International Journal of Advances in Engineering Research http://www.ijaer.com

(IJAER) 2015, Vol. No. 9, Issue No. VI, June **ISSN: 2231-5152**

ASYNCHRONOUS FINE-GRAIN POWER-GATED LOGIC WITH PARTIAL CHARGE REUSE MECHANISM

[1]K. Venu Jayanth Reddy, [2]Mrs.M.Valarmathi

[1]Student, Department of Electronics and Communication/ VLSI Design
SRM University, Chennai, India
[2]Assistant Professor (Sr.G), Department of Electronics and Communication
SRM University, Chennai, India

ABSTRACT

This project presents a novel low-power logic family, called asynchronous fine-grain power-gated logic (AFPL). Each pipeline stage in the AFPL circuit is comprised of efficient charge recovery logic (ECRL) gates, which implement the logic function of the stage, and a handshake controller, which handles handshaking with the neighboring stages and provides power to the ECRL gates which reduces power by using protocols.

Keywords- Asynchronous fine-grain power-gated logic (AFPL), partial charge reuse (PCR) mechanism.

INTRODUCTION

Most digital circuits designed and fabricated today are "synchronous". In essence, they are based on two fundamental assumptions that greatly simplify their design: (1) all signals are binary, and (2) all components share a common and discrete notion of time, as defined by a clock signal distributed throughout the circuit. Asynchronous circuits are fundamentally different [1-3]. For synchronous circuits, power gating can be implemented in the fine-grain or coarse-grain manner. The fine-grain power gating approach has more opportunities to reduce leakage at run-time than the coarse-grain power gating approach [4]. However, there are several design issues associated with incorporating fine-grain power-gating in synchronous circuits. First, fine-grain power-gating needs significant buffering and routing resources to distribute the sleep control signal to all the cells in the synchronous system [5].

ASYNCHRONOUS VERSUS HANDSHAKING PROTOCOL

In an asynchronous circuit the clock signal is replaced by some form of handshaking between neighboring registers; for example the simple request-acknowledge based handshake protocol [6]. In the following chapter we look at alternative handshake protocols and data encodings.

An important message is that the "handshake-channel and data-token view" represents a very useful abstraction that is equivalent to the register transfer level (RTL) used in the design of synchronous circuits. This *data-flow abstraction*, as we will call it, separates the structure and function of the circuit from the implementation details of its components.

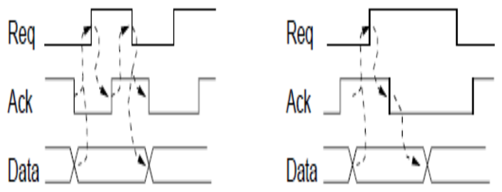


Fig. 1 (a) A bundled-data channel. (b) A 4-phase bundled-data protocol.

A. *AFPL* pipelines

The handshake protocol used in the AFPL pipeline is the four phase dual-rail protocol, in which the request signal is encoded into the data signals (i.e., there is no separate request signal) and *n* pairs of wires are required to encode *n*-bit data.. It can be combined with the PCR mechanism. When AFPL incorporates the PCR mechanism, it is denoted by AFPL-PCR; otherwise, it is denoted by AFPL w/o PCR. Fig. 1 shows the structure of the AFPL pipelines.

AFPL –PCR MECHANISM

An asynchronous system is made up of many autonomous modules, each of which operates at its own rate and communicates with its neighboring modules only when it needs to exchange information. The synchronization [7] between those autonomous modules is not achieved by a global clock but rather by local handshake signals, *request* and *acknowledge*. The handshake protocol used in the AFPL pipeline is the four phase dual-rail protocol [8], in which the request signal is encoded into the data signals (i.e., there is no separate request signal) and *n* pairs of wires are required to encode *n*-bit data.

For example, one-bit information, denoted by *d*, can be encoded with a pair of wires *d.t* and *d. f* . If (*d.t, d. f)* = (1, 0), the codeword (*d.t, d. f)* is a valid token and represents a logic 1; if (*d.t, d. f)* = (0, 1), the codeword (*d.t, d. f)* is a valid token and represents a logic 0; if (*d.t, d. f)* = (0, 0), the codeword (*d.t, d. f)* is an empty token and represents no data. In the four-phase dual-rail protocol, the transferring of data from the sender to the receiver involves the following four actions:

A *The 4-Phase Dual-Rail Protocol*

The 4-phase dual-rail protocol encodes the request signal into the data signals using two wires per bit of information that has to be communicated, figure 2.2. In essence it is a 4-phase protocol using 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). When observing a 1-bit channel one will see a sequence of 4-phase handshakes where the participating.

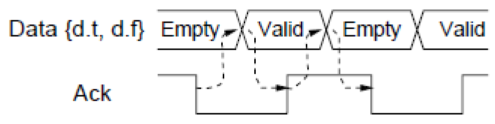


Fig. 2 A delay-insensitive channel using the 4-phase dual-rail protocol.

By using dual-rail carry signals (*d_t_d_f*) it is possible to design circuits that indicate completion as part of the computation and thus achieve actual case latency.

The sender issues a valid codeword on the communication channel; 2) the receiver acquires the valid codeword from the communication channel, and then asserts the acknowledge signal; 3) the sender responds by issuing an empty codeword (i.e., taking all data wires LOW) to indicate that the data on the communication channel is no longer valid; and 4) the receiver de asserts the acknowledge signal to complete the communication cycle.

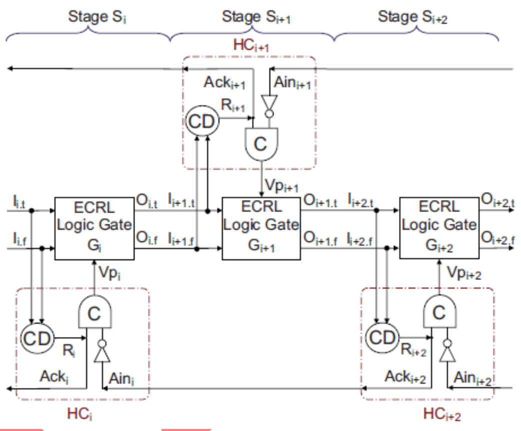


Fig. 3 HC: Handshake Controller

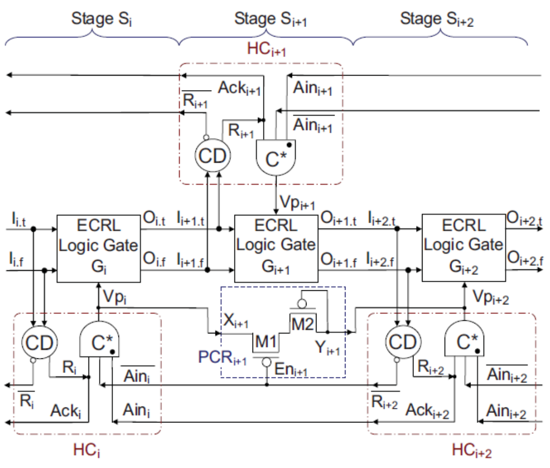


Fig. 4 PCR: Partial Charge Reuse Unit

ENHANCED PCR MECHANISM:

The C*-element in *HCi* has three inputs, *Ri* , *Aini*, and *Aini* , the latter two of which are complementary. *Ri* is the request signal from the CD in *HCi* . *Aini* and *Aini* are the acknowledge signals from *HCi*+2. After reset, *Ri* = 0, *Aini* = 0, and *Aini* = 1. The transitions of *Ri* and *Aini* involve the following four events.

1) Event Req↑: *Ri* transits from LOW to HIGH. This event occurs when a valid token arrives at stage *Si* .

2) Event Ack↑: *Aini* transits from LOW to HIGH and *Aini* transits from HIGH to LOW. This event occurs when the valid output of stage *Si* has been received by stage *Si*+2.

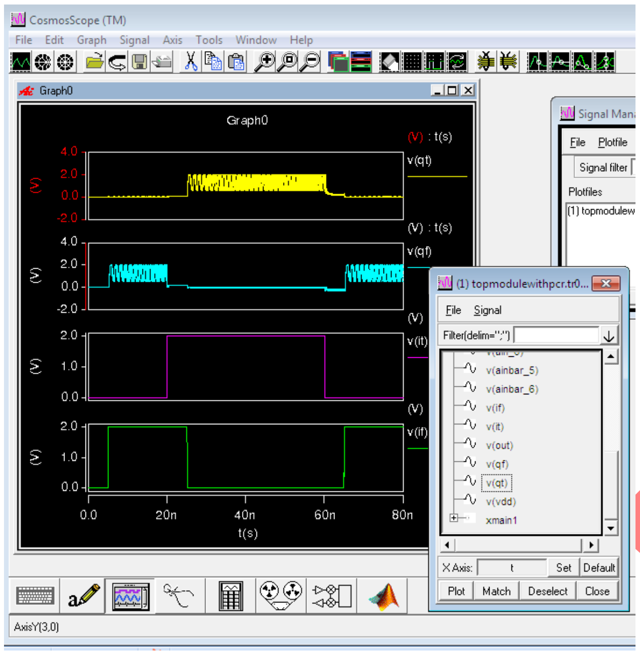
3) Event Req↓: *Ri* transits from HIGH to LOW. This event occurs when an empty token arrives at stage *Si* .

4) Event Ack↓: *Aini* transits from HIGH to LOW and *Aini* transits from LOW to HIGH. This event occurs when the empty output of stage *Si* has been received by stage *Si*+2.

In this brief, reconfigurable radix- FFT architecture is proposed. The proposed architecture consists of butterfly units, signal generator, coherent sampling (ADC), FFT block, and calibration and spectrum analyzers.

PERFORMANCE RESULTS

Here CMOS based asynchronous system is designed with 5 stages pipelined architecture using HSPICE net list and its parameter optimization is proved over AFPL-PCR mechanism. The parallel prefix adder -based algorithm is highly suitable for VLSI implementation, since it is built using prefix and parallel computations. PCR enable Pipelined prefix adders are designed and its functionality is verified using Modelsim simulation.

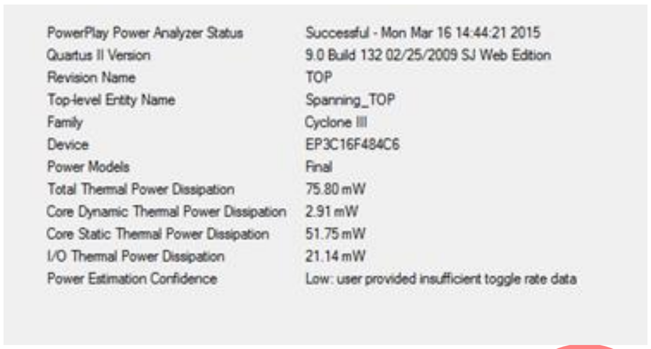


We used to prove the efficiency of pipelined architecture and PCR enabled system through EDA tool synthesis. To realize the low hardware Complexity VLSI implementation also using only binary shift and addition operations. Moreover, the computational accuracy can be selected based on the trade-off between the hardware Complexity and approximation error. In addition, since our proposed algorithm has uniform post-scaling factor, it is also suitable for scaled RCU implementation.

TYPE	AVG POWER	PEAKPOWER
AFPL. w/o PCM	69.11mW	137.29mW
AFPL. with PCM	14.20mW	25.30mW

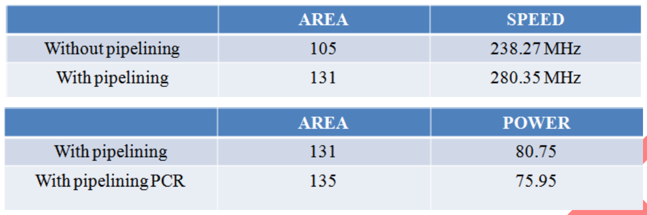
TABLE I

Efficiency of PCR mechanism through HPSICE simulated parameters



PowerPlay Power Analyzer Status	Successful - Mon Mar 16 14:44:21 2015
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	TOP
Top-level Entity Name	Spanning_TOP
Family	Cyclone III
Device	EP3C16F484C6
Power Models	Final
Total Thermal Power Dissipation	75.80 mW
Core Dynamic Thermal Power Dissipation	2.91 mW
Core Static Thermal Power Dissipation	51.75 mW
I/O Thermal Power Dissipation	21.14 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Fig 5 power dissipation report



	AREA	SPEED
Without pipelining	105	238.27 MHz
With pipelining	131	280.35 MHz

	AREA	POWER
With pipelining	131	80.75
With pipelining PCR	135	75.95

TABLE II

Efficiency of spanning and sparse based parallel prefix adder over pipelining

CONCLUSION

Here in this paper efficiency of AFPL-PCR over five stages pipelined circuits and its performance over power reduction is verified in both HSPICE simulation and FPGA based front end hardware synthesis .Number of gates required in hardware implementation, such as on an FPGA, is minimum as hardware complexity is greatly reduced compared to other complex synchronization techniques. And finally we synthesized successfully and efficiency of speed enhancement and power reduction in both sparse and spanning prefix adder scheme with QUARTUS II EDA tool.

REFERENCES

[1] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, “Leakage current mechanisms and leakage reduction techniques in deepsubmicrometer CMOS circuits,” Proc. IEEE, vol. 91, no. 2, pp. 305–327, Feb. 2003.

[2] N. H. E. Weste and D. M. Harris,CMOS VLSI Design: A Circuits and Systems Perspective, 4th ed. Upper Saddle River, NJ: Pearson Education, 2010.

[3] International Technology Roadmap for Semiconductor 2010 Update Overview. (2011) [Online]. Available: http://www.itrs.net/Links/ 2010ITRS/Home2010.htm

[4] Z. Chen, M. Johnson, L. Wei, and K. Roy, "Estimation of standby leakage power in CMOS circuits considering accurate modeling of transistor stacks," in Proc. Int. Symp. Low Power Electron. Design, 1998, pp. 239–244.

[5] S. Narendra, S. Borkar, V. De, D. Antoniadis, and A. Chandrakasan, "Scaling of stack effect and its application for leakage reduction," in Proc. Int. Symp. Low Power Electron. Design, 2001, pp. 195–200.

[6] IEEE Int. Symp. Circuits Syst., May 2007, pp. 3720–3723.

[7] T. Enomoto, Y. Oka, and H. Shikano, "A self-controllable voltage level (SVL) circuit and its low-power high-speed CMOS circuit applications," IEEE J. Solid-State Circuits, vol. 38, no. 7, pp. 1120–1126, Jul. 2003.

[8] M.-C. Chang and C.-C. Tsai, "Handshaking quasi-adiabatic logic," in Proc. Int. Midwest Symp. Circuits Syst., 2009, pp. 531–534. [27] Y. Moon and D. K. Jeong, "An efficient charge recovery logic circuit," IEEE J. Solid-State Circuits, vol. 31, no. 4, pp. 514–522, Apr. 1996.