ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Parallel, Single-Rail Self-Timed Adder. Formulation for Performing Multi Bit Binary Addition. Without Any Carry Chain Propagation

Y.Gowthami
PG Scholar,
Dept of ECE,
MJR College of Engineering & Technology,
Piler, A.P, India.

J.Sukumar
Associate Professor & HOD,
Dept of ECE,
MJR College of Engineering & Technology,
Piler, A.P, India.

ABSTRACT:

This brief presents a parallel single-rail self-timed adder. It is based on a recursive formulation for performing multi bit binary addition. The operation is parallel for those bits that do not need any carry chain propagation. Thus, the design attains logarithmic performance over random operand conditions without any special speedup circuitry or look-ahead schema. A practical implementation is provided along with a completion detection unit. The implementation is regular and does not have any practical limitations of high fan outs. A high fan-in gate is required though but this is unavoidable for asynchronous logic and is managed by connecting the transistors in parallel. Simulations have been performed using an industry standard toolkit that verify the practicality and superiority of the proposed approach over existing asynchronous adders. There are a myriad designs of binary adders and we focus here on asynchronous self-timed adders. Self-timed refers to logic circuits that depend on and/or engineer timing assumptions for the correct operation. Self-timed adders have the potential to run faster averaged for dynamic data, as early completion sensing can avoid the need for the worst case bundled delay mechanism of synchronous circuits.

Index Terms:

Asynchronous circuits, binary adders, CMOS design, digital arithmetic.

Introduction:

Binary addition is the single most important operation that a processor performs. Most of the adders have been designed for synchronous circuits even though there is a strong interest in clock less/asynchronous processors/circuits [1]. A valid dual-rail carry output also provides acknowledgment from a single-bit adder block. Thus, asynchronous adders are either based on full dual-rail encoding of all signals (more formally using null convention logic [2] that uses symbolically correct logic instead of Boolean logic) or pipelined operation using single-rail data encoding and dual-rail carry representation for acknowledgments. A majority of the present-day digital systems are clock based or synchronous, which assume that signals are binary and time is discrete.

The state updates within the registers are carried out on the rising edge (positive edge) or falling edge (negative edge) of the global clock – single edge triggering. The state of the global clock permits either data loading or data storage. Since the overall clock utilization is only 50% for single edge triggered systems, double edge triggered flip-flops were subsequently proposed in the literature with the motive of increasing the system throughput as data can be loaded on both the rising and falling clock edges and data is retained when the clock signal does not toggle The problems of clock skew and power dissipation have been the major drivers for the worldwide resurgence of interest in asynchronous design.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The design of clock-free or asynchronous systems has thus become attractive for digital system designers during the past two decades although asynchronous logic was explored from the infancy of integrated circuit design. Asynchronous circuits assume that signals are binary but the notion that time is not discrete. An asynchronous system is one in which there is no global synchronization within the system; subsystems within the system are synchronized locally by the communication protocols between them. The results produced by the subsystems in an asynchronous system can be consumed by other subsystems as soon as they are generated without having to wait for a global clock tick. Moreover in asynchronous systems, a sub-system can easily be replaced by another subsystem with the same functionality but with different performance, but this is not a straightforward task in case of a synchronous system as the clock period might have to be recomputed. An asynchronous system stage that involves request/acknowledge handshake (signal exchange) signaling protocol is shown in figure 1.1. However, robust asynchronous systems embed the request information within the data wires and are usually referred to as self-timed systems. Self-timed systems are characterized by the absence of any timing reference to which all the operations are synchronized – being in stark contrast to synchronous systems where all operations are synchronized to the global clock signal.

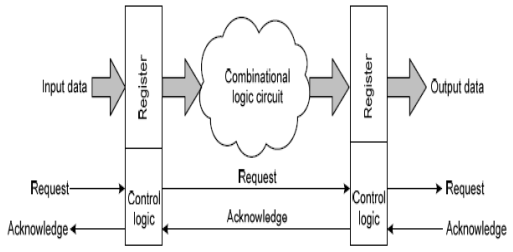

Fig 1.1: A typical asynchronous system stage

The half adder adds two one-bit binary numbers A and B. It has two outputs, S and C (the value theoretically carried on to the next addition); the final sum is 2C + S.

The simplest half-adder design, pictured on the right, incorporates an XOR gate for S and an AND gate for C. With the addition of an OR gate to combine their carry outputs, two half adders can be combined to make a 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.[2] The full-adder is usually a component in a cascade of adders, which add 8, 16, 32, etc. binary numbers. The circuit produces a two-bit output sum typically represented by the signals C_{out} and S, where $\text{sum} = 2 \times C_{out} + S$.

1.2 Existing System:

Addition is the most common and often used arithmetic operation on microprocessor, digital signal processor, especially digital computers. Also, it serves as a building block for synthesis all other arithmetic operations. Therefore, regarding the efficient implementation of an arithmetic unit, the binary adder structures become a very critical hardware unit.

1.2.1 Carry select adder (CSA):

The carry select adder comes in the category of conditional sum adder. Conditional sum adder works on some condition. Sum and carry are calculated by assuming input carry as 1 and 0 prior the input carry comes. When actual carry input arrives, the actual calculated values of sum and carry are selected using a multiplexer. The conventional carry select adder consists of k/2 bit adder for the lower half of the bits i.e. least significant bits and for the upper half i.e. most significant bits (MSB's) two k/ bit adders. In MSB adders one adder assumes carry input as one for performing addition and another assumes carry input as zero. The carry out calculated from the last stage i.e. least significant bit stage is used to select the actual calculated values of output carry and sum. The selection is done by using a multiplexer.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

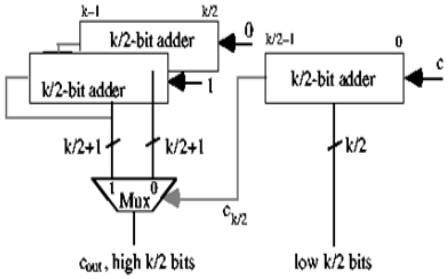

Fig 1.2 : Carry select adder

In electronics, a carry-select adder is a particular way to implement an adder, which is a logic element that computes the $(n+1)$-bit sum of two n-bit numbers. The carry-select adder is simple but rather fast, having a gate level depth of $O(\sqrt{n})$. The carry-select adder generally consists of two ripple carry adders and a multiplexer. Adding two n-bit numbers with a carry-select adder is done with two adders (therefore two ripple carry adders) in order to perform the calculation twice, one time with the assumption of the carry being zero and the other assuming one. After the two results are calculated, the correct sum, as well as the correct carry, is then selected with the multiplexer once the correct carry is known. The number of bits in each carry select block can be uniform, or variable. In the uniform case, the optimal delay occurs for a block size of $\lfloor\sqrt{n}\rfloor$. When variable, the block size should have a delay, from addition inputs A and B to the carry out, equal to that of the multiplexer chain leading into it, so that the carry out is calculated just in time. The $O(\sqrt{n})$ delay is derived from uniform sizing, where the ideal number of full-adder elements per block is equal to the square root of the number of bits being added, since that will yield an equal number of MUX delays.

Problem in the system:

This technique of dividing adder into stages increases the area utilization but addition operation fastens.

1.2.2 Carry look ahead adder (CLA):

A carry-look ahead adder (CLA) is a type of adder used in digital logic. A carry-look ahead adder improves speed by reducing the amount of time required to determine carry bits. It can be contrasted with the simpler, but usually slower, ripple carry adder for which the carry bit is calculated alongside the sum bit, and each bit must wait until the previous carry has been calculated to begin calculating its own result and carry bits. The carry-look ahead adder calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger value bits. The Kogge-Stone adder and Brent-Kung adder are examples of this type of adder. A ripple-carry adder works in the same way as pencil-and-paper methods of addition. Starting at the rightmost (least significant) digit position, the two corresponding digits are added and a result obtained. It is also possible that there may be a carry out of this digit position (for example, in pencil-and-paper methods, "9+5=4, carry 1").

Accordingly all digit positions other than the rightmost need to take into account the possibility of having to add an extra 1, from a carry that has come in from the next position to the right. This means that no digit position can have an absolutely final value until it has been established whether or not a carry is coming in from the right. Moreover, if the sum without a carry is 9 (in pencil-and-paper methods) or 1 (in binary arithmetic), it is not even possible to tell whether or not a given digit position is going to pass on a carry to the position on its left. At worst, when a whole sequence of sums comes to ...99999999... (in decimal) or ...11111111... (in binary), nothing can be deduced at all until the value of the carry coming in from the right is known, and that carry is then propagated to the left, one step at a time, as each digit position evaluated "9+1=0, carry 1" or "1+1=0, carry 1". It is the "rippling" of the carry from right to left that gives a ripple-carry adder its name, and its slowness. When adding 32-bit integers, for instance, allowance has to be made for the possibility that a carry could have to ripple through every one of the 32 one-bit adders.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Carry look ahead depends on two things:

1. Calculating, for each digit position, whether that position is going to propagate a carry if one comes in from the right.
2. Combining these calculated values to be able to deduce quickly whether, for each group of digits, that group is going to propagate a carry that comes in from the right.

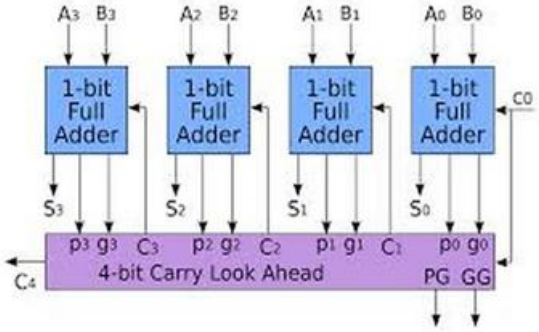

Fig 1.3: Carry look ahead adder

For each bit in a binary sequence to be added, the Carry Look Ahead Logic will determine whether that bit pair will generate a carry or propagate a carry. This allows the circuit to "pre-process" the two numbers being added to determine the carry ahead of time. Then, when the actual addition is performed, there is no delay from waiting for the ripple carry effect.

Problem in the system:

The disadvantage of the CLA adders is that the carry expressions become quite complex for more than 4 bits.

Proposed system:

The adder first accepts two input operands to perform half additions for each bit. Subsequently, it iterates using earlier generated carry and sums to perform half-additions repeatedly until all carry bits are consumed and settled at zero level.

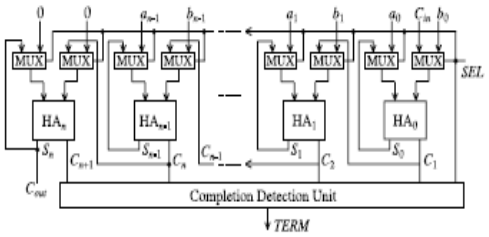

Fig 3.1: General block diagram of parallel self timed adder (PASTA)

1 Architecture of PASTA:

The general architecture of the adder is shown in Fig. 3.1. The selection input for two-input multiplexers corresponds to the Req handshake signal and will be a single 0 to 1 transition denoted by SEL. It will initially select the actual operands during SEL = 0 and will switch to feedback/carry paths for subsequent iterations using SEL = 1. The feedback path from the HAs enables the multiple iterations to continue until the completion when all carry signals will assume zero values.

2 State Diagrams:

In Fig. 3.2, two state diagrams are drawn for the initial phase and the iterative phase of the proposed architecture. Each state is represented by (Ci+1 Si) pair where Ci+1, Si represent carry out and sum values, respectively, from the ith bit adder block. During the initial phase, the circuit merely works as a combinational HA operating in fundamental mode. It is apparent that due to the use of HAs instead of FAs, state cannot appear. During the iterative phase (SEL = 1), the feedback path through multiplexer block is activated. The carry transitions (Ci) are allowed as many times as needed to complete the recursion. From the definition of fundamental mode circuits, the present design cannot be considered as a fundamental mode circuit as the input–outputs will go through several transitions before producing the final output. It is not a Muller circuit working outside the fundamental mode either as internally, several transitions will take place, as shown in the state diagram.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

This is analogous to cyclic sequential circuits where gate delays are utilized to separate individual states.

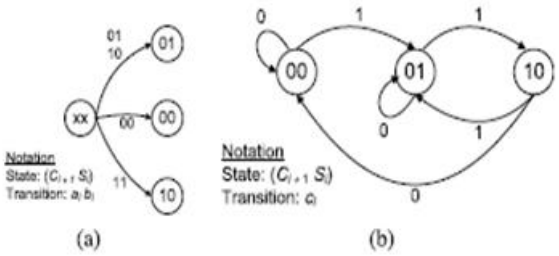

Fig 3.2: State diagram of PASTA (a) initial phase (b) iterative phase

3 Recursive Formula for Binary Addition

Let S_i^j and C_{i+1}^j denote the sum and carry, respectively, for ith bit at the j th iteration. The initial condition (j = 0) for addition is formulated as follows:

$$S_i^0 = a_i \oplus b_i$$

$$C_{i+1}^0 = a_i b_i.$$

The j th iteration for the recursive addition is formulated by

$$S_i^j = S_i^{j-1} \oplus C_i^{j-1}, \quad 0 \le i < n$$

$$C_{i+1}^j = S_i^{j-1} C_i^{j-1}, \quad 0 \le i \le n.$$

The recursion is terminated at kth iteration when the following condition is met:

$$C_n^k + C_{n-1}^k + \cdots + C_1^k = 0, \quad 0 \le k \le n.$$

3.2 Software required

- Dsch (for schematics)
- Microwind (for Layouts)

1. Dsch (for schematics)

The DSCH program is a logic editor and simulator. DSCH is used to validate the architecture of the logic circuit before the microelectronics design is started. DSCH provides a user-friendly environment for hierarchical logic design, and fast simulation with delay analysis, which allows the design and validation of complex logic structures. DSCH also features the symbols, models and assembly support for 8051 and 16F84 controllers.

Designers can create logic circuits for interfacing with these controllers and verify software programs using DSCH.

- User friendly environment for rapid design of logic circuits
- Handles both conventional pattern based logic simulation and intuitive on screen mouse simulation
- Supports hierarchical logic design
- Built-in extractor which generates a SPICE netlist from the schematic diagram
- Current and power consumption analysis.
- Generates a VERILOG description of the schematic for layout editor
- Immediate access to symbol properties (Delay, fanout)
- Models and assembly support for 8051 and PIC 18f84
- Sub-micron, deep-submicron, nanoscale technology support.
- Supported by huge symbol library.

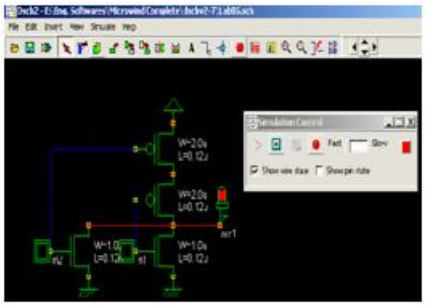

Fig 3.3 : Dsch window

2. Micro wind:

The MICROWIND2 program allows the student to design and simulate an integrated circuit at physical description level. The package contains a library of common logic and analog ICs to view and simulate. MICROWIND2 includes all the commands for a mask editor as well as original tools never gathered before in a single module (2D and 3D process view, VERILOG compiler, tutorial on MOS devices). You can gain access to Circuit Simulation by pressing one single key. The electric extraction of your circuit is automatically performed and the analog simulator produces voltage and current curves immediately.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

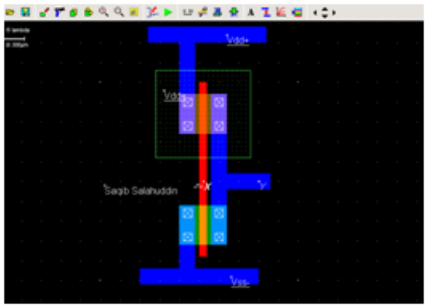

Fig 3.4 : Microwind window

RESULTS:

VI. CONCLUSION:

This brief presents an efficient implementation of a PASTA. Initially, the theoretical foundation for a single-rail wave-pipelined adder is established. Subsequently, the architectural design and CMOS implementations are presented. The design achieves a very simple n-bit adder that is area and interconnection-wise equivalent to the simplest adder namely the RCA. Moreover, the circuit works in a parallel manner for independent carry chains, and thus achieves logarithmic average time performance over random input values. The completion detection unit for the proposed adder is also practical and efficient. Simulation results are used to verify the advantages of the proposed approach.

REFERENCES:

1. D. Geer, "Is it time for clockless chips? [Asynchronous processor chips]," IEEE Comput., vol. 38, no. 3, pp. 18–19, Mar. 2005.

2. J. Sparsø and S. Furber, Principles of Asynchronous Circuit Design. Boston, MA, USA: Kluwer Academic, 2001.

3. P. Choudhury, S. Sahoo, and M. Chakraborty, "Implementation of basic arithmetic operations using cellular automaton," in Proc. ICIT, 2008, pp. 79–80.

4. M. Z. Rahman and L. Kleeman, "A delay matched approach for the design of asynchronous sequential circuits," Dept. Comput. Syst. Technol., Univ. Malaya, Kuala Lumpur, Malaysia, Tech. Rep. 05042013, 2013.

5. M. D. Riedel, "Cyclic combinational circuits," Ph.D. dissertation, Dept. Comput. Sci., California Inst. Technol., Pasadena, CA, USA, May 2004.

6. R. F. Tinder, Asynchronous Sequential Machine Design and Analy-sis: A Comprehensive Development of the Design and Analysis of Clock-Independent State Machines and Systems. San Mateo, CA, USA: Morgan, 2009.

7. W. Liu, C. T. Gray, D. Fan, and W. J. Farlow, "A 250-MHz wave pipelined adder in 2-µm CMOS," IEEE J. Solid-State Circuits, vol. 29, no. 9, pp. 1117–1128, Sep. 1994.

8. F.-C. Cheng, S. H. Unger, and M. Theobald, "Self-timed carry-lookahead adders," IEEE Trans. Comput., vol. 49, no. 7, pp. 659–672, Jul. 2000.

9. S. Nowick, "Design of a low-latency asynchronous adder using spec-ulative completion," IEE Proc. Comput. Digital Tech., vol. 143, no. 5, pp. 301–307, Sep. 1996.

10. N. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective. Reading, MA, USA: Addison-Wesley, 2005.

11. C. Cornelius, S. Koppe, and D. Timmermann, "Dynamic circuit tech-niques in deep submicron technologies: Domino logic reconsidered," in Proc. IEEE ICICDT, Feb. 2006, pp. 1–4.

12. M. Anis, S. Member, M. Allam, and M. Elmasry, "Impact of technology scaling on CMOS logic styles," IEEE Trans. Circuits Syst., Analog Digital Signal Process., vol. 49, no. 8, pp. 577–588, Aug. 2002.