Revisiting the Cascade Circuit in Logic Cells of Lookup Table Based FPGAs

Nam-Sung Woo
AT&T Bell Laboratories, 2A-230
600 Mountain Ave., Murray Hill, New Jersey, 07974
nswoo@research.att.com

This paper shows that cascade circuits in the logic cells of all current lookup table based FPGAs support only linear cascading chain and, as a result, contribute to long cascading delay. We present an enhanced cascade circuit that will reduce cascading delay significantly: from linear time to log time in terms of the number of logic cells cascaded. We show that the additional area for the new cascade circuit is very small. We discuss an interaction between architecture design decision and CAD (in particular, placement) for the design of dedicated routing structure for cascade signals between logic cells. We illustrate the advantage of the new cascade circuit with an example of 32-bit equality checking circuit.

1 Introduction

FPGA is an off-the-shelf VLSI chip that contains programmable logic cells and programmable routing resource. FPGAs can be divided into two classes — coarse-grain and fine-grain — based on the logic capacity of each logic cell. The coarse-grain FPGA contains logic cells each of which has rich functionality, while the fine-grain FPGA contains logic cells with limited functionality. Coarse-grain FPGAs include AT&T's ORCA [1, 2, 3], Xilinx's XC4000 [4, 5], Altera's Flex 8000 [6], and XC/ATT 3000 [7, 8]. As an example, Figure 1 shows the combinational portion of a logic cell, called PLC (programmable logic cell), of the ORCA FPGA. (We omitted four flipflops and some logic associated with them.) It contains four lookup tables, T0 – T3, each of which is a 16-bit memory, along with a fast carry circuit, multiplexers and gates.

Most of the coarse-grain FPGAs contain a cascade circuit in each logic cell to collect partial results. However, all existing FPGAs provide only the *linear* cascade circuit which results in long cascading delay: see the next section.

The design technique of collecting partial results by cascading is frequently used for implementing both datapath modules and control (i.e., random) circuits in FPGAs. Reducing cascading delays in FPGA implementations is important to speed up the implemented (application) circuits.

In this paper, we present a new cascade circuit that supports a *tree structure* in collecting partial results. The new circuit reduces the cascading delay significantly: from linear to "log" in the number of logic cells cascaded. We show the advantage of the new cascade circuit by an

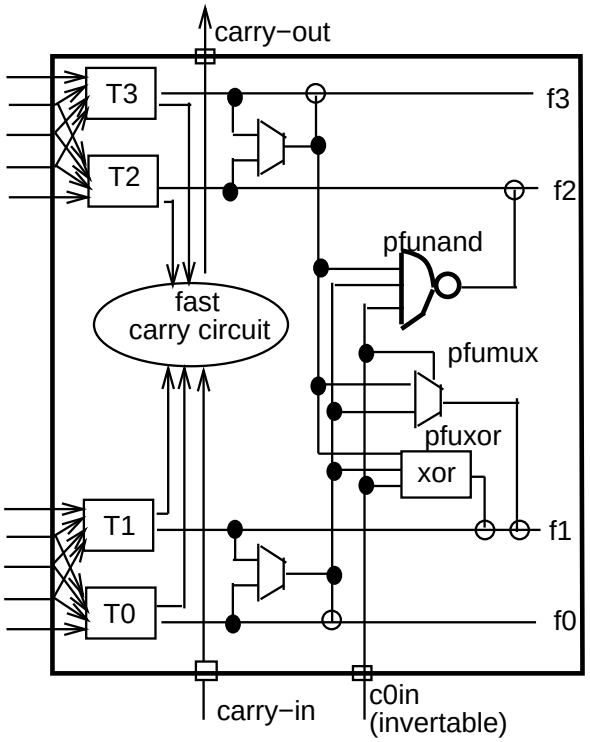

Figure 1: Combinational Portion of A Logic Cell of ORCA FPGA

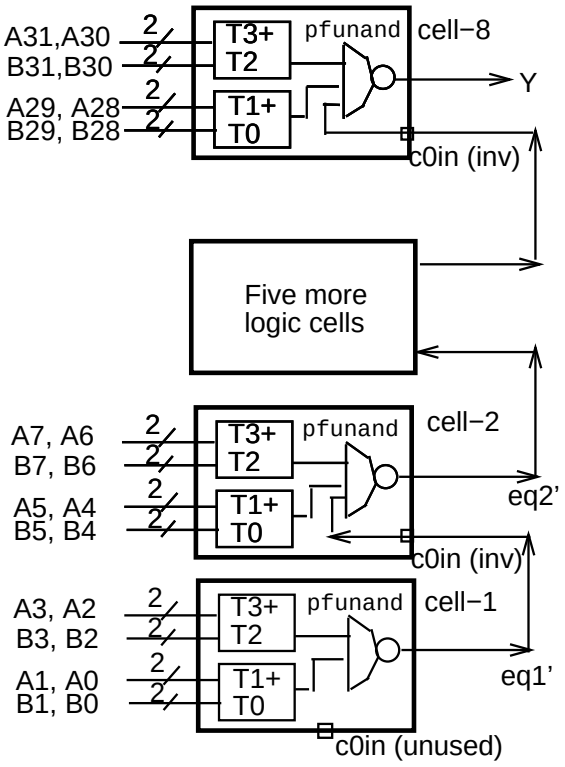

Figure 2: An area-optimal implementation of 32-bit equality checking circuit in ORCA

example, 32-bit equality checking circuit. We also discuss an interaction between FPGA architecture design and FPGA CAD in terms of the dedicated routing wires for cascading signals.

2 A Survey of Cascading Circuits in Current FPGAs

Let us consider the cascade circuit in the logic cells of three commercial lookup table-based FPGAs: ORCA, XC4000 and Flex 8000.

The ORCA FPGA contains a 3-input NAND gate, called "pfunand", in each logic cell: see Figure 1. Two inputs of the `pfunand` gate come from internal lookup tables of the same logic cell, and the other input comes from an external source (through `c0in` port). The `pfunand` gate and its direct input from `c0in` are used as a cascade circuit as explained below.

Suppose we have two 32-bit data, A[31:0] and B[31:0], and we want to produce an *active-low* signal, Y, when the two data are equal. The best ORCA implementation in terms of area is shown in Figure 2, in which eight logic cells are used. In each logic cell, four lookup tables are divided into two groups and each group compares a pair of two bits (e.g., A1, A0 and B1, B0). Each lookup table produces "1" output if the input pair is equal. The `pfunand` gate in cell-1 produces an active-low equal signal , eq1', for the least significant four-bit pair. The eq1' signal is inverted when it goes through the `c0in` port of cell-2. (This inversion comes for free in the ORCA FPGA.)

The `pfunand` gate in cell-2 cascades the above equal signal with two equal signals for the next four bits (i.e., A7-A4 and B7-B4) produced by the lookup tables in the same cell. The output of the `pfunand` gate, eq2', is an active-low signal that shows the equality of a pair of 8 bits (A7-A0 and B7-B0). Finally, the `pfunand` gate in cell-8 combines the equal signal for low order 28-bit pair with two equal signals for the most significant 4-bit pair to produce the final equality signal, Y. Note that the `pfunand` gates in cell-2 through cell-8 collect (or cascade) partial results to produce the final result. [1]

The cascading pattern in Figure 2 is "linear" because one needs N-1 cascading stages to collect partial results from N logic cells.

Now, let us consider the XC4000 FPGA. The combinational circuit portion of a logic cell, called CLB (configurable logic block), of the XC4000 FPGA is shown in Figure 3. (We omitted two flipflops in this figure.) There are three lookup tables (LUTs): F, G and H. The first two are 16-bit RAMs, while the last one is an 8-bit RAM. The H-LUT can be used for cascading partial results; see p. 5 of XC4000 data book [5]. Note that the H-LUT has three input signals, two of which come from internal (i.e., F and G) LUTs and the other input comes from outside of the logic cell. That is, the input structure of the H-LUT is the same as that of the `pfunand` gate in ORCA. Therefore, the H-LUT also provides only the linear cascading capability.

An area optimal implementation of the 32-bit equality checking circuit using XC4000 FPGA would require eight CLBs with the same circuit structure as the one in Figure 2. The only difference would be the polarity of the signals generated from the CLBs, i.e., the equal signal from

[1] Of course, one can combine intermediate results by using an extra logic cell; however, this solution uses more logic cells and requires more routing resource than the above solution.

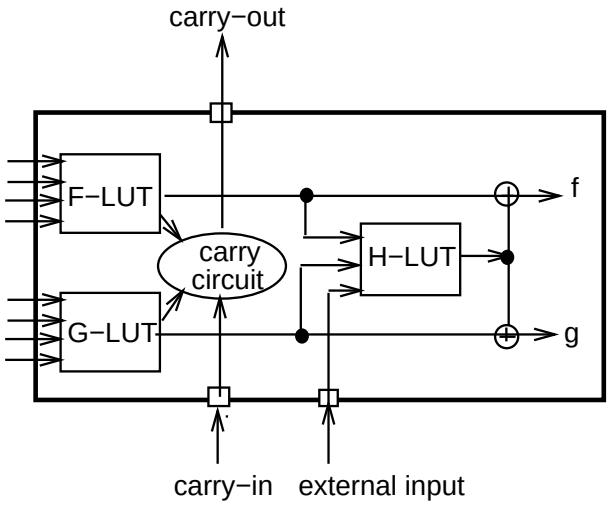

Figure 3: Combinational Portion of XC4000 Logic Cell (CLB)

each CLB can have either active-high or active-low polarity.

A logic cell, called logic element (LE), of the Flex 8000 FPGA contains a cascade circuit which is either a 2-input AND gate or a 2-input OR gate. One input comes from the internal LUT, a 16-bit RAM, and the other comes from outside of the logic cell. (The latter input comes from a neighboring logic cell through a dedicated wire.) As a result, this logic cell also provides only the linear cascading capability.

In summary, the logic cells of the above three lookup table-based FPGAs contain cascade circuits that support only linear cascading.

3 Tree-structured Cascade Circuit

After giving a motivation, we will describe the new cascade circuit and its advantage. We will use the ORCA FPGA in Figure 1 for our discussion.

3.1 Motivation

When an FPGA implementation of an application circuit contains cascaded circuit(s), the delay of the cascading chain usually determines the speed of the implemented circuit.

For instance, consider the circuit in Figure 2. Let D_t denote the delay of a lookup table, D_c the delay of the cascade circuit inside a logic cell. (D_c is the delay from either the output of a lookup table or the `c0in` input port to the output of the `pfunand` gate, i.e., cascade circuit element). Also, let D_r denote the routing delay of a cascade signal between two logic cells; for simplicity, we assume that D_r is the same for every cascade signal. Assuming all input signals (A[31:0] and B[31:0]) arrive at the same time, we get the total delay from input to output, T_8, as follows: $T_8 = D_t + D_c + 7(D_c + D_r)$.
The first two terms are for the lookup table and the `pfunand` gate, respectively, in cell-1. The third term is for cascading in the other seven cells.

In general, for a linear cascaded circuit of N logic cells, the total delay, T_N, is
$T_N = D_t + D_c + (N-1)(D_c + D_r)$
Thus, the total delay is dominated by the delay of the cascading path.

We will describe a way of reducing T_N by reducing the co-efficient of the third term (i.e., $N-1$) into a "log" term. We will also describe a way of reducing D_r.

3.2 A Balanced-Tree Cascade Circuit

We propose that the cascade circuit element in each logic cell have two or more inputs from external source (i.e., from outside of the logic cell). Considering the area overhead associated with a new (input) port, we think two external inputs to a cascade circuit element are reasonable. Having two external inputs to the cascade element, we can build a *balanced tree* instead of linear chain for cascading, which will reduce cascading delay significantly.

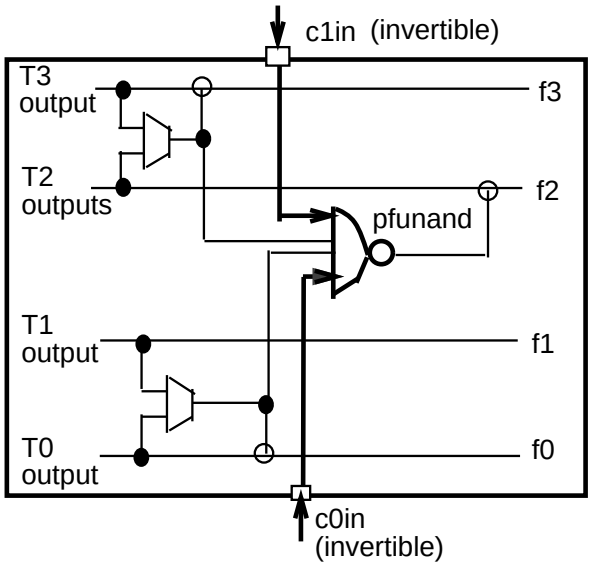

Figure 4: An Enhanced Cascade Circuit for ORCA Logic Cell

As a concrete example of the proposed circuit, we show an enhanced cascade circuit of ORCA logic cell in Figure 4: compare it with the corresponding portion of Figure 1. The new `pfunand` gate is a 4-input NAND gate: two inputs come from internal LUTs and the other two come from external sources (`c0in` and `c1in`).

Other FPGAs can also be equipped with the proposed cascade circuit in a similar way. For example, a logic cell of XC4000 FPGA can be upgraded to have a 16-bit H-LUT and an extra input connection to it. As another example, a logic cell of Flex 8000 FPGA can be upgraded to have 3-input gate (instead of 2-input gate) in its cascade circuit portion and a new connection to the gate.

Another way of reducing T_N in Section 3.1 is to reduce D_r by using "dedicated routing wires" for cascading signals between logic cells. While the Flex 8000 FPGA uses such dedicated cascading

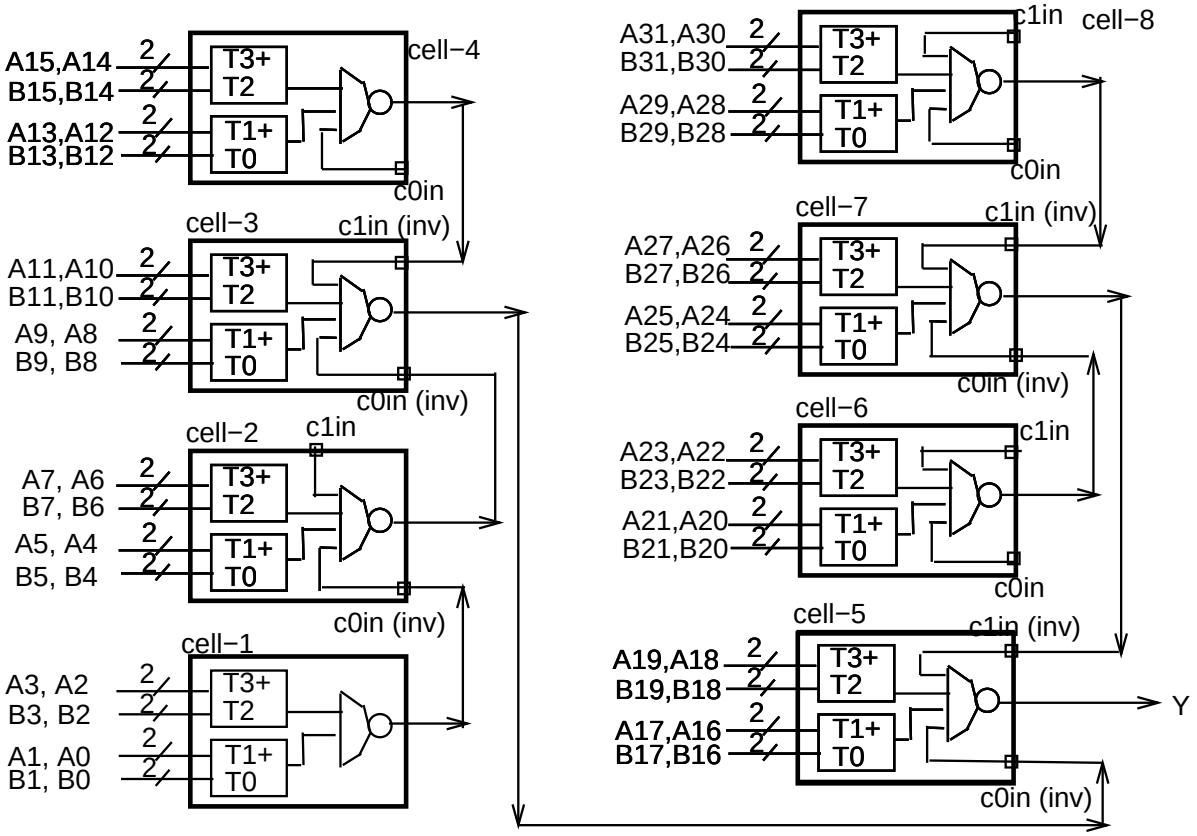

Figure 5: An Implementation of 32-bit Equality Checking by Proposed New Logic Cells of ORCA

wires, ORCA and XC4000 FPGAs do not. More details of the dedicated cascading wires are described in Sections 4 and 3.3.

3.3 An Analysis

The proposed cascade circuit allows us to have a cascading tree rather than a cascading chain. The total delay of cascaded circuit with N logic cells, U_N, is

$U_N = D_t + D_c + log_2 N(D_c + D_r)$

Comparing U_N with T_N, we can see that the former grows much slowly than the latter as N grows.

As an example, let us reimplement the 32-bit equality circuit described in Section 2. Figure 5 shows a new implementation using the logic cell structure in Figure 4. The longest path consists of cell-1, cell-2's `pfunand` gate, cell-3's `pfunand` gate, and cell-5's `pfunand` gate. The total delay of this implementation is

$U_8 = D_t + D_c + 3(D_c + D_r)$

Comparing this delay with the above T_8, we can see the coefficient of the third term is reduced from 7 to 3.

Considering the current technology, we can assign values to the above variables as follows : D_t = 4 ns, D_c = 2 ns, and D_r = 1.5 ns. (The D_c contains not only the delay of the pfunand gate, but also the delay from either LUT output or c0in to the input of the gate.) Replacing these numbers to T_8 and U_8, we get,

$T_8 = 4 + 2 + 7(2 + 1.5) = 30.5ns$
$U_8 = 4 + 2 + 3(2 + 1.5) = 16.5ns$
Therefore, the proposed cascade circuit contributes to 46% speed-up in an implementation of the 32-bit equality checking circuit.

Now, let us assume that we use dedicated wires for the cascading signals between logic cells, as described in Section 3.2. This will reduce the value of D_r; let us say the new D_r = 0.5 ns. Then, the updated value of U_8 is 13.5 ns; we got 56% speed-up compared to the above T_8.

3.4 Cost of the New Cascade Circuit

The additional (silicon) area introduced by the proposed cascade circuit consists of two parts:

1. Additional area for the new cascade circuit element (e.g., `pfunand` gate of ORCA or H-LUT of XC4000),

2. Additional area for an extra input line to the cascade circuit element.

The first part varies with the architecture of FPGA. For ORCA, it is an increase from a 3-input NAND gate to a 4-input NAND gate: a very small increase. The second part would vary with the routing architecture of FPGA. If we use regular routing resource or existing dedicated routing resource (e.g., carry wires) for cascading signals, the cost of the second part is close to zero. If we use new dedicated routing resource, the cost of the second part would include the area for two multiplexers and a few bits of configuration RAM. (See the next section for more details.)

4 Routing Structure for Dedicated Cascading Wires

As mentioned above, the dedicated routing wires for cascading signals between logic cells would futher reduce the total cascading delay. In this section, we will discuss a relationship between routing architecture design and CAD. In particular, we will consider two routing structures for dedicated cascading signals and their impact on the placement (of logic cells).

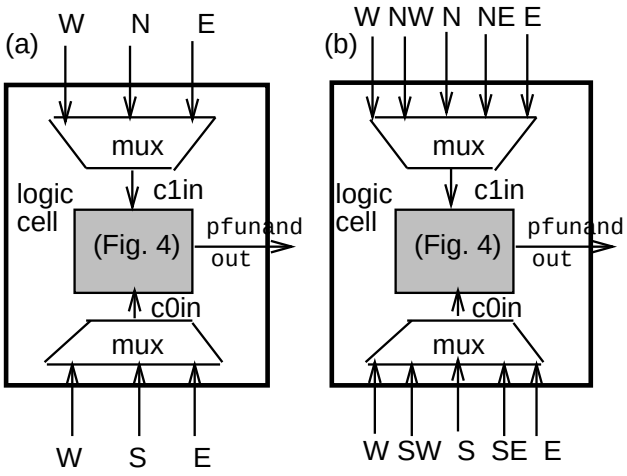

Figure 6: Two dedicated routing structures for cascading signals

First, let us assume that `c1in` signal of a logic cell comes from three neighbor logic cells (North, East and West) and that `c0in` signal comes from three neighbor logic cells (South, East and West). Figure 6(a) shows this routing structure. We need two 3-to-1 multiplexers for `c0in` and `c1in` ports. The number of fanout of the `pfunand` gate of a logic cell is 6.

Second, let us assume that `c1in` signal comes from five neighbor logic cells (North, East, West, Northeast, Northwest) and that `c0in` signal comes from five neighbor logic cells (South, East, West, Southeast, Southwest). Figure 6(b) shows this structure. We need two 5-to-1 multiplexers for `c0in` and `c1in` ports. The number of fanout of the `pfunand` gate of a logic cell is 10.

Comparing the above two routing structure, we can tell that the former is more area efficient. However, the former may make placement (of

logic cells) impossible in some cases, as described below.

Suppose that we want to place the eight logic cells in Figure 5 into nine logic cells of 3x3 array. Let us assume that the least significant 12 bits of A and B data must be placed at the leftmost three logic cells because of a user's constraint (e.g., pin or routing constraint). Then, we cannot find a valid placement with the routing structure of Figure 6(a). On the other hand, if the routing structure of Figure 6(b) is used we can find valid placements one of which is shown in Figure 7.

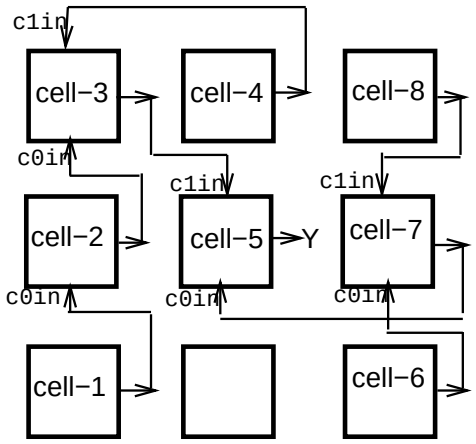

Figure 7: Placement examples

5 Concluding Remarks

We presented a new cascade circuit that allows tree-structured collection of partial results. The main idea is to provide (at least) two external inputs to the cascade circuit element in each logic cell. We showed how two existing FPGAs can be changed to incorporate the new cascade circuit. We also showed that the amount of speed-up by the proposed circuit is significant and that the cost of the proposed circuit is very small. We discussed an interaction between architecture decisions of FPGA and CAD.

The new cascade circuit element (e.g., 4-input NAND for ORCA, and 16-bit H-LUT for XC4000) would also enhance the efficiency of implementing random logic circuits. The work of measuring the amount of the enhancement is under way.

References

[1] B. Britton, D. Hill, W. Oswald, N-S Woo, and S. Singh. Optimized reconfigurable cell array architecture for high-performance field programmable gate arrays. *Proc. of IEEE Custom Integrated Circuits Conf. 1993*, pages 7.2.1–7.2.5, 1993.

[2] D. Hill, B. Britton, B. Oswald, N-S Woo, S. Singh, C-T Chen, and B. Krambeck. *ORCA : A New Architecture for High-Performance FPGAs*, pages 52–60. Springer-Verlag, 1993.

[3] AT&T Microelectronics. *Optimized Reconfigurable Cell Array (ORCA) Series Field-Programmable Gate Arrays*, 1993.

[4] H-C Hsieh, W. Carter, J. Ja, E. Cheung, S. Schreifels, C. Erickson, P. Freidin, and L. Tinkey. Third-generation architecture boosts speed and density of field-programmable gate arrays. In *Proc. of IEEE 1990 Custom Integrated Circuits Conf.*, pages 31.2.1–31.2.7, 1990.

[5] Xilinx. *XC4000 Data Book*, 1991.

[6] Altera. *FLEX 8000 Programmable Logic Device Family*, 1993.

[7] Xilinx. *Technical Data Book: XC3000 Logic Cell Array Family*, 1990.

[8] AT&T Microelectronics. *ATT3000 Series Field Programmable Gate Arrays*, 1991.