$

$s_i = (a_i \oplus b_i). \oplus c_i$

The above two equations can be written in terms of two new signals P_i and G_i , which are shown as:

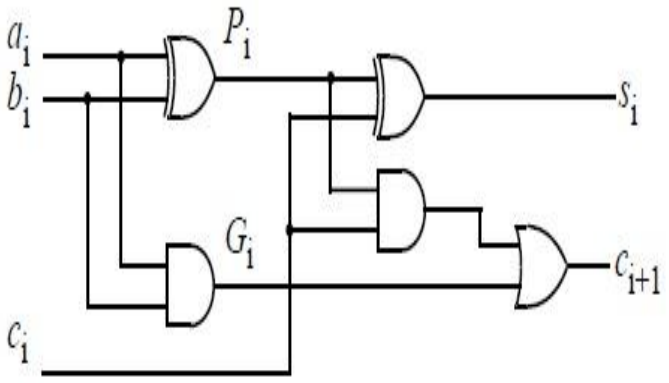

Fig. 6 Full adder at stage i with P_i and G_i shown

$c_{i+1} = G_i + P_i.c_i$
$s_i = P_i \oplus c_i$
where
$G_i = a_i.b_i$
$P_i = a_i \oplus b_i$

G_i and P_i are called the carry generate and carry propagate terms, respectively. Notice that the generate and propagate terms only depends on the input bits and thus will be valid after one and two gate delay, respectively. If one uses the above expression to calculate the carry signals, one does not need to wait for the carry to ripple through all the previous stages to find its proper value.

7 WALLACE TREE MULTIPLIER

Wallace tree multiplier consists of three step process, in the first step, the bit product terms are formed after the multiplication of the bits of multiplicand and multiplier, in second step, the bit product matrix is reduced to lower number of rows using half and full adders, this process continues till the last addition remains, in the final step, final addition is done using adders to obtain the result [1].

7.1 4x4 Wallace Tree Multiplier

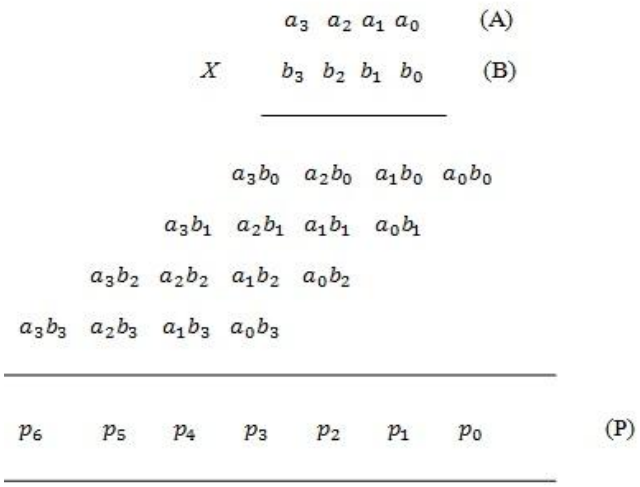

Fig. 7 Multiplication of 4x4 numbers

Fig 7 shows the multiplication of two 4-bit numbers. The numbers are denoted by A and B where a_0 , a_1 , a_2 , a_3 represents the bits of multiplicand A with a_0 as its least significant bit and a_3 as its most significant bit and b_0 , b_1 , b_2 , b_3 represents the bits of multiplier B with b_0 as its least significant bit and b_3 as its most significant bit. The product of the two 4-bit numbers is denoted by P which is of 8-bit with p_0 as the least significant bit and p_7 as the most significant bit. Fig 7 shows the basic multiplication of two numbers and thus producing the result, P. Now the use of half adders and full adders is explained in next Fig. 8.

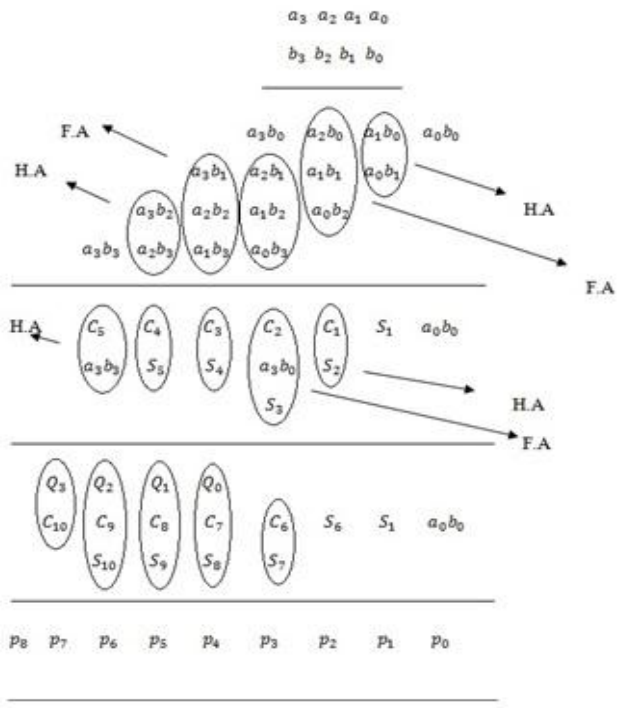

Fig. 8 Multiplication of two 4-bit numbers showing the adders used for addition of intermediate terms

Fig. 8 shows the multiplication of two numbers A and B as explained in Fig. 7 and producing its result as P. Fig. 8 explains the method of addition of different intermediate terms. The different intermediate terms formed after the multiplication of two 4-bit numbers are a_3b_3, a_2b_3 ,a_1b_3 ,a_0b_3 ,a_3b_2 ,a_2b_2 , a_1b_2 , a_0b_2 , a_3b_1 ,a_2b_1 ,a_1b_1 , a_0b_1 , a_3b_0 ,a_2b_0 , a_1b_0 ,a_0b_0. Two intermediate terms in one column are added using a half adder and more than two terms in one column are added using full adder as explained in fig 8. The sum obtained after each addition is denoted by $S_{i,}$ where i varies from 1 to 10. Similarly carries are denoted by $C_{j,}$ where j varies from 1 to 10 and $Q_{k,}$ denotes next carries, where k varies from 0 to 3.

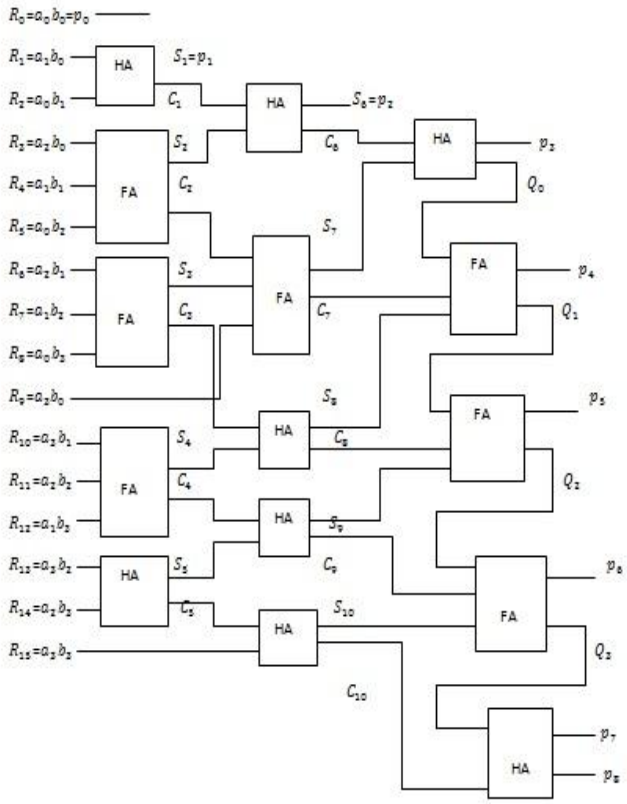

Fig. 9 Structural representation of 4x4 multiplier

Fig. 9 shows the structural representation of 4x4 multiplier using half adders and full adders for the addition of intermediate terms formed after the multiplication of two numbers. Finally, the product output is shown, showing each bit of the product obtained.

R_0 to R_{15} denotes the various product terms obtained at the first stage of multiplication. Product term a_0b_0 is represented by R_0. Similarly the other product terms are represented by different notations from R_1 to R_{15}. The first bit p_0 of the product P is obtained by the first product term a_0b_0 which is denoted by R_0. R_1 and R_2 then become the inputs of the half adder to give two outputs, sum S_1 and carry C_1. Sum S_1 is nothing but the next bit of the product P which is denoted by p_1. R_3, R_4, and R_5 become the input bits of the full adder to give outputs as sum S_2 and carry C_2. Previous carry C_1 and sum S_2 becomes the input bits of next half adder to produce two outputs sum S_6 and carry C_6. Sum S_6 is the third bit of the product P which is named as p_2. The remaining bits of the product P i.e. p_3, p_4, p_5, p_6, p_7, p_8 respectively are obtained in the same way as explained above.

7.2 8x8 Wallace Tree Multiplier

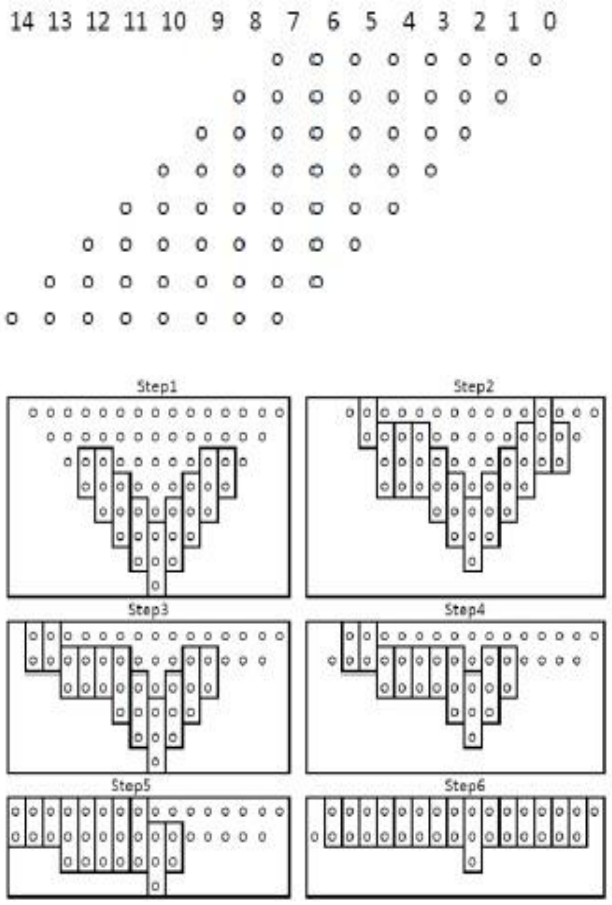

Fig. 10 Logic used in 8 bit Wallace Tree Multiplier

Fig. 10 shows the basic architecture and the steps involved in 8x8 wallace tree multiplier. The procedure of multiplication is same as that of 4x4 wallace multiplier. In the Wallace Tree method, the circuit is quite irregular although the speed of operation is high.

8 SIMULATION AND RESULTS

The VHDL simulation of the two multiplier is presented in this section. For simulation Modelsim tool is used. The waveform of the results are shown. Fig. 11 shows the multiplication of 4x4 multiplier and Fig. 12 shows the multiplication of 8x8 multiplier.

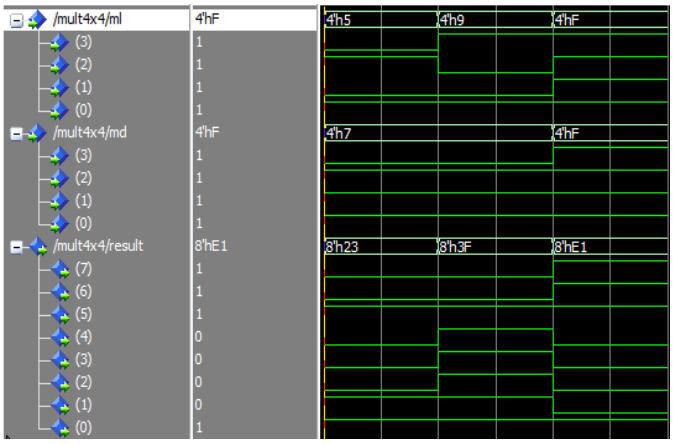

Fig. 11 Simulation Waveform for 4x4 multiplier

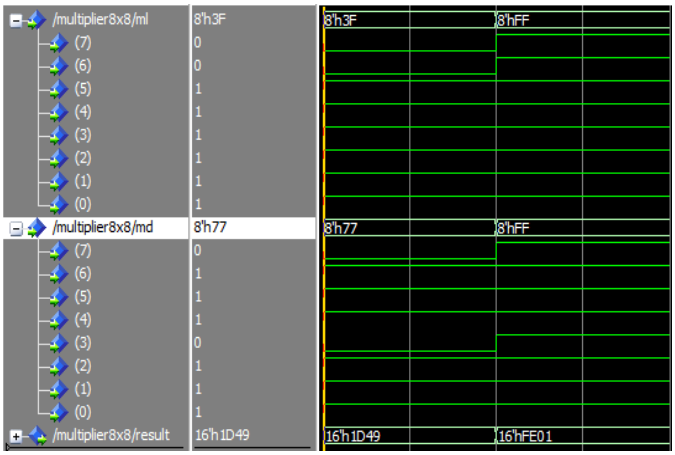

Fig. 12 Simulation Waveform for 8x8 multiplier

The table shown below describes the values taken for multipliers and multiplicands thereby obtaining the product result.

Table 1 : Example of 4x4

Multiplier(Hex)	Multiplicand(Hex)	Product(Hex)
5	7	23
9	7	3F
F	F	E1

Table 2 : Example of 8x8

Multiplier(Hex)	Multiplicand(Hex)	Product(Hex)
3F	77	1D49
FF	FF	FE01

9 CONCLUSION AND FUTURE WORK

This paper presents two different nxn multipliers that are modeled using VHDL. This work is performed on 4-bit and 8-bit unsigned data. Therefore it can be extended for higher value of n.

REFERENCES

[1] Whitney J. Townsend, Earl E. Swartzlander, Jr., and Jacob A. Abraham, "*A comparison of Dadda and Wallace multiplier delays*",The University of Texas at Austin, TX 78712.

[2] C. Jaya Kumar, R.Saravanan,"*VLSI Design for Low Power Multiplier using Full Adder*", European Journal of Scientific Research, ISSN 1450-216X ,Vol.72, No.1 (2012), pp. 5-16.

[3] R. Naveen,K. Thanushkodi,C. Saranya,"*Reduction of Static Power Dissipation in Wallace Tree Multiplier*", European Journal of Scientific Research, ISSN 1450-216X, Vol.84, No.4 (2012), pp.522-531

[4] Priyanka Gautam,R.S.Meena,"*Designing of 4X4 Wallace Tree Multiplier Using 8T Higher Order Compressor*", International Journal of Advanced Technology & Engineering Research (IJATER).

[5] Naveen Kr. Gahlan, Prabhat Shukla, Jasbir Kaur,"*Implementation of Wallace tree Multiplier Using Compressor*",Vol 3(3),1194-1199.

[6] Dakupati.Ravi Sankar, Shaik Ashraf Ali,"*Design of Wallace Tree Multiplier by Sklansky Adder*",International Journal of Engineering Research and Applications (IJERA), ISSN:2448-9622,Vol.3,Issue 1,pp.1036-1040.