VLSI Implementation of Adders for High Speed ALU

Prashant Gurjar
UG Scholar,Dept. EC,
GGITM, Bhopal
India

Rashmi Solanki
UG Scholar,Dept. EC,
GGITM, Bhopal
India

Pooja Kansliwal
UG Scholar,Dept. EC,
GGITM, Bhopal
India

Mahendra Vucha
Asst. Prof., Dept. EC,
GGITM, Bhopal,
India

ABSTRACT
This paper is primarily deals the construction of high speed adder circuit using Hardware Description Language (HDL) in the platform Xilinx ISE 9.2i and implement them on Field Programmable Gate Arrays (FPGAs) to analyze the design parameters. The motivation behind this investigation is that an adder is a very basic building block of Arithmetic Logic Unit (ALU) and would be a limiting factor in performance of Central Processing Unit (CPU).

Design of a high speed single core processor is the future goal of this paper. Single core processor would have many advantages over a multiple-core approach. Task execution on a single core is a well understood process, while execution on many cores is a problem that has not yet been solved. There are also computational tasks which parallelize very badly, where a single high clock rate processor would suit them very well. Such a high speed processor needs certain components that should support high speed. The two main components of processors are the ALU and the register file. The one of the critical path within an ALU may be the carry-chain in addition operation.

In this research article, we have simulated and synthesized the various adders like full adder, ripple carry adder, carry-look ahead adder, carry-skip adder and carry –select adder by using VHDL and Xilinx ISE 9.2i. The simulated results are verified and the functionality of high speed adders and the parameters like area and speed is analyzed. Finally this paper concludes that the carry-skip adder is the more efficient in speed and area consumption.

Keywords:
High Speed Adder, Field Programmable Gate Array, Carry Skip Adder, Carry Select Adder.

1. INTRODUCTION
Digital computer ALU is an aspect of logic design with the objective of developing appropriate algorithms in order to achieve an efficient utilization of the available hardware. The hardware can only perform a relatively simple and primitive set of Boolean operations and the arithmetic operations are based on a hierarchy of operations that are built by using algorithms against the hardware. Since, ultimately, speed, power and utilization of ALU are the most often used measures of the efficiency of an algorithm.

1.1 What Is an Adder?
In digital electronics, adder is a digital circuit that performs addition of two numbers. As described in [2], many computers and other kinds of processors, adders are used not only in the ALU(s), but also in other parts of the processor, where they are used to calculate addresses, table indices, and many more.

1.2 Concept of Adders
Consider two binary variables x and y. As shown in [4], the binary sum is denoted by x + y, such that

0+0 = 0   0+1 = 1   1+0 = 1   1+1 = 10

Here, the result in the last case is a binary 10 (i.e., 2 in base 10). The sum of two numbers can be out of the range of the digits in binary set. This, of course, is the origin of the concept of a carry out. In the binary sum 1+1, the result 10 is viewed as a 0 with a 1 shifted to the left to give a “carry-out is 1”.

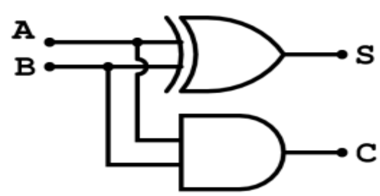
1.3 Half Adder


Figure 1. Half adder

A Half Adder (HA) is a logical circuit that performs an addition operation on two binary digits. The half adder produces a sum and a carry value which are both binary digits. The logic diagram of HA is shown in figure 1.

A HA adds two one-bit binary numbers *A* and *B*. It has two outputs, *S* and *C* (the value C theoretically carried on to the next addition).The simplest half-adder design, shown in figure 1, incorporates an XOR gate for *S* and an AND gate for *C*. The Boolean equation and Truth table of half adder is shown bellow in Table 1.

S = A XOR B   …… (i) C = A AND B …. (ii)

Table 1. Truth table for half adder

Input		Output	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

1.4 Full Adder

A Full Adder (FA) is a logical circuit that performs an addition operation on three binary digits. The full adder produces a sum and a carry value, which are both binary digits. The logical diagram of full adder is shown in figure 2.

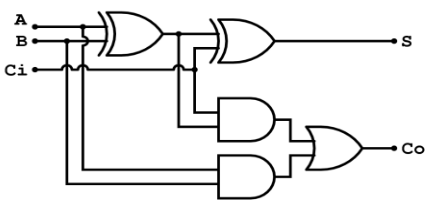


Figure 2. Full adder

A FA 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_i here *A*, *B* are the operands, and C_i is a bit carried in (in theory from a past addition by [6]). The circuit produces a two-bit output sum typically represented by the signals C_o (Carry) and *S* (Sum). The Boolean equation and truth table are shown bellow.

S = A **XOR** B **XOR** C_i ……(iii)

Co = (A **AND** B) **OR** (B **AND** C_i) **OR** (C_i **AND** A) ……(iv)

Table 2. Truth table for full adder

A	B	C_{in}	C_{out}	Sum
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

A FA can be constructed by cascading of two HA. The *A* and *B* are connected to the input of first HA and the sum of first HA is connected as one input along with C_i to second HA and it give SUM output. The logical OR of first and second HAs carry outputs a gives CARRY output of FA shown in [3].

2. COMPLEX ADDERS

The reference to eve of adding single bits, let's extend it to adding binary words. In general, adding two n-bit words yields an n-bit sum and a carry-out bit C_n. The carry is carried from lower bit adder to higher bit adder. Based on carry transfer from LSB to MSB, the adders are classified.

2.1 Ripple Carry Adder

It is possible to create a logical circuit using multiple full adders to add *N*-bit numbers. Each full adder inputs a carry C_{in} which is the C_{out} of the previous adder. This kind of adder is a Ripple Carry Adder (RCA) in [9], since each carry bit "ripples" to the next full adder. Note that the first (and only the first) full adder may be replaced by a half adder. The layout of a ripple carry adder is simple, which allows fast design time. However, the ripple carry adder is relatively slow, since each full adder must wait for the carry bit which is coming from the previous full adder. The RCA is shown in figure. 3.

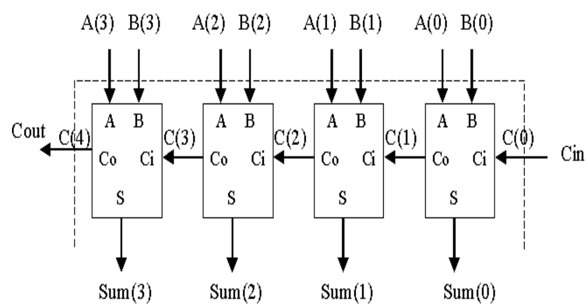


Figure 3. Ripple carry adder

2.2 Carry - Look Ahead Adder

Carry- Lookahead Adder (CLA) is designed to overcome the latency introduced by the repelling effect of the carry bits in RCA. The CLA improves speed by reducing the amount of time required to determine carry bits. Carry lookahead logic uses the concepts of *generating (G)* and *propagating (P)* carries. Its work is based on two signals called *P* and *G* for each bit position. The P and G are shown bellow.

$C_{i+1} = G_i + P_i.C_i$ Here, $G_i = A_i.B_i$ and $P_i = (A_i \oplus B_i)$

$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i$.

The Si and C_{i+1} represent the sum and carry from i^{th} full adder respectively. The carry-lookahead adder can be broken up in two modules: (1) The Partial Full Adder, PFA, which generates Si, Pi and Gi. (2) The Carry Look-Ahead Logic, which generates the carry-out bits. The structure of CLA for 4-bit adder is shown in figure 4.

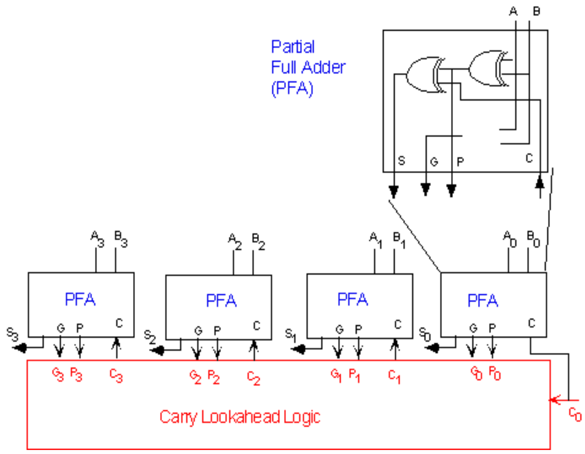


Figure 4: Carry Lookahead adder

3. IMPLEMENTATION OF HSA

The alternate approaches for designing High Speed Adders (HSA) have been designed in the literature [1], [2], [3], [4]. All of them have the objective of decreasing the computation time and different tradeoffs. This paper examines few of them bellow.

3.1 Carry – Skip Adder

A carry-skip adder is designed to speed up a wide adder by adding the propagation of carry bit around a portion of the entire adder. The idea is illustrated in figure 5 for the case of a 4 bit adder. The carry-in bit is designated as C_i and the adder itself produces a carry-out bit of C_{i+4}. The carry skip circuitry consists of two logic gates. The AND gate accepts the carry-in bit and compares it to the group propagate signals.

$$P_{(i,i+3)} = P_{i+3}.\ P_{i+2}.P_{i+1}.P_i$$

Using the individual propagate values, the output from the AND gate is ORed with C_{i+4} to produce a stage output of

$$Carry = C_{i+4} + P_{(i,i+3)}.\ C_i$$

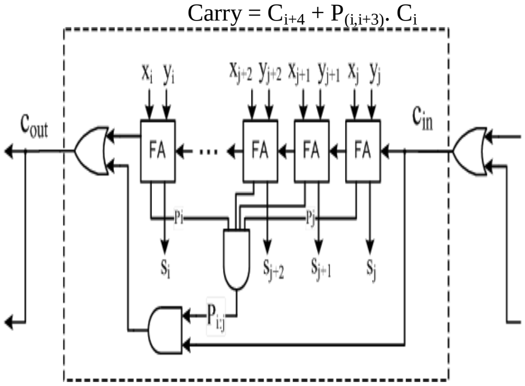


Figure 5. Carry skip adder

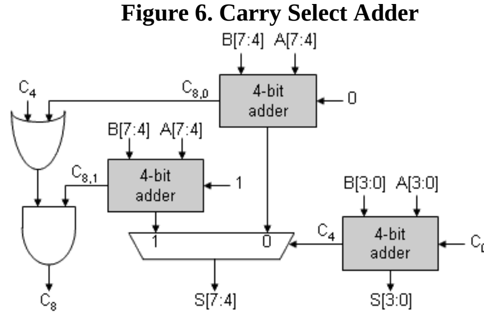
As shown in the figure 5, if $P_{(i,\ i+3)} = 0$, then the carry-out of the group is determined by the value of C_{i+4}. However, if $P_{(i,i+3)} = 1$, then the carry-in bit is $C_i= 1$, then the group carry-in is automatically send to the next group of adders. The name "carry-skip" is due to the fact that if the condition $P_{(i,i+3)}.\ C_i$ is true and then the carry-in bit skips the block entirely.

3.2 Carry – Select Adder

Carry Select Adders (CSA) use multiple narrow adders to create fast wide adders. Consider the addition of two n bit numbers with a = $a_{n-1}\ldots.a_0$, and b = $b_{n-1}\ldots.b_0$. At the bit level the adder delay increases from the least significant 0^{th} position upward, with the $(n-1)^{th}$ requiring the most complex logic. A carry select adder breaks the addition problem into smaller groups. A carry-select adder provides two separate adders for the upper words, one for each possibility. A multiplexer (MUX) is then used to select the valid result. The figure 6 shows the block diagram of CSA.

As a concrete example, consider an 8-bit adder that is split into two 4-bit groups. The lower order bits $a_3\ a_2\ a_1\ a_0$ and $b_3\ b_2\ b_1\ b_0$ are fed into the 4-bit adder to produce the sum bits $S_3\ S_2\ S_1 S_0$ and a carry-out bit C_4 as shown.

Figure 6. Carry Select Adder



The higher order bits $a_7\ a_6\ a_5\ a_4$ and $b_7\ b_6\ b_5\ b_4$ are used as two 4-bit adders. Adder calculates the sum with a carry in of C=0, while the other adder does the same only it has a carry-in value of C=1. Both sets of results are used as inputs to an array of 2:1 MUXs. The carry bit C_4 from the first adder is used as the select signal to MUX. If $C_4 = 0$, then the result of C=0 adder are sent to the output, while a value of C_4=1 selects the result of C=1 adder for $S_7\ S_6\ S_5\ S_4$. The carry-out bit C_8 is also selected by the MUX array. The design speeds up the addition of the word by allowing the upper and lower portions of the sum to be calculated simultaneously. The price paid is that it requires an additional word adder, a set of multiplexers and associated interconnect wiring. The design becomes viable if speed is more important than area consumption.

3.3 Carry – Save Adder

Carry – save adder are based on the idea that a full adder really has three inputs and produces two outputs as shown. While it is usually associates the third input with a carry in, it could equally well be used as a "regular" value. The full adder is used as 3:2 reduction network, where it starts with bits from 3 bits words, adds them and then has an output that is 2-bits wide. An n-bit carry save adder can be build by using n separate adders. The name 'carry-save' arises from the fact that we save the carry out words instead of using it immediately to calculate the final sum. Carry-save adders are useful in situations where we need to add more than two numbers. Since the design automatically avoids the delay in the carry-out bits.

4. RESULT AND DISCUSSIONS

The design of high speed adders is necessary to increase the computation speed of ALU and it supports to the design of high speed processor. In this research, the hardware implementation of various adders has been done to analyze the speed and area. The RTL code is written in VHDL, Xilinx ISE 9.2i is used to simulate and synthesize the design. The simulation helps to verify the design and the synthesis report gives the speed and area of the design. Finally, the VLSI implemented designs are targeted to the FPGA device ***xc3s500e-5-ft256*** and captured the real time speed and area of the designs. The comparison table is shown in bellow. The table 3, 4 and Figure 8 shows synthesis report of 16 –bit adder, synthesis report of 8 – bit adder and speed comparison of various adders respectively. The figure 8 represents the comparison chart by taking speed in MHz on Y axis and various adders on X axis

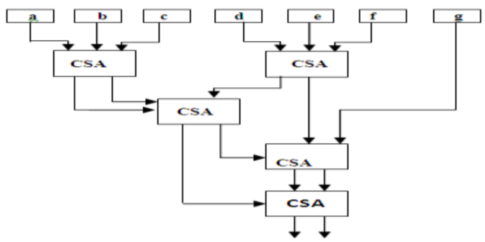


Figure 7. Carry save Adder

4.1 COMPARISON OF ADDERS

4.1.1 16-bit adders:

Table 3: synthesis report of 16-bit adders

S.No.	Parameter	Ripple carry	Carry-look ahead	Carry-skip	Carry-select
1.	**XOR (1-bit)**	16	32	32	24
2.	**No. of Slices**	18/960	18/960	21/960	22/768
3.	**Levels of Logic**	18	18	15	16
4.	**Processing Time**	3.77 s	3.555 s	4.67 s	3.66 s
5.	**Memory Usage**	140796 Kb	140796 Kb	141820 Kb	134356 Kb
6	**Logic Delay**	14.067 ns	14.067 ns	11.316 ns	12 ns
	Route Delay	7.623 ns	7.623 ns	5.326 ns	11.163 ns
	Total Delay	21.69 ns	21.69 ns	16.642 ns	23.163 ns

4.1.2 8-bit adders:

Table 4: Synthesis report of 8-bit adders

S.No.	Parameter	Ripple carry	Carry-look ahead	Carry-skip	Carry-select
1.	**XOR (1-bit)**	8	16	16	12
2.	**No. of Slices**	9/960	9/960	11/960	11/768
3.	**Levels of Logic**	10	10	9	9
4.	**Processing Time**	3.453 s	3.44 s	3.44 s	2.945 s
5.	**Memory Usage**	139772 Kb	139772 Kb	140796 Kb	134356 Kb
6	**Logic Delay**	9.171 ns	9.171 ns	8.254 ns	8.977 ns
	Route Delay	4.032 ns	4.032 ns	3.286 ns	6.945 ns
	Total Delay	13.203 ns	13.203 ns	11.54 ns	15.922 ns

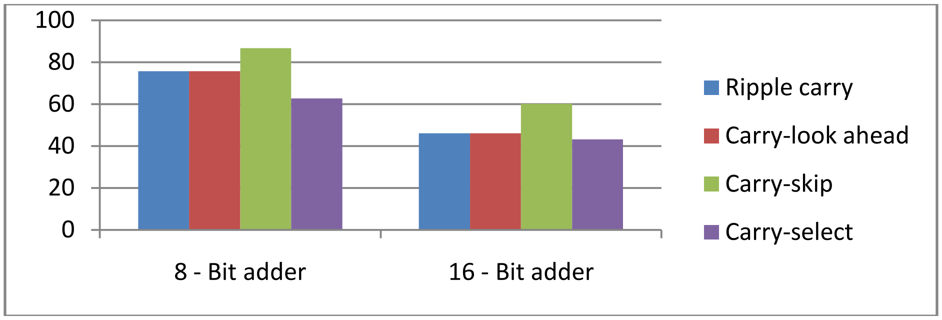


Figure 8. Speed (in MHz) comparison chart of adders

5. CONCLUSION

The research article describes about the hardware implementation of high speed adders. In this paper, the various adders like full adder, ripple carry adder, carry-look ahead adder, carry-skip adder and carry –select adder have been simulated and synthesized on Xilinx ISE 9.2i platform and their parameters are captured. Finally, the captured parameters like speed and area are compared for 8 –bit and 16-bit adders. From the table 5, this paper concludes that the carry-skip adder is the efficient adder in speed and area consumption. The analysis in table 5 for 16 – bit adder is shown bellow.

Table 5: Speed & Area analysis for 16 – bit adder

Adder	Speed (MHz)	Area (XOR gate)
Ripple Carry Adder	46.1	16
Carry-look ahead adder	46.1	32
Carry-skip adder	60.1	32
Carry-select adder	43.2	24

6. REFERENCES

[1] Bruce Shriver and Bennett Smith, The Anatomy of a High- Performance Microprocessor, IEEE Computer society Press, Los Alamitos, CA, 1998.
[2] James M. Feldman and Charles T. Retter, Computer Architecture, McGraw-Hill, New York, 1994.
[3] Ken Martin, Digital Integrated Circuit Design, Oxford University Press, New York, 2000.
[4] Behrooz Parhami, Computer Arithmetic, Oxford University Press, New York, 2000. A comprehensive, in depth treatment of the subject.
[5] David A. Patterson and John L. Hennessy, Computer Organization & Design, 2nd edition, Morgan-Koufmann Publishers, San Fransisco, 1998.
[6] Jan M.Rabaey, Digital Integrated Circuits, Prentice Hall, Upper Saddle River, NJ, 1996.
[7] Abdellatif Bellaouar and Mohamed I.Elmasry, Low-power Digital VLSI Design, Kluwer Academic Publishers, Norwell, MA, 1995.
[8] William Stallings, Computer Design and Architecture, 4th Edition, Prentice Hall, Upper Saddle River, NJ, 1996.
[9] John P. Uyemura, CMOS Logic Circuit Design, Kluwer Academic Publishers, Norwell, MA, 1999.
[10] Neil H.E. Weste and Kamran Eshraghian, Principles of CMOS VLSI Design, 2nd edition, Addision-Wesley, Reading, MA, 1993.
[11] Wayne Wolf, Modern VLSI Design, 2nd edition, Prentice Hall PTR, Upper Saddle River, NJ, 1998.

7. ACKNOWLEDGEMENT

The authors would like to express their gratitude to the **Dr. P.S. Venkataramu**, Principal, GGITM, who have given support and **Ashutosh Agrawal**, our classmate and friend who never seemed to lose faith that the paper would be eventually Completed.

8. AUTHORS BIOGRAPHY

Prashant Gurjar working for his B.E degree at Gyan Ganga Institute of Technology and Management, Dept. of Electronics and communication Engineering, Bhopal (M. P), India.. His areas of interest are VLSI.

Rashmi Solanki working for her B.E degree at Gyan Ganga Institute of Technology and Management, Dept. of Electronics and communication Engineering, Bhopal (M. P), India.. Her areas of interest are VLSI.

Pooja Kansliwal working for her B.E degree at Gyan Ganga Institute of Technology and Management, Dept. of Electronics and communication Engineering, Bhopal (M. P), India.. Her areas of interest are VLSI.

Mahendra Vucha received his B. Tech in Electronics & Communication Engineering from JNTU, Hyderabad in 2007 and M. Tech degree in VLSI and Embedded System Design from MANIT, Bhopal in 2009. He is currently working for his Ph. D degree at MANIT and also working as Asst. Prof in Gyan Ganga Institute of Tech & Mgmt, Dept. of Electronics and Communication Engineering, Bhopal (M.P), India. His areas of interest are Hardware Software Co-Design, Analog Circuit design, Digital System Design and Embedded System Design.