International Journal of Research & Development Organisation

ISSN-4567-7860

Journal Of Electrical And Electronics Engineering

Comparative Analysis of Traditional Adder and Pipelined RCA

T.Thamarai manalan[1], G.Jagadeeswaran [2] A.Shanmugapriya[3], M.Ishwarya Niranjana[4]

[1] Assistant Professor (ECE), Sri Eshwar College of Engineering, Coimbatore.
[2]PG Scholar(Applied Electronics), Sri Eshwar College of Engineering, Coimbatore.
[3] Assistant Professor (ECE), Pollachi Institute of Engineering and Technology, Pollachi.
[4]Assistant Professor(ECE),. Pollachi Institute of Engineering and Technology, Pollachi.

[1]jagadeeswaran046@gmail.com
[3]priyasamy30@gmail.com
[4]ishu.niranjana@gmail.com

Abstract—Addition is the basic arithmetic operation in any processing system. A fast and accurate operation of digital system is greatly influenced in performance of resident adders. Hence improving the performance of adder would greatly enhance the execution of circuit operation. In Conventional adders results in high power dissipation and increased complexity when number of bits are increased. Pipelining is an emerging technique used to obtain high throughput over conventional design. The whole task can be divided into number of subtasks to speed up the operation. In this paper traditional adder is compared with pipeline based parallel RCA and it has been proven that the efficiency of pipelined adder is much higher. The results are simulated using Xilinx 12.1 and various parameters are compared and tabulated.

Keywords- Pipelining, Full Adder, Binary Adder.

1. INTRODUCTION

Addition is one of the basic arithmetic operations in binary arithmetic by adding two binary numbers. An adder should be designed to meet the requirements of many modern devices: small chip area and high circuit speed. Since high speed computer arithmetic units such as adders, multipliers and fast dividers that dominate the power dissipation and device complexity is dramatically increasing. Now-a-days, low power design has come to the forefront in addition to the two traditional issues mentioned above. Especially, the adder is critical in the aim to reduce overall power consumption since they are used for implementation of multipliers.

1.1 Basic Adder Unit

The most basic arithmetic operation is the addition of two binary digits, i.e. *bits*. A combinational circuit that adds two bits, according the scheme outlined below, is called a half adder. A full adder is one that adds three bits, the third produced from a previous addition operation. One way of implementing a full adder is to utilizes two half adders in its implementation. The full adder is the basic unit of addition employed in all the adders.

1.1.1 Half Adder

A half adder is used to add two binary digits together, **A** and **B**. It produces **S**, the sum of A and B, and the corresponding carry out **Co**. Although by itself, a half adder is not extremely useful, it can be used as a building block for larger adding circuits (FA). One possible implementation is using two AND gates, two inverters, and an OR gate instead of a XOR gate as shown in Fig. 1.The equations of sum and carry are,

$$S = A \oplus B = \overline{A}B + \overline{B}A \qquad (1)$$

$$C_0 = AB \qquad (2)$$

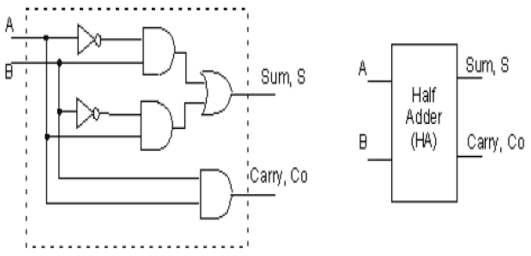


Fig.1 Half Adder Module

Truth Table:

A	B	S	C_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

1.1.2 Full Adder

A full adder is a combinational circuit that performs the arithmetic sum of three bits: A, B and a carry in, C, from a previous addition, Fig. 2a. Also, as in the case of the half adder, the full adder produces the corresponding sum, S, and a carry out Co. As mentioned previously a full adder maybe designed by two half adders in series as shown below in Figure 2b. The sum of A and B are fed to a second half adder, which then adds it to the carry in C (from a previous addition operation) to generate the final sum S. The carry out, Co, is the result of an OR operation taken from the carry outs of both half adders. There are a variety of adders in the literature both at the gate level and transistor level each giving different performances. The equations of sum and carry are,

$$S = C \oplus (A \oplus B) \qquad (3)$$

$$C_0 = AB + C(A \oplus B) \qquad (4)$$

Truth Table:

A	B	C	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

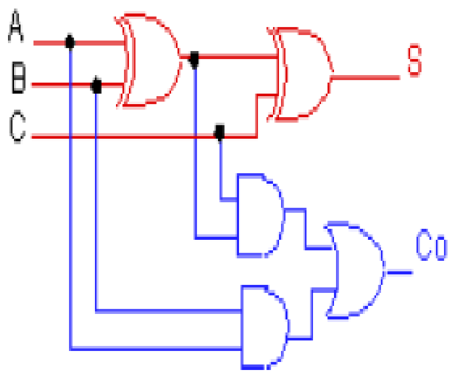


Fig.2 Full Adder Module

2. PIPELINE ADDERS

Pipelining is a technique that shortens the circuit delay by placing a register in a combinational logic path to break the critical path. Pipelining has the advantage to get high throughput of a circuit because the register to register delay is the delay path that sets the clock rate. For the adder structures discussed in this thesis, except for the carry save adder, the critical path is always the carry of the adders, that is, from carry-in to carry-out. For instance, in Fig.3.(a), there is an execution unit that is composed of two logic blocks connected in series. By inserting registers (Fig.3.(b)) between them, the two series blocks are isolated so that the delay for each logic block is 1/2 of the total delay of the previous execution unit.

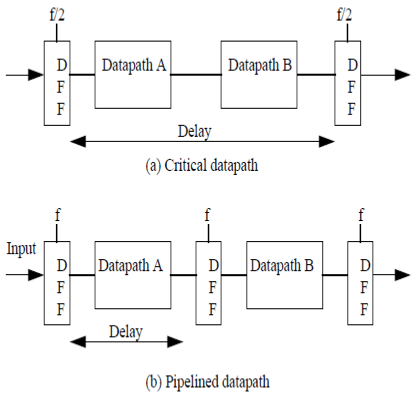


Fig.3 Pipelining

2.1 Ripple Carry Adder

The ripple carry adder is constructed by cascading full adders (FA) blocks in series. One full adder is responsible for the addition of two binary digits at any stage of the ripple carry. The carryout of one stage is fed directly to the carry-in of the next stage. A number of full adders may be added to the ripple carry adder or ripple carry adders of different sizes may be cascaded in order to accommodate binary vector strings of larger sizes. For an n-bit parallel adder, it requires n computational elements (FA). Fig.4 shows an example of a parallel adder: a 4-bit ripple-carry adder. It is composed of four full adders. The augends' bits of x are added to the addend bits of y respectfully of their binary position. Each bit addition creates a sum and a carry out. The carry out is then transmitted to the carry in of the next higher-order bit. The final result creates a sum of four bits plus a carry out (c4).

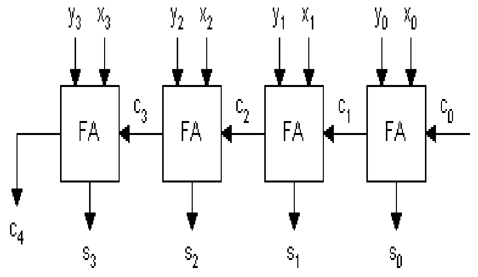
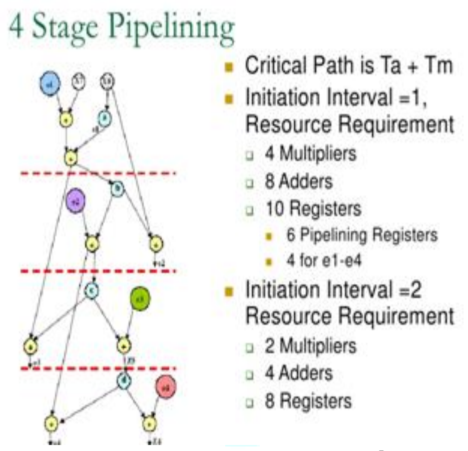


Fig.4 Ripple Carry Adder (RCA)



4 Stage Pipelining

- Critical Path is Ta + Tm
- Initiation Interval =1, Resource Requirement
  - 4 Multipliers
  - 8 Adders
  - 10 Registers
    - 6 Pipelining Registers
    - 4 for e1-e4
- Initiation Interval =2 Resource Requirement
  - 2 Multipliers
  - 4 Adders
  - 8 Registers

Fig.5 4 Stage Pipelining

2.1.1 Classical RCA Pipelining

The RCA delay is proportional to the addition size. It has three components. First, the LUT delay, δ_{LUT}, used to precompute the carry multiplexer select signal. Then there is a worst-case delay of $(w-1)\delta_{carry}$ for carry propagation. Finally, δ_{xor}, the delay of the xor gate used to compute the MSB sum bit. Fig.6 shows the 16-bit Pipelined adder designed using 4 bit RCA adders. In this the efficiency of addition is increased and pipelining reduces the delay in conventional design.

A tight frequency-driven pipelining is obtained by first determining the maximal addition size α in equation 1 for which the critical path delay is less than the target period T:

$$\alpha = 1 + \left[\frac{T - \delta_{LUT} - \delta_{XOR}}{\delta_{CARRY}} \right] \tag{5}$$

Next, the addition is split into k chunks of α bits (except the last chunk denoted by β, $\beta \le \alpha$) such that $w = (k-1)\alpha + \beta$. An instantiation of this architecture highlighting the previously discussed parameters is presented in Fig. 8(a) for k = 4. As k decreases, the number of registers used for synchronization decreases. When the critical path of the w bit addition is $\le$ T, no pipelining is required (k = 1) and the addition may be expressed as a simple + in VHDL

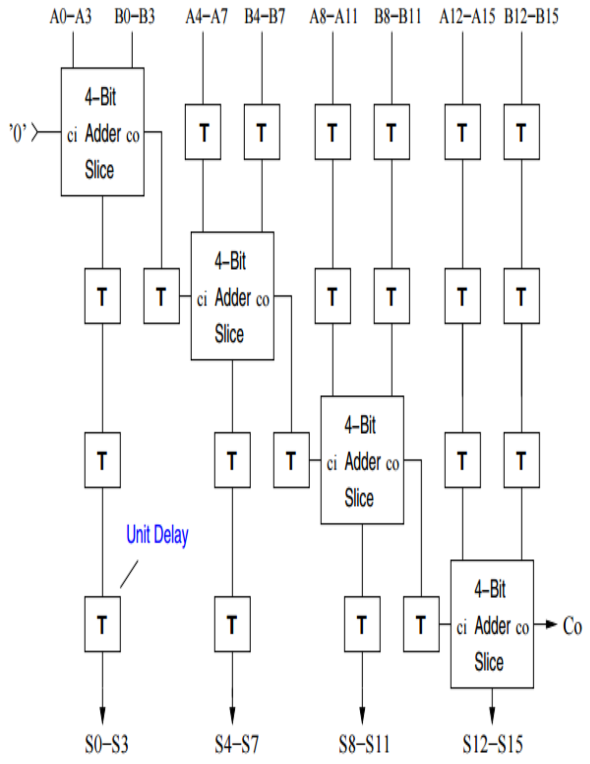
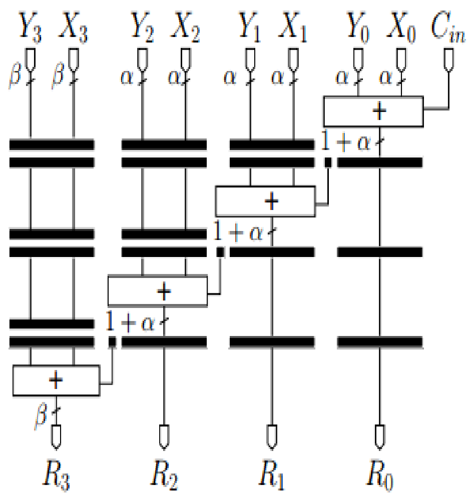
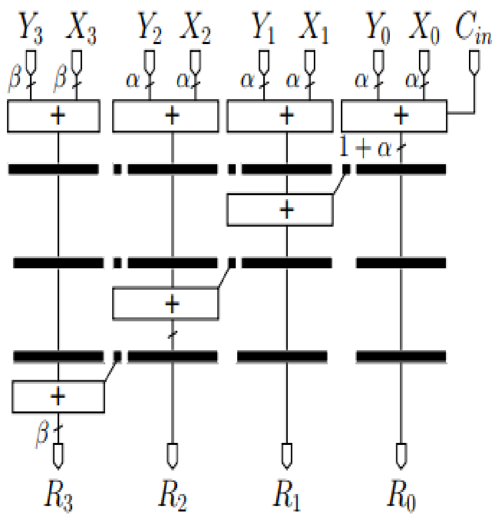


Fig.6 16 bit pipelined RCA adder


Y_3 X_3 Y_2 X_2 Y_1 X_1 Y_0 X_0 C_{in}
β β α α α α α α
$1+\alpha$
β
R_3 R_2 R_1 R_0

a) Classical RCA Pipelining


Y_3 X_3 Y_2 X_2 Y_1 X_1 Y_0 X_0 C_{in}
β β α α α α α α
$1+\alpha$
β
R_3 R_2 R_1 R_0

a) Novel RCA Pipelining

Fig. 7 RCA Pipelining

The classical pipelining technique requires a significant amount of registers for input synchronization. This number may be lowered by performing the chunk additions at the first pipeline level and then propagating these sums instead. When no SRL are allowed, the number of registers propagated above the diagonal will be approximately halved, and may still be packed in shift registers. An instantiation of this architecture for k = 4 is presented in Fig.7. Each adder on the addition diagonal takes as input an operand on α+1 bits and a 1-bit carry in and returns a α+1-bit wide result. This addition does not overflow, as the α + 1-bit input was the result of an addition of two α-bit numbers with a carry-in of 0.

The algorithm first determines the chunk size α. Next, two sums are computed for each pair of chunks: $X_i + Y_i$ and $X_i + Y_i + 1$. The final result R is a combination of the corresponding sub-sums and is found in a space of 2 k combinations. Selecting the appropriate sub-sum is done by using a carry-in bit. The novel idea in this algorithm is the use of the dedicated fast-carry chains to compute the carry-bits for the result selection. Actually, for each chunk, a pair (sum, carry-out) is computed for both possible values of the carry-in. We use the following notations to denote the concatenation of the subsums and their corresponding carry-out bits.

$$c_i^o s_i^o = X_i + Y_i \tag{6}$$

$$c_i^1 s_i^1 = X_i + Y_i + 1 \tag{7}$$

We denote by Ri the ith sub-result such that R = Rk−1 . . . R1R0. The value of Ri can be expressed in the following way knowing Si $_0$, Si $_1$ and c_{i-1}.

$$if\,(c_{i-1} = 0)\,then\;\; R_i \leftarrow S_I^0$$

$$else \quad R_i \leftarrow S_i^1 \tag{8}$$

The addition architectures presented so far make extensive use of the shift-registers available in the sliceM. However, this resource is getting rarer over the years. All the slices in a VirtexII-Pro device were similar to sliceM, but they were reduced to half the total number of slices for Virtex4 and Spartan3, and about a quarter in Virtex5 and Virtex6 devices (with higher density at the input of the DSP48E blocks). There is an ISE option that prevents using this resource.

It may therefore be relevant to be able to generate adders with this in view. Out of the presented architectures, the low-latency one will behave better when no shift registers are allowed. This is due to the fact that it requires fewer registers for synchronization. When k = 2, the alternative implementation behaves better than the classical one, as it propagates approximately half as many signals on the upper part of the addition diagonal.

3. RESULTS AND PERFORMANCE ANALYSIS

Pipelining improves throughput at the expense of latency, however, once the pipe is filled we can expect one data item per unit of time. Some of the conventional pipelined adder designs that have been reported include one that uses overlapped clocks in an effort to eliminate sources of overhead [5]. The 4-bit carry propagate adder employs a series of three registers to equalize the delays in adding the four bits and hasa three cycle latency [4].

Time borrowing is performed to shorten the critical path and the adder design has been realized in 0.18μm technology. A number of pipeline registers are introduced and in a similar fashion as in [4] several of these registers are inserted in series to equalize data arrival times at adder units. The gain in speed is achieved by clocking sub-circuits faster than would be possible with a ripple carry adder.

The conventional pipelined adder architectures achieve path delay equalization by inserting registers. In the instances where these registers are used for delay equalization no logic is used between the set of registers, the output of one register connects directly to the input of the next.

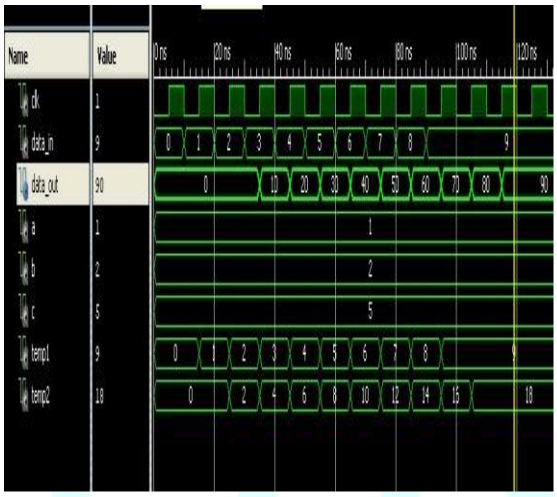
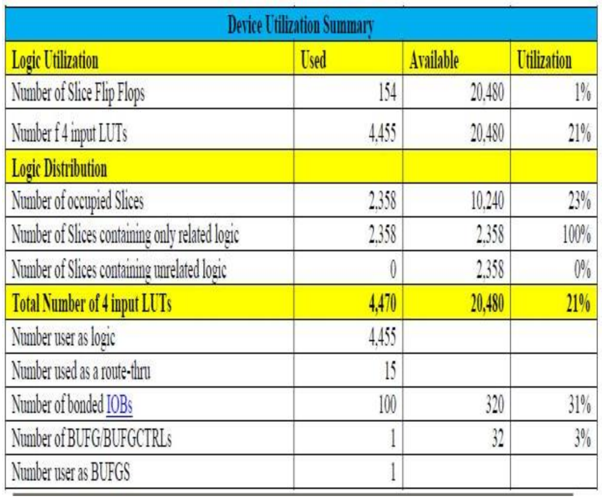


Fig .8 Simulation of Novel RCA



Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	154	20,480	1%
Number f 4 input LUTs	4,455	20,480	21%
Logic Distribution			
Number of occupied Slices	2,358	10,240	23%
Number of Slices containing only related logic	2,358	2,358	100%
Number of Slices containing unrelated logic	0	2,358	0%
Total Number of 4 input LUTs	**4,470**	**20,480**	**21%**
Number user as logic	4,455		
Number used as a route-thru	15		
Number of bonded IOBs	100	320	31%
Number of BUFG/BUFGCTRLs	1	32	3%
Number user as BUFGS	1		

Fig .9 Design summary of Novel RCA

Table 1 Performance comparison

Architecture	Results		
	LUTs	Registers	Slices
Traditional adder	5120	120	190
Classical RCA	4940	95	175
Novel RCA	4470	80	154

Conclusion

The traditional adder circuit and pipelined RCA circuit has been successfully simulated using Xilinx and Modelsim tools. The various parameters are of adder circuits are recorded. Finally, the recorded parameters are compared and concluded that, the pipelined RCA is the efficient in speed and throughput. Future work involves reducing the area and power. The purpose of speed-limiting factor as the system is divided into very short stages and limits the maximum speed-up that can be attained. Thus, the output can be obtained in the stages+1 th clock-rate.

References

[1] I. Unwala and E. Swartzlander, "Superpipelined Adder Designs," in Circuits and Systems, 1993., ISCAS '93, 1993 IEEE International Symposium on, May 1993, pp. 1841–1844.

[2] L. Dadda and V. Piuri, "Pipelined Adders," Computers, IEEE Transactions on, vol. 45, no. 3, pp. 348–356, Mar 1996.

[3] M. D. Ercegovac and T. Lang, Digital Arithmetic. Morgan Kaufmann Publishers, 2004. [8] S. Xing and W. W. Yu, "FPGA Adders: Performance Evaluation and Optimal Design," IEEE Design and Test of Computers, vol. 15, pp. 24– 29, 1998.

[4] P. Bunyk and P. Litskevitch, "Case study in RSFQ design: Fast pipelined parallel adder," *IEEE Trans. Appl. Supercond.*, vol. 9, no. 2, pp. 3714– 3720, Jun. 1999. [5] Virtex-II Platform FPGA Handbook, Xilinx, 2000.

[6] W. J. Kim and Y-B. Kim, "Automating Wave-Pipelined Circuit Design," *IEEE Design and Test of Computers*, December-November 2003, pp.51-58.

[7] V. Sukumar, D. Pan, K. Buck, H. Hess, H. Li, D. Cox and M. M. Mojarradi, "Design of a pipelined Adder Using Skew Tolerant Domino Logic in a 0.35 _m TSMC Process," *2004* IEEE Workshop on Microelectronics and Electron Devices, September 2004, pp. 55-59.

[8] C-H. Huang, J-S.Wang, C. Yeh and C-J.Fang, "The CMOS Carry- Forward Adders," IEEE Journal of Solid-State Circuits, Vol. 39, NO. 2, February 2004, pp. 327-336.

[9] Spartan-3 Generation FPGA User Guide, Xilinx, 2009