ISSN:2231-0711
www.ijcset.net

A Low Power Asynchronous FPGA With Power Gating and Dual Rail Encoding

K.Naveena[1],N.Kirthika[2]
M.E VLSI design
Sri Ramakrishna Engineering College
Coimbatore, India
[1]k.naveena89@gmail.com, [2]kirthi.com@gmail.com

***Abstract*—A low power Asynchronous FPGA with LEDR encoding and 4-Phase dual Rail Encoding is designed in this paper. 4-Phase Dual Rail encoding is to achieve small area and LEDR encoding is to get high throughput and low power. LEDR encoding is done at input and 4-phase dual rail encoding is done at the output, which reduces power. Power gating is done to each Logic Block, which shuts down power to the block which is ideal. Power reduction is achieved by selectively setting the functional units into a low leakage mode when they are inactive. The circuit is simulated using Xilinx tool and programmed using verilog language.**

Keywords- Asynchronous Field-Programmable Gate Array(FPGA), Level Encoded Dual Rail (LEDR encoding), 4-Phase Dual Rail Encoding, Power Gating.

I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are widely used to implement special purpose processors. FPGAs are cost effective because interconnections of logic resources can be directly programmed by end users. Despite their design cost advantage, FPGAs impose large dynamic and standby power consumption overheads compared to custom silicon alternatives as in [1].

In FPGAs, the clock network occupies a large proportion of the dynamic power because it has significantly more registers than custom VLSIs. To solve the problem caused by clock, asynchronous FPGAs are proposed. Asynchronous FPGAs have mainly three advantages.

1. Low power consumption because of no dynamic power in inactive circuits.
2. Less emission of Electro Magnetic Interference (EMI) since each Logic Block (LB) operates at random points in time.
3. Robustness because there are automatically delay variations and there is no clock skew.

Asynchronous FPGAs based on dual rail encoding is proposed as in [2] and [3]. They use 4-phase dual-rail encoding because it has a simple hardware. However, a spacer must be inserted between two consecutive data values. So Level Encoded Dual Rail (LEDR) which requires no spacer is used here as in [4] and [5]. As a result, throughput and power consumption are small.

The remaining part of the paper is organized as follows. The corresponding work of this paper is detailed in section II. Section III deals with the overall architecture. The Logic Block implementation is given in section IV. The simulation result and conclusion are briefed in section V and VI respectively.

II. RELATED WORK

A. *Asynchronous FPGAs*

In asynchronous FPGAs, the clock and clock distribution network create several difficult challenges, namely dynamic power consumption and clock skew. References [6] and [7] proposed asynchronous FPGAs based on bundled-data encoding, the most common asynchronous encoding. In this encoding, delay elements are used for the control path. The use of delay elements limits the throughput. Especially, for FPGAs, since the data path is programmable, complex programmable delay elements are required. References [8] and [9] proposed asynchronous FPGAs based on dual-rail encoding which requires no delay insertion.

B. *Four phase dual rail Encoding*

Four-phase dual-rail encoding is one of the asynchronous FPGAs used, because of relatively small hardware cost. However, as shown in Fig. 1, in four-phase dual-rail encoding, a spacer must be inserted between two consecutive valid data values. This results in low throughput and high dynamic power consumption because of the large number of signal transitions. The use of spacer also increases the delay in the Logic Block.

The main feature is that sender sends the data in alternate phases of phase 0 and phase 1. The receiver knows the arrival of data value by detecting the change of either bit “0” or “1”.

The insertion of spacers make the coding law simple. This results in simple hardware for function unit. The hardware overhead of the four phase dual rail architecture is smaller than dual rail architecture. Table I shows the code table of 4-Phase dual rail encoding.

TABLE I. CODE TABLE OF 4-PHASE DUAL RAIL ENCODING

	Codeword (t, f)
Data 0	(0,1)
Data 1	(1,0)
Spacer	(0,0)

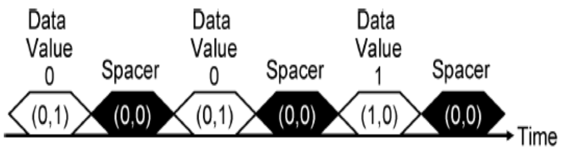

Fig. 1. Example of four phase dual rail encoding.

III. ARCHITECTURE

A. *Overview*

Fig. 3 shows the overall architecture of the proposed FPGA which has a mesh-connected cellular array based on a bit-serial architecture. The proposed architecture requires three wires: two for data and one for acknowledge (*ACK*).

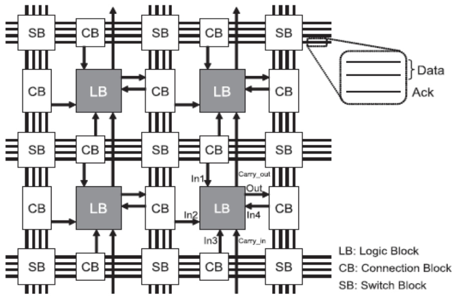

Fig. 3. Overall architecture.

B. *LEDR Encoding*

Asynchronous FPGAs based on LEDR encoding is proposed as in [10] and [11]. LEDR is one of several two-phase dual-rail encodings as in [12]. In LEDR encoding, no spacer is required. Table II shows the code table of LEDR encoding. In LEDR encoding, each data value has two types of code words with different phases. Fig. 2 shows the example where data values “0” “0” and “1” are transferred. The main feature is that the sender sends data values alternately in phase 0 and phase 1. Because no spacer is required, the number of signal transitions is half of four-phase dual-rail encoding. As a result, the throughput is high and the power consumption is small. Based on this observation, in the proposed FPGA, LEDR encoding is employed for implementing the asynchronous architecture to reduce the dynamic power.

TABLE II. CODE TABLE OF LEDR ENCODING.

		Code word (V, R)
Phase 0	Data 0	(0,0)
	Data 1	(1,1)
Phase 1	Data 0	(0,1)
	Data 1	(1,0)

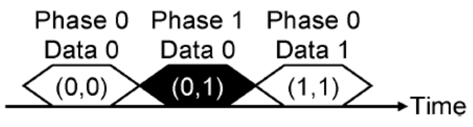

Fig. 2. Example of LEDR encoding.

Fig. 4 shows a simplified LB structure. As mentioned in previous section, 4-phase dual rail encoding achieves small area and low power for function unit, while LEDR encoding achieves high throughput and low power for data transfer. Based on this observation, as shown in Fig. 4, the proposed FPGA combines 4-phase dual rail encoding for implementing function units and LEDR encoding for the data transfer using programmable interconnection resources.

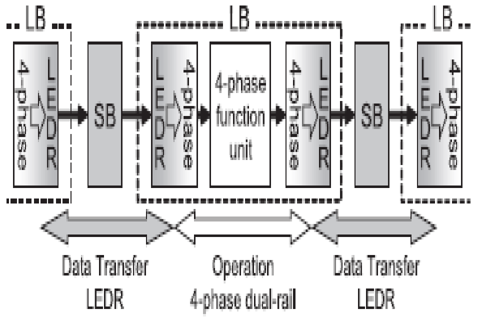

Fig. 4. Simplified structure of an Logic Block.

IV. CIRCUIT IMPLEMENTATION

A. *Logic Block Structure*

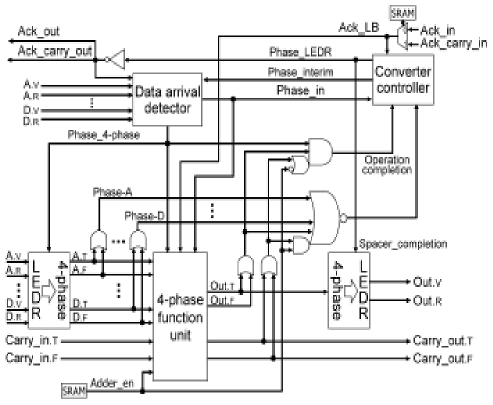

Fig. 5. Detailed structure of an LB.

The detailed structure of an LB is shown in Fig. 5. The LB mainly consists of a LEDR to 4-phase converter, a function unit, a 4-phase to LEDR converter, a data arrival detector and a converter controller.

The ACK out signal is distributed to the acknowledge lines of all the input data. The function unit is implemented using 4-phase dual rail encoding to achieve small area and low power.

Converters are used because LEDR encoding is used for data transfer between LBs to achieve high throughput and low power. The converter controller and a data arrival detector are used to control the LEDR to 4-phase converter and the 4-phase to LEDR converter.

The data in the carry chain is always encoded in 4-phase dual rail encoding and is not converted to LEDR encoding since the power and delay overheads of the protocol converters are much larger than those of the carry chain. As mentioned, the 4-phase to LEDR converter also performs as an output register. Since the output of the functional unit is always converted to LEDR encoding, the output is always latched. Therefore, large combination logic with more than four inputs cannot be implemented.

In the proposed FPGA, such a large combinational logic is partitioned into sub logics like pipelining. The structure of function unit is shown in Fig. 6. The function unit is implemented using 4-phase dual rail encoding. Like conventional FPGAs, the function units mainly consist of a LUT and carry logic. The LUT executes a four input and one output Boolean function.

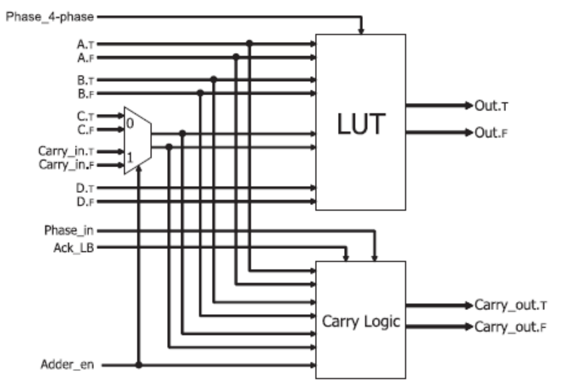

Fig. 6. Structure of function unit.

B. *Converter controller*

The structure of the proposed converter controller is shown in Fig. 7. The converter controller is used to generate the phase of LEDR encoding and generate a signal for data arrival detector to generate the phase of 4-phase dual rail encoding. The behavior of converter controller is as follows.

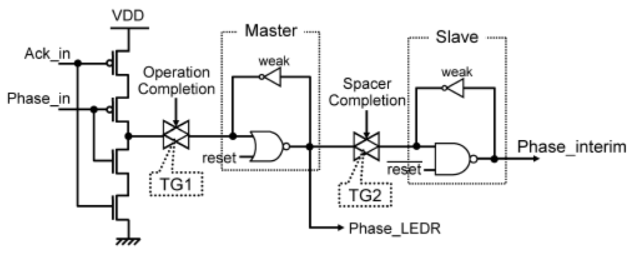

Fig. 7. Converter controller.

When the operation is complete, the first transmission gate (TG1) turns on. As a result, if the phases of the inputs (phase-in) match the required phase from next LB (ACK_in) and the operation is completed, phase_LEDR turns to the same as phase_in. To retain phase_LEDR when the operation is not completed, an SRAM cell (Master) is used.

After the operation completion, a spacer for 4-phase dual rail encoding is generated for the function unit. To guarantee the new data have not been input to the function unit before the spacer has been made, phase_interim is used. Phase_interim is used to generate the phase of 4-phase dual rail encoding and as an input of the data arrival detector. To retain phase_interim when the spacer has not been made, an SRAM cell (slave) is used.

C. *Data arrival detector*

The structure of the proposed data arrival detector is shown in Fig. 8. A Muller's C-element as in [13] is used in the data arrival detector. The Muller's C-element is a state holding element. When all inputs are "0" the output is set to "0" and when all inputs are "1" the output is set to "1". For other input combinations the output does not change. In Fig. 8, phase_in is used as an input of the converter controller. It changes when the phases of all inputs match. It remains in this state until the phases of all inputs transit to the other state. Signal phase_4-phase is the phase of dual rail encoding. It is used as a control signal of the LEDR to 4-phase converter and as a pre-charge signal of the function unit.

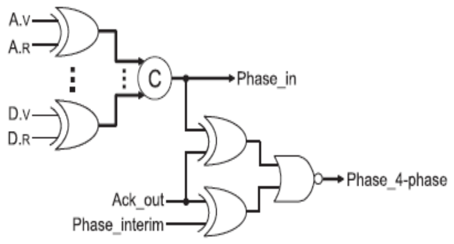

Fig. 8. Data arrival detector.

D. *LEDR to 4-phase converter*

The structure of the LEDR to 4-phase converter is shown in Fig. 9. The LEDR to 4-phase converter used in the proposed FPGA is a conventional one because the overhead of the conventional LEDR to 4-phase converter is small. The converter generates two bits for one bit data value.

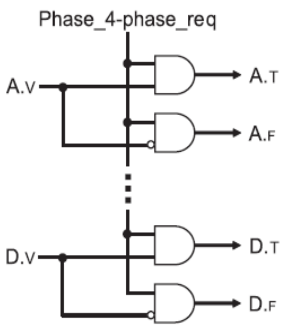

Fig. 9. LEDR to 4-phase converter

In the converter, the true bit (T) and the false bit (F) of 4-phase dual rail encoding are generated from the value bit (V) of LEDR encoding using phase_4-phase. If phase_4-phase is "0", all the outputs are "0". That is the output represents a spacers in 4-phase dual rail encoding. If phase_4-phase is "1", T is set to V, while F is set to V bar. That is the output represents a valid data in 4-phase dual rail encoding.

E. *4-phase to LEDR converter*

The proposed 4-phase to LEDR converter is shown in Fig. 10. It also performs as an output register. It consists of two modules. One is Out.V and other is Out.R generating circuit. The double edge trigger flip flop is essentially required for LEDR encoding data storage as in [10]. Therefore, the hardware overhead of the converter is only this Out.R generating circuit. When the operation is completed, phase-LEDR changes. Then Out.T is stored in an SRAM cell and Out.V turns to the same as Out.T. The upper SRAM cell retains Out.V if phase-LEDR is "0" and the lower SRAM cell retains Out.V if ohase LEDR is "1". When phase-LEDR is "0", Out.R is set to the same as Out.V. This is, the outputs represent a data of phase "0". When phase-LEDR is "1", Out.R is set to the same as Out.V inverted.

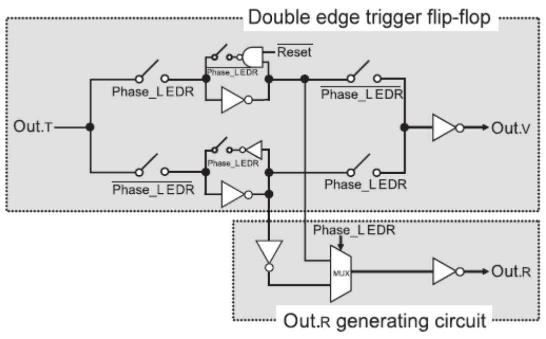

Fig. 10. 4-phase to LEDR converter.

V. COMPARISON RESULT

The comparison result for power analysis of the conventional Logic Block and the Proposed Logic Block is shown in Table III. The proposed Logic Block has 3mw less power consumed than the conventional Logic Block.

TABLE III. POWER ANALYSIS

	CONVENTIONAL LOGIC BLOCK	PROPOSED LOGIC BLOCK
Power	30mW	23mW

The total Logic utilization is shown in Table IV. The number of LUTs was found to be same, number of Input Output Blocks (IOBs) was reduced in the proposed architecture and delay was increased.

TABLE IV. AREA ANALYSIS

LOGIC UTILIZATION	CONVENTIONAL LOGIC BLOCK	PROPOSED LOGIC BLOCK
Number of bonded IOBs	10	3
Delay(nS)	4.040	4.910

A. *Implementation of Ripple Carry Adder*

The ripple Carry Adder is implemented using proposed Logic Block is shown in Fig. 11. The Function unit acts as a Full Adder. The LUT produces the sum output and the carry logic produces the carry output. So a single Logic Block implements a Full Adder. This Ripple Carry Adder using proposed Logic Block is compared with the Ripple Carry Adder implemented using conventional Logic Block and an ordinary Ripple Carry Adder.

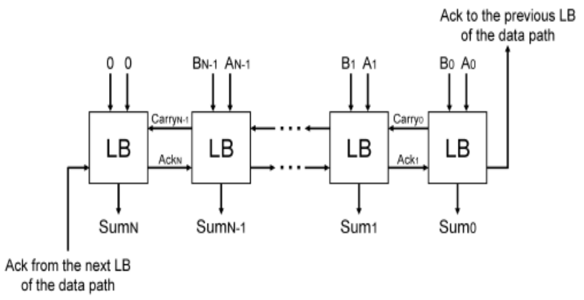
Ack to the previous LB of the data path
0 0
BN-1 AN-1
B1 A1
B0 A0
LB
CarryN-1
AckN
Carry0
Ack1
SumN
SumN-1
Sum1
Sum0
Ack from the next LB of the data path

Fig. 11. Implementation of Ripple Carry Adder using Logic Blocks

The comparison result for power analysis for implementation of Ripple Carry Adder is shown in Table V. Comparison is done for a normal Ripple Carry Adder, Ripple Carry Adder using conventional Logic Block and Ripple Carry Adder (RCA) using Proposed Logic Block. The RCA using proposed Logic Block consumes less power compared to ordinary and conventional Logic Block.

TABLE V. POWER ANALYSIS FOR RIPPLE CARRY ADDER

RIPPLE CARRY ADDER	ORDINARY	USING CONVENTIONAL LOGIC BLOCK	USING PROPOSED LOGIC BLOCK
Power	54mW	53mW	31mW

VI. CONCLUSION

The conventional Logic Block has the problem in detecting the data arrival for each Logic Block. So when data are detected in inaccurate timings, by which power consumption increases. This results in low throughput. To overcome this problem, the proposed architecture which detects the data in accurate timings using the data arrival detector is used here. So reduction in power is achieved.

ACKNOWLEDGEMENT

The authors thank the Management and Principal of Sri Ramakrishna Engineering College, Coimbatore, for providing excellent computing facilities and encouragement.

REFERENCES

[1] H. Z. V. George and J. Rabaey, “The design of a low energy FPGA,” in *Proc. Int. Symp. Low Power Electron. Des.*, CA, Aug. 1999, pp. 188–193.

[2] Y. Zhang, J. Roivainen, and A. Mammela, “Clock-gating in FPGAs: A novel and comparative evaluation,” in *Proc. EUROMICRO Conf. Digit. Syst.Des.*,2006, pp. 584–590.

[3] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger, “A 90 nm low-power FPGA for battery-powered applications,” in *Proc. FPGA*, Feb. 2006, pp. 22– 24.

[4] Xilinx Inc., San Jose, CA, “Spartan-3 FPGA family datasheet,” 2009. [Online]. Available: http://www.xilinx.com

[5] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual: For System-on-Chip Design*. New York: Springer, 2007.

[6] K. Maheswaran and V. Akella, “PGA-STC: Programmable gate array for implementing self-timed circuits,” *Int. J. Electron.*, vol. 84, no. 3, pp. 255–267, 1998.

[7] R. Payne, “Self-timed FPGA systems,” in *Proc. Int. Workshop Field Program. Logic Appl.*, 1995, pp. 21–35.

[8] J. Teifel and R. Manohar, “An asynchronous dataflow FPGA architecture,” *IEEE Trans. Computers*, vol. 53, no. 11, pp. 1376–1392, Nov. 2004.

[9] R. Manohar, “Reconfigurable asynchronous logic,” in *Proc. IEEE Custom Integer. Circuits Conf.*, Sep. 2006, pp. 13–20.

[10] M. Hariyama, S. Ishihara, C. C. Wei, and M. Kameyama, “A field programmable VLSI based on an asynchronous bit-serial architecture,” in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Jeju, Korea, Nov. 2007, pp. 380–383.

[11] M. Hariyama, S. Ishihara, and M. Kameyama, “Evaluation of a field programmable VLSI based on an asynchronous bit-serial architecture,” *IEICE Trans .Electron*, vol. E91-C, no. 9, pp. 1419–1426, 2008.

[12]. M. E. Dean, T. E. Williams, and D. L. Dill, “Efficient self-timing with level-encoded 2-phase dual-rail (LEDR),” in *Proc. Univ.California/ Santa Cruz Conf. Adv. Res. VLSI*, 1991, pp. 55–70.

[13] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective*. Norwell, MA: Kluwer, 2001.