IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
ISSN: 2278-2834, ISBN: 2278-8735. Volume 3, Issue 6 (Nov. - Dec. 2012), PP 07-12
www.iosrjournals.org

Implementation of ALU Using Asynchronous Design

P. Amrutha[1] and G. Hanumantha Reddy[2]
[1&2]*Department of Electronics & Communication Engineering Chaitanya Engineering College Visakhapatnam, A.P, India*

Abstract: *Power consumption has become one of the biggest challenges in design of high performance microprocessors. In this paper we present a design technique using GALs (Globally-Asynchronous Locally-Synchronous) for implementing asynchronous ALUs, which aims to eliminate the global clock. Here ALUs are designed with delay insensitive dual rail four phase logic and CMOS domino logic. It ensures economy in silicon area and potentially for low power consumption. This has been described and implemented in order to achieve a high performance in comparison with synchronous and available asynchronous design. Also simulation results, show significant reduction in the number of transistors as well as delay.*

I. Introduction

The ALU (Arithmetic Logic Unit) is implemented by using asynchronous pipeline architecture. The architecture is having simple handshake cells and the handshake cells are embedded in the pipeline stage as normal logic cells. As a result, the speed of the ALU can be very fast. Pipeline is an important methodology to speed up the design [1]. It allows many operations to occur in parallel. Most pipeline systems use synchronous pipeline, in which a global clock is used to clock all the stages in the synchronous pipeline system. As a result the global clock signal is always connected to many logic cells and so the time of the clock signal to reach different logic cells may be different due to different length of connection from the global clock to different logic cells and hence this a problem. Asynchronous pipeline may be a solution. In asynchronous pipeline global clock is removed. Instead of global clock, handshake control signals are used to govern the operation of the pipeline [1&2].

One approach which promises high performance with low power consumption is the use of asynchronous computing techniques. To investigate this self-timed implementation of the ARM microprocessor - a 32-bit RISC architecture developed by Advanced RISC Machines Limited - is being produced as a commercially realistic technology demonstrator. Other researchers [3] have demonstrated the feasibility of building a complete asynchronous microprocessor; the current project addresses the problems associated with the re-implementation of an existing commercial architecture with the specific goal of minimizing power consumption several approaches to asynchronous circuit design are currently being investigated [4].

Some propose by using gated clocks to reduce the switching activity of logic in redundant cycles [5]. Although it minimizes the skew, this methodology is limited when operating frequency is very high. On the other hand, Globally Asynchronous and Locally Synchronous (GALS) technique aims to eliminate the global clock, by partitioning the system into several synchronous blocks and communicating asynchronously among blocks [7]. However, the global signaling protocol increases the total area power penalty and affects performance of the system. Several researchers propose asynchronous approaches to cope with performance and timing issue. Designed a 16-bit asynchronous ALU with an asynchronous pipeline architecture [6]. In this approach, simple handshake cells embedded in pipeline stages make the ALU run fast. Increases total area power penalty and affects performance of the system. However, large power has consumed by this design while waiting for the incoming data.

Since the inappropriate rotate-wire concept of data buses, the time required for each multiplication operation becomes larger. In reducing the time dependency of an asynchronous design of Quantum dot Cellular Automata (QCA), Paper [8] used GALS with delay-insensitive data encoding scheme. Here each gate has locally synchronized by corresponding clocking zone(s). We focus in this asynchronous design to reduce the transistors count, power consumption and delay significantly by using delay insensitive dual rail logic and bundled data bounded delay model.

II. Introduction of Asynchronous Circuits

Asynchronous circuits are fundamentally different; they also assume binary signals, *but there is no common and discrete time*. Instead the circuits use handshaking between their components in order to perform the necessary synchronization, communication, and sequencing of operations. Expressed in ‘synchronous terms’ this results in a behavior that is similar to systematic fine-grain clock gating and local clocks that are not in phase and whose period is determined by actual circuit delays – registers are only clocked where and when

needed. In all protocols, Muller pipeline is used. The 4-phase bundled data and 2–phase bundled data are pipelined designs in which matching delay elements needed to be inserted between latches to maintain correct behavior in the request signal path. On the other hand, 4-phase dual rail has designed to combine encoding of data and request. [10].

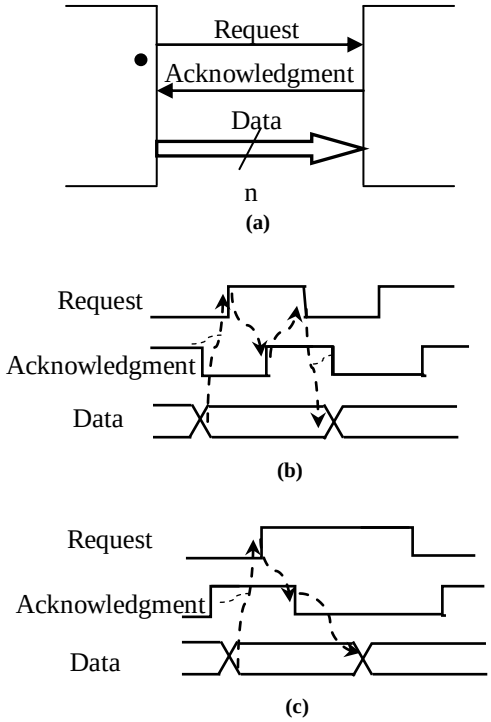

Figure 1. (a) A bundled-data channel (b) A 4-phase bundled-data protocol (c) A 2-phase bundled-data protocol

2.1. Bundled-data Protocols

The term *bundled-data* refers to a situation where the data signals use normal Boolean levels to encode information, and where separate request and acknowledge wires are bundled with the data signals, Figure. 1(a). In the *4- phase* protocol illustrated in Figure. 1(b) the request and acknowledge wires also use normal Boolean levels to encode information, and the term 4-phase refers to the number of communication actions: (1) the sender issues data and sets request high, (2) the receiver absorbs the data and sets acknowledge high, (3) the sender responds by taking request low (at which point data is no longer guaranteed to be valid) and (4) the receiver acknowledges this by taking acknowledge low. At this point the sender may initiate the next communication cycle.

2.2. The 4-phase dual-rail protocol concept

This protocol makes the reliable communication between blocks of designed architecture regardless of delays in the wires connecting two blocks and also which is delay insensitive [9&10]. It uses a single wire for each data bit and one extra control line for each data word. It provides the reliable communication between blocks by combined encoding of data and request with an acknowledge signal after completion detection. This logic uses two request wires per bit of information d:; one wire d.t is used for signaling a logic 1(or true), and another wire d.f is used for signaling logic 0(or false). In a single bit channel with 4 phase dual rail logic, the request signal can be either of d.t or d.f for handshaking purpose. Viewed together the {x.f, x.t}={1,0} and {x.f, x.t}={0,1} represent "valid data" (logic 0and logic 1 respectively) and {x.f, x.t}={0,0} represents "no data" (or "empty value" or "E"). The codeword {x.f, x.t}={1,1} is not used, and a transition from one valid codeword to another valid codeword is not allowed, as illustrated in Figure. 2(a).

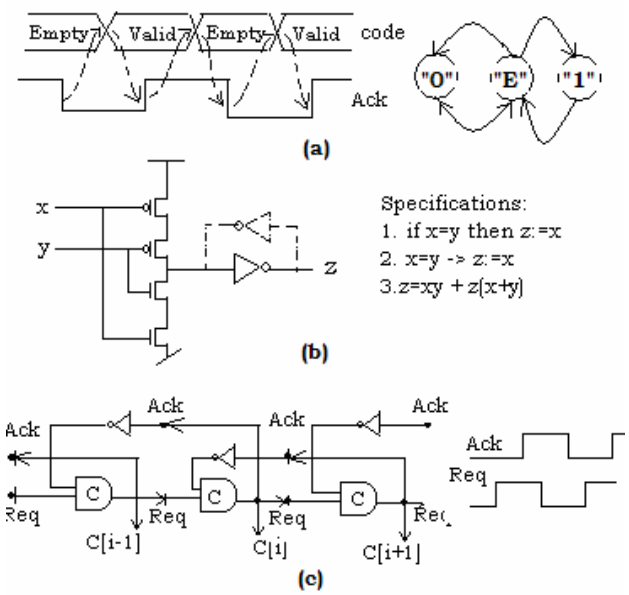

Figure 2. (a) The 4-phase dual rail logic (b) Muller C-element and indication concept and (c) Muller Pipeline

2.3. Muller C-element and Indication Concept

The concept of indication or acknowledgement plays an important role in the design of asynchronous circuits for synchronization. Muller C-element is a state-holding element much like an asynchronous set-reset latch [1]. When both inputs are 0 the output is set to 0, and when both inputs are 1 the output is set to 1. For other input combinations {(0, 1) or (1, 0)} the output does not change. Consequently, one can see the output changes from 0 to 1 can conclude that both inputs are now at 1, similarly one can see the output changes from 1 to 0 may conclude that both inputs are now 0. In this circuit design, the absence of a clock means that, in many circumstances, signals are required to be "valid data: {x.f, x.t}={1, 0} and {x.f, x.t}={0, 1}" all the time that every signal transition has a meaning and, consequently, that hazards and races must be avoided. Signal transitions are not indicated (acknowledged) for the other signal transitions such as {(0, 1), (1, 0)} and that are used to avoid the source of hazards. A circuit accomplish this requirement with Muller C-element is as shown in Figure. 2(b). The Muller C-element is indeed a fundamental component that is extensively used in asynchronous circuit design [1].

2.4. Muller Pipeline

A 4-phase dual-rail pipeline is based on the Muller pipeline that relays handshakes in Figure. 2(c). In the Muller pipeline, there is a 1-bit wide and 3-stage deep pipeline that uses a common acknowledge signal per stage to synchronize. Here the pipeline stage can store empty codeword {d.t, d.f}={0, 0}, causing the acknowledge signal out of that stage to be logic 0 or one of the two valid code words {0, 1} and {1, 0}, causing the acknowledge signal out of that stage to be logic 1. Initially all of the C-elements have been initialized to 0 and during the operation, according to the successor value; the current C-element transfers its predecessor's value for handshaking [11]. To understand what happens let's consider the i^{th} C-element, c[i]: It will propagate a 1 from its predecessor c[i-1], only if its successor c[i+1] is 0. In a similar way it will propagate a 0 from its predecessor if its successor is 1.

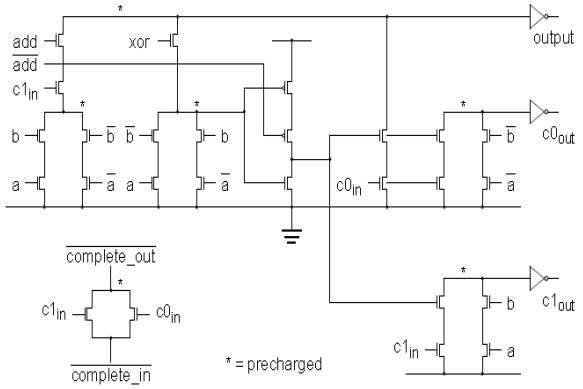

Figure 3. The Adder Circuit in ALU

This ALU has no special fast carry logic and performs addition with a chain of thirty two full adders. The only concession to the asynchronous nature of the unit made at the design stage was that the carries between the individual bits are encoded onto pairs of wires signaling the "0" and "1" states of the carry bit respectively. In this fashion it is possible to detect that a carry signal has arrived at a given bit position by observing a change in state of one of these signals. ALU completion is signaled when a carry has been transmitted to all 32 bits in the word [10].

III. Design and Implementation of ALU

Using the logics and principles outlined in Section II, an ALU has designed at the transistor level for single bit operation as shown in the Figure. 3 to demonstrate our design concept. A single bit-slice ALU uses only 53 transistors and is capability of operations in Table 1. Since we emphasize on the design of asynchronous component, there is no hardware implementation for 4- phase dual rail with Muller C. However, the proposed circuit assumes the signaling from such logic blocks. For example, $c0_{out}$ and $c1_{out}$ act as two wires of 4- phase logic, which makes reliable operation between its predecessor and successor blocks.

3.1. Architecture design & Implementation:

We designed a 32-bit ALU, which requires 1696 transistors. The basic principle of Bundled data – Bounded delay model of Sutherland's micro pipelines is used here [1]. The timing characteristics of all data busses of this architecture are bundled together. Here the statuses of the data busses are indicated by 4 phase-dual rail hand shake signals. The clock power reduction at the architectural level is mainly due to pipeline technique. The dynamic logic of completion detection unit ensures precise internal operation, because of its 4-phase dual logic. It is also carrying the timing information because it uses common timing characteristics.

Table 1. Functions available for the proposed ALU

Logic Function	Basic Operation	a-input	b-input
and	AND	True	True
add	AND	True	True
add with carry	AND	True	True
subtract	AND	True	complement
reverse subtract	AND	complement	True
subtract with carry	AND	True	Complement
reverse subtract with carry	AND	complement	True
test bits	AND	True	True
compare	AND	True	Complement
compare negative	AND	True	True
bit clear	AND	True	Complement
xor	XOR	True	True
test equal	XOR	True	True
or	OR	True	True
move	OR	Zero	True
move NOT	OR	Zero	complement

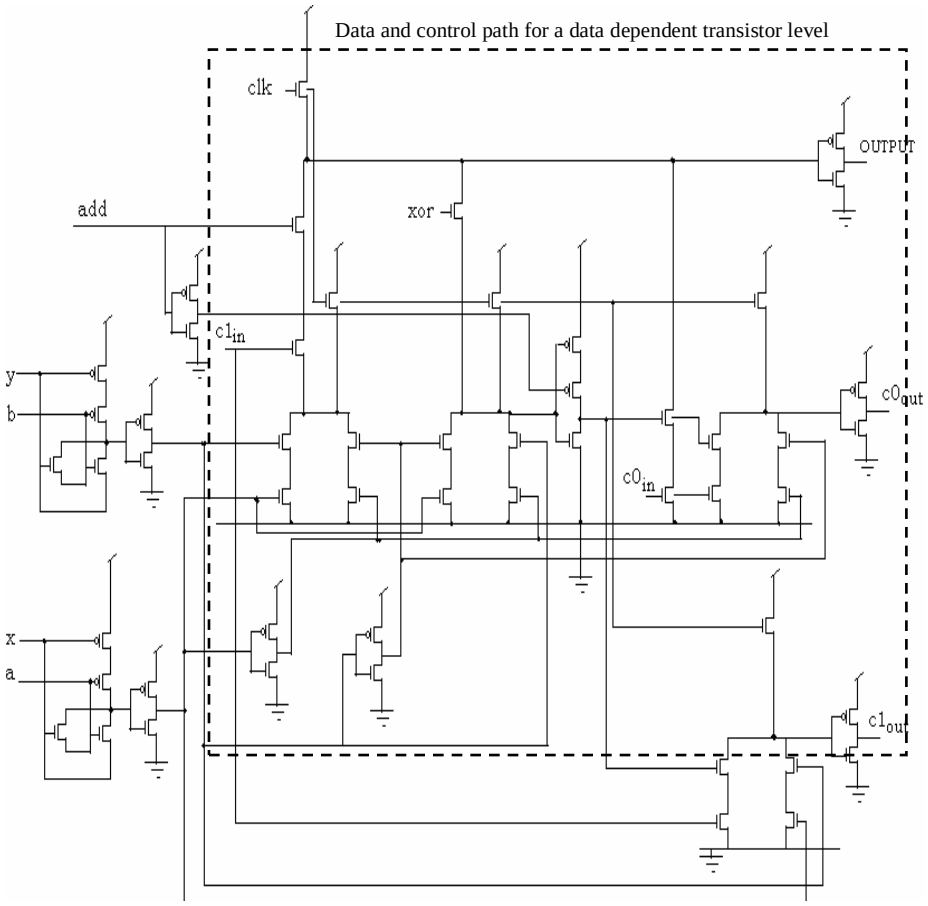

Figure 4. A CMOS level Asynchronous ALU circuit for 1-bit operation

IV. Analysis

Addition is one of fundamental functions of the ALU. We start by analyzing the number of transistors used in the addition. About 80% of the operations require some form of addition [11]. If we improve the processing time of addition operation, the performance of complete ALU can also be improved. The latency required by our design is depended upon the operation, the input data at that incident and the carry flow across the whole word length, i.e. it needs to propagate carry until it has predicted by the completion detection stage. The average length of the mean carry propagation distance is varying according to input data. In this 32-bit operation, a sum of 140 transistors has used for precharging (domino logic) and buffer purposes to meet the specifications at the layout.

The simulated output waveform for the addition operation performed by this ALU presented in Figure. 5. It is performed with V_{DD}=1.8V, input sequence $c1_{in}$=1111, $c0_{in}$=0000, a=0011, b=0101 and the simulated output sequence is output=1001, $c0_{out}$=1000, $c1_{out}$=0111 which coincides the expected specification. This simulation has done by HSPICE with 10ns local clock period at room temperature. The simulation results for the power consumption of typical addition operation with different supply voltages are shown in the Figure. 5. The simulation results of HSPICE 0.18um technology shows, the average power consumption for typical addition operation is $1.02x10^{-4}$W under 1.8V supply with 1000 sample inputs at room temperature and average time delay is 2.5ns. A comparison of the simulated time performance and transistor count of this design with other published alternatives as shown in Table 2 and Table 3. They clearly indicate a significant reduction in transistors count. Our design has much reduction in silicon area. In addition, this architecture enables to have reduced switching capacitance because of missing master clock in the transistors of the ALU circuit design. It gives reduced switching actions for every arithmetic operation and reduction in silicon area.

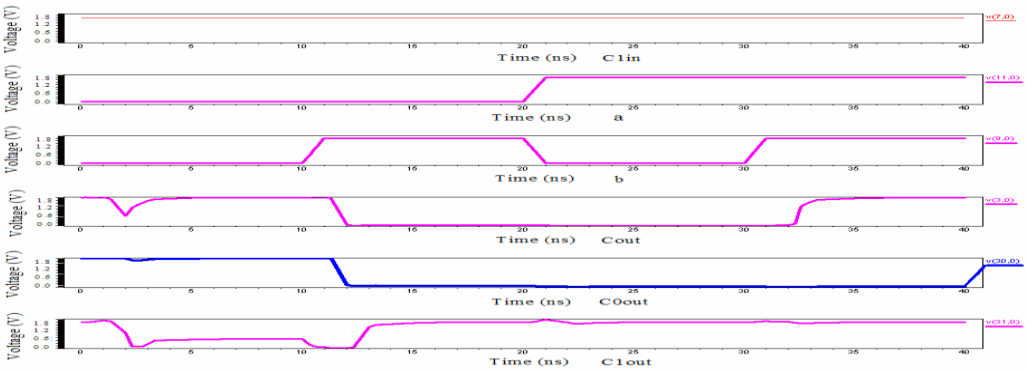

Figure 5. Simulated waveforms for the 1 bit addition operation

Table 2. Simulated Results for Time Performance

Time (ns)	Best Case	Worst Case	Average
Our Design	3	4	3.5
Synchronous	-	-	4
Ref [12]	3	6	4.5
Ref [13]	2.5	7.5	5

Table 3. Comparison of Reported Design

Comparison	Discussed Design and Existing Designs		
	Synchronous ARM ALU [12]	Asynchronous ARM ALU [12]	Our Design
Technology	1.2 um CMOS	1.2 um CMOS	0.18 um CMOS
Supply Voltage	~5V (CO/SS)	~5V (CO/SS)	~1.8V (CO/MS)
Data Width	32 bit	32 bit	32 bit
Timing Purpose	-	140 (Transistors)	140 (Transistors)
Self Time Unit	3000 (Transistors)	2300 (Transistors)	1696 (Transistors)

V. Conclusion

The proposed design can reduce silicon area and improve the performance on average. This is mainly due to the use of the asynchronous design concepts with CMOS dynamic domino logic. They result in reduction in the transistor count & parasitic capacitances. Subsequently, power consumption can be reduced. Furthermore, this architecture provides glitch free operation, which is an important key factor to better reliability and performance. Effective reduction in area, power consumption and reasonable performance improvement are the special features of our designed architecture.

References

[1] I. E. Sutherland, "Micropipelines", Communications of the ACM, pp 720-738, vol. 32, No. 6, 1989.
[2] Scott Hauck, Asynchronous Design Mythologies: An Overview, in 1995, *Proc. of the IEEE*, vol. 83, no. 1, pp. 69-93.
[3] A. J. Martin, S. Burns, T. K. Lee, D. Borkovie, and P. J. Hazewindus, The Design of an Asynchronous Microprocessor, in 1989, *Proc. of the Decennial CalTech Conference on VLSI*, MIT Press, pp 351-373.
[4] G. Gopalakrishnan, and P. Jain Some Recent Asynchronous System Design Methodologies, Technical Report Number UU-CS-TR-90-016, University of Utah, 1990.
[5] L. Benini, and G. De Micheli, Transformations and synthesis of FSM's for low power gated clock implementation, *IEEE Trans. Computer- Aided Design*, vol. 15, 1996.
[6] J. M. Rabaey, and M. Pedram, Low Power Design Methodologies, (Kluwer Academic Publishers, 1996, ISBNO-7923-9630-8).
[7] A. Hemani, T. Meincke , S. Kumar, A. Postula, P. Nilsson, J. Oberg, P. Ellervee, and D. Lundqvist, Lowering power consumption in clock by using globally asynchronous locally synchronous design style, in *1999, Proc. IEEE Design Automat. Conf.*, pp. 873-878.
[8] M. Choi, and N. Park, Locally synchronous, globally asynchronous design for quantum-dot cellular automata (LSGA QCA), in *2005, Proc. IEEE NANO*, pp. 121-124.
[9] S. Hauck, Asynchronous design methodologies: an overview, in *1995, Proc. IEEE*, vol. 83, pp. 63-69.
[10] J. Sparso, and S. Furber, Principles of Asynchronous Circuit Design: A Systems Perspective, Kluwer Academic Publishers, 2001.
[11] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel and F. Baez, "Reducing power in high performance microprocessors", in *1998, Proc. IEEE Design Automat. Conf.*, pp. 600-610.
[12] J. D. Garside, "A CMOS VLSI implementation of an asynchronous ALU", in *1993, Proc. IFIP working conference on Asynchronous Design Methodologies, Manchester*, M139PL.
[13] D. Kearney, and N. W. Bergmann, "Bounded data asynchronous multipliers with data dependent computation times", in *1997, Proc. IEEE Advanced research in asynchronous circuits and systems*, pp.186-197.