Low Power Logical Element for FPGA Fabric

Mouna Nakkar and Paul Franzon

Mouna Nakkar is with the University Of Sharjah and Paul Franzon is with North Carolina State University

ABSTRACT

Logical Element (LE) is the basic building block of FPGA fabric or any re-configurable computing machines. Logical Element basically consists of look-up tables [1]. This paper shows a low power LE. The Low power is achieved by taking advantage of the commutative property of operations. This property will allow the data to be re-organized such that there will be separate paths for logic state 0 and logic state 1. The approach is targeted to reduce switching activity when possible. This paper shows 28% increase in power savings.

1. INTRODUCTION

Low power CMOS is becoming a significant part of integrated circuits performance especially for mobile technology and portable devices. Low power systems prolong the lifetime of these devices. Low power design is not only desirable for portable devices, but also for none-portable machines. High power consumption Integrated Circuits increases heat density in the device package. The increased heat density has negative effects on performance. This prompted researchers and leading microprocessor companies to spend big efforts on low energy design.

Most low power CMOS designers solve the problem at a gate level. For example, low power is achieved by either reducing the voltage level, reduced swing logic [3], or by re-distributing the charge. While reducing the voltage level has its advantages in lowering the total power consumption, it has huge disadvantages in terms of circuit sensitivity to noise. Low voltage level will reduce the noise margin for a CMOS circuit and affect the switching level of threshold voltage. Reduced swing logic is sensitive to loading capacitance and threshold voltage variations. Charge re-distribution circuits have disadvantages in design complexity and leakage currents. The leakage currents will reduce the voltage swing and may result in wrong logic values.

This paper approaches low power design by taking advantage of the commutative properties of operations. Basically this an architectural approach rather than a gate level approach. In this design, low power is achieved by reducing the switching power [4].

This is achieved by separating logic paths, i.e. trying to keep a path for logic state 0 and another for logic state 1. The result of the operation will not change whether the paths are separated or not.

This paper shows a low power design for Logical Element (LE). LEs are the basic building block of any re-configurable devices such as Xilinx [5] and Altera chips [6]. Further, LEs are used for the Dynamically Programmable Cache DPC [1].

The flow of the paper will be as follows: section 2 shows a block diagram of typical LE. Section 3 shows a block diagram of low power LE. Section 4 shows simulation results. Section 5 is the concluding remarks.

2. LOGICAL ELEMENT

The Logical Element is the basic building block of re-configurable computing devices such as FPGA chips. This LE has the ability to be programmed to implement several functions logical or arithmetic. Logical functions such as AND, OR, XOR…etc. Arithmetic such as ADD with Carry and Multiply. This is based on a look-up table as shown in Fig. 1. The logical element as can be seen from the Figure is a two input with a carry element that implements the function using the SRAM look-up table. The Logical Element produces two outputs one is Sum and the other called Carry. The Carry output is meaningful only for arithmetic operations. The four flip-flops are used for storage. Typical FPGA LE may not use four flip-flops instead they could use two or one depending on the chip. Because this paper is an extension of the DPC project, the LE used has four storage elements.

The logical operations are: AND, OR, NAND, NOR, XOR, XNOR, and INV. The arithmetic operations are: ADD, COMPARE EQUAL, COMPARE LESS THAN, COMAPRE GREATER THAN, SHIFT LEFT, SHIFT RIGHT, and ONE BIT MULTIPLY. The "Function" control bits selects the one the above operations depending how this LE been programmed. Data is three bit input bus that represents: two inputs and a carry_in. "Output select" selects one of the outputs of the four flip-flops or directly from the look-up table.

0-7803-7573-4/02/$17.00

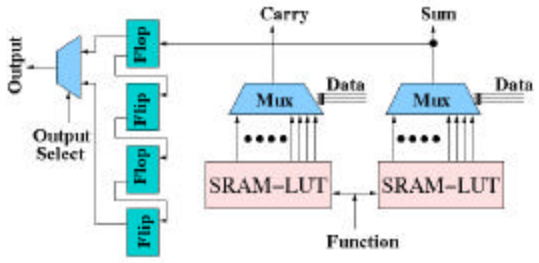

Fig. 1. Block diagram of DPC Logical Element.

3. LOW POWER LOGICAL ELEMENT

The approach for low power in this paper is at the architectural level where the cumulative property is utilized.

3.1. Commutative Property of operations

All logical operations mentioned above are two inputs functions only. Further, all of these operations have commutative property, meaning that D_0 AND D_1 will result the same as D_1 AND D_0, D_0 OR D_1 will result the same as D_1 OR D_0.etc. The arithmetic operations, on the other hand, have two inputs and a carry. Only three functions posses the commutative property: ADD, COMPARE, and one bit MULTIPLY. The other functions such, as SHIFT LEFT AND RIGHT does not hold the property.

3.2. Low Power Logical Element

Low power is achieved by taking advantage of the commutative property to reduce switching activity in the LE cell. The idea is to always try to have a fixed path for inputs with logical state 0 and another path for input with logical state 1. This reduces switching activity, if possible, in the circuit. For example, the following table shows an ADD operation re-organized such that one path is always directed to carry state 0 and another carries state 1.

TABLE I
Two-bit with Carry ADD operation re-organized to suit separate 0 and 1 paths.

Original			Re-organized Data			Output	
D_0	D_1	C_{in}	D_0-new	D_1-new	C_{in}-new	Sum	C_{out}
0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0
0	1	0	1	0	0	1	0
0	1	1	1	1	0	0	1
1	0	0	1	0	0	1	0
1	0	1	1	1	0	0	1
1	1	0	1	1	0	0	1
1	1	1	1	1	1	1	1

It is shown that the new data reserves C_{in} path for logic state 0 and the D_0 path for logical state 1. Channeling data will reduce the switching activity of data in C_{in} path and D_0 path if possible. The separation will add new logic as shown in Fig. 2. The added logic combination is very simple where:

$$D_{0\_new} = D_{0\_old} + D_{1\_old} + C_{in\_old}$$
$$D_{1\_new} = D_{0\_old} \bullet D_{1\_old} + C_{in\_old}\left(D_{o\_old} \oplus D_{1\_old}\right)$$
$$C_{in\_new} = D_{0\_old} \bullet D_{1\_old} \bullet C_{cin\_old}$$

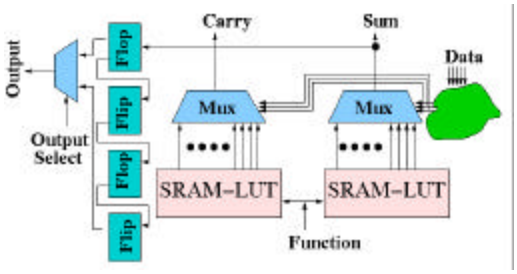

Fig. 2. Low power LE block diagram.

The added logic overhead is shown in Fig. 3. The new data is re-generated from old data.

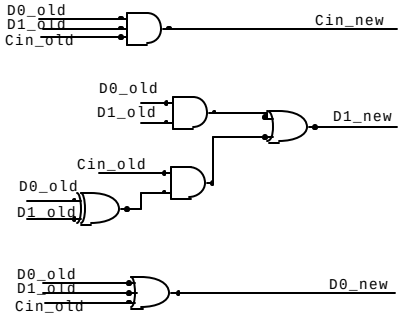

Fig. 3. The logic for data re-organization.

4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

The low power logical element is implemented using 0.5µm CMOS process and simulated with Spectre. The simulation includes generating an input set of 500 random vectors, meaning that 500 different inputs are produced randomly. New vectors applied every 5ns period over 1.0 ms spectrum.

Table 2 shows the average power consumption for Low Power LE (LPLE) and conventional LE. The table shows up to 28% in RMS power consumption.

TABLE II
Average power consumption comparisons for LPLE and LE.

Operation	RMS_power (mW) LE	RMS_power (mW) LPLE	% Savings
AND	2.825	2.013	28.7
OR	2.812	2.019	28.2
NAND	2.788	2.006	28.0
NOR	2.856	2.036	28.7
XOR	2.935	2.199	25.0
ADD	3.422	2.793	18.3
MULT	3.145	2.265	27.9
COMP	3.288	2.634	19.9

The additional logic increases both area and delay. As for the area, the logic shown in Fig. 3 is 57μm^2 and the total area of LE without this addition is 512μm^2. The percentage increase in area is 11%. The total delay increases as well. The delay for the LE is 1.91ns. With the additional logic, the total delay becomes 2.25ns, which is 18% increase. Fig. 4 shows a picture of the Low Power Logical Element.

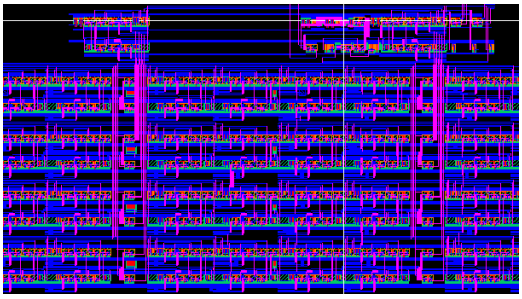

Fig. 4. Circuit Diagram of Low Power LE and conventional LE

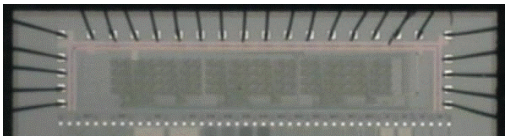

Fig. 5. Picture of Low Power LE implemented on 0.5mm HP process.

5. CONCLUDING REMARKS

This paper shows Low Power Logical Element with up to 28% saving. This is achieved by taking advantage of the commutative property of logical and arithmetic functions to reduce switching activity. Because of this property, it is possible to separate data paths trying to have one path always for logical state 0 and one path for logical state 1. This will reduce the switching activity.

The data will first be exposed to the separating logic that will add delay. The implementation shows that the total delay increases by a factor of 18% and the area increase is 11% as well. The low power logical element is implemented using a 0.5µm process.

6. REFERENCES

[1] A.B. Smith, C.D. Jones, and E.F. Roberts, "Article Title," *Journal,* vol. 1, no. 3, pp. 1-10, Mar. 2000.

[2] C.D. Jones, A.B. Smith, and E.F. Roberts, *Book Title*, Publisher, Location, 2000.

[3] A.B. Smith, C.D. Jones, and E.F. Roberts, "Article Title," in *Proc. IEEE Int. Symp. Circuits and Systems,* Monterey, CA, pp. 11-14, June 1998.

[1] M. Nakkar, J. Harding, D. Schwartz, P. Franzon, and T. Conte, "Dynamically Programmable Cache," in *SPIE Configurable Computing: Technology and Applications,* Boston, MA, Vol. 3526, pp. 218-226, 1998.

[2] M. Nakkar, D. Bentlage, J. Harding, D. Schwartz, P. Franzon, and T. Conte, "Dynamically Programmable Cache Evaluation and Virtualization," in the *1999 Seventh ACM International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, February 1999.

[3] A.M. Fahim, and M.I. Elmasry, "Low-Power High-Performance Arithmetic Circuits and Architectures," in *IEEE J. Solid-State Circuits*, vol. 37, No. 1, pp. 90-94, January 2002.

[4] A.M. Shams and M.A. Bayoumi, "A Novel High-Performance CMOS 1 Bit Full-Adder Cell," in *IEEE Tran. on Circuits and Systems-II: Analog and digital signal processing*, vol. 47, No. 5, pp. 478-481, May 2000.

[5] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design.* Reading, MA: Addison-Wesley, 1993.

[6] Xilinx, *Xilinx, The Programmable Logic Data Book*, Xilinx Inc., 1994.

[7] Algotronix, *Algotronix Data Book*, Edinburgh, UK CAL1024 Preliminary Data Sheet, 1988.