設 計 自 動 化 85－18
(1997. 10. 29)

ビットシリアル FPGA のフルカスタム設計

太田 章久　清水頭 武信　Imanuddin Amril　一色 剛　國枝 博昭

東京工業大学 工学部 電気・電子工学科

〒 152 東京都目黒区大岡山 2-12-1

Tel : 03-5734-2574

E-mail : aohta@ss.titech.ac.jp

あらまし　本稿ではビットシリアルパイプラインデータパスに適した新しい FPGA のアーキテクチャを提案し、それに基づき試作した LSI について述べる。我々は以前に FPGA を使った大規模書き替え可能システムについてのビットシリアルデータパスの自動合成をおこない既存の FPGA 上でのビットシリアルパイプラインデータパスの有用性を示した。今回の研究ではビットシリアルパイプラインデータパスに適した FPGA の設計をおこなった。0.5μm 2-metal の CMOS プロセスを用いて試作した結果、3.5mm 角 (IO パッドを除く) でトランジスタ数は 200k、クロック周波数は 156MHz であった。また更にこのアーキテクチャの問題を示すと共に改善策として新しいルックアップテーブルのアーキテクチャについて提案する。

キーワード　ビットシリアルデータパス, FPGA

Full Custom Design for Bit Serial FPGA

Akihisa Ohta, Takenobu Shimizugashira, Imanuddin Amril,
Tsuyoshi Isshiki and Hiroaki Kunieda

Department of Electrical and Electronic Engineering, Tokyo Institute of Technology

2-12-1, O-okayama, Meguro-ku, Tokyo, 152 Japan

Tel : 03-5734-2574

E-mail : aohta@ss.titech.ac.jp

Abstract　In this paper, we present our work on the design of a new FPGA architecture targeted for high-performance bit-serial pipeline datapath and its VLSI implementation. We have previously developed a bit-serial datapath synthesis system for large-scale configurable systems composed of a number of FPGA devices and successfully demonstrated the advantages of bit-serial pipeline datapath on conventional FPGA devices which are high device utilization and high speed. Here, we have developed our own FPGA architecture which is customized for high-performance bit-serial pipeline datapaths. The chip consists of 200k transistors on 3.5mm square substrate (excluding the IO pad area) using 0.5μ 2-metal process technology. The estimated clock frequency is 156MHz. Moreover we will describe disadvantages of our previous architecture and propose new lookup table architecture as improvement of performance.

key words　bit-serial datapath, FPGA

1 Introduction

Large-scale configurable systems composed of a number of FPGA devices have been demonstrated to have the capability of high-performance computing in various areas such as signal and image processing, data base search, pattern recognition, encryption, and embedded real-time systems. The drastic speed up of these systems is achieved by the highly parallelized hardwired computations whose architecture is strictly customized to a specific application, and also the large flexibility of the SRAM-based FPGAs for implementing a wide variety of applications.

FPGA technology itself has evolved significantly over the past few years, in which the logic capacity of the latest FPGA devices reaches well beyond 100k gates. As many studies suggests, the amount of required routing resource necessary for guaranteeing a high logic utilization increases as the logic capacity grows. This indicates that as the FPGAs continues to increase its logic capacity, the device utilization, the portion of area utilized to implement logic functions, decreases and the overhead area for implementing programmable interconnect networks between logic blocks begins to dominate the chip. For implementing configurable systems, where the main target is to implement high-speed datapaths, there are many opportunities in tuning the FPGA architecture to maximize device utilization and minimize routing area overhead.

In our previous work, we have developed a high-performance bit-serial pipeline datapath synthesis system for large-scale configurable systems [1]. Our bit-serial pipeline synthesis system can handle non-regular problems unlike systolic arrays, and covers a wide variety of applications such as FIR filter, IIR filter, adaptive filter, DCT, neural networks, etc. The design is captured at the algorithm-level by means of difference equations using C++, and the rest of the synthesis tasks are fully automated including the layout. Yet, we are able to empirically guarantee a near 100% logic utilization with 100% routability, consistently high speed clock operation ($\sim$ 40MHz on Xilinx 3100A FPGAs), without any low-level manual intervention. The most significant reason for our system's capability is the high routability of our bit-serial circuits. This is best described by referring to the Rent's Rule [2] which represents the degree of circuit connectivity given by the equation $P = k \cdot G^{\gamma}$ where P is the average pin count of the subcircuits, G is the gate count of the subcircuits, k is the Rent constant, and γ is the Rent exponent. The Rent exponents of various bit-serial circuits synthesized by our system are observed to be between 0.22 and 0.37 [3]. This is a strong contrast to the results of other studies on various benchmark circuits whose Rent exponent is observed between 0.47 and 0.75 [2], [5]. (A linear array has a Rent exponent of 0, 2D mesh has a Rent exponent of 1/2, 3D mesh has a Rent exponent of 2/3, 4D mesh has a Rent exponent of 3/4). It is known that if the Rent exponent is below 0.5, the expected average wiring length becomes independent of the circuit size, whereas if the Rent exponent is higher than 0.5, the average wiring length increases as the circuit size grow (this is given by the equation $r = G^{\gamma-0.5}$ where r is the average wiring length, G is the gate count, and γ is the Rent exponent) [4]. This is a very significant feature of bit-serial circuits which indicate that *we do not need to increase the routing resource while increasing the gate capacity*, unlike the situation which the current general-purpose FPGAs are facing.

Based on the above motivation, we have designed our own FPGA architecture, emphasizing on both logic architecture and routing architecture. Logic architecture is tuned to efficiently map the bit-serial operators in our library, such as multipliers, adders, rounders, etc. Routing architecture is designed to handle the local connectivity of the bit-serial circuits efficiently and reduce the routing delays.

2 Logic Block Architecture

One of the distinctive characteristics of bit-serial circuits is that the connectivity inside the cell is dense, while the connectivity of bit-serial cells. Our strategy here is to increase the logic capacity of the logic block and absorb the dense interconnection inside the logic block to reduce the inter-block routing resource. Fig.1 shows the logic block architecture. The following summarizes the features of the logic block architecture :

1. 4-input LUTs (lookup tables) $\times 4$ and 6 flip-flops.

2. The two multiplexers in front of the LUTs are targeted for carry-save operations which are frequently used in bit-serial computations.

3. There are 18 signal inputs and 6 signal outputs, plus a clock input.

4. Inputs c_2, c_3, c_4, c_5 can be connected to either GND or VDD. Therefore, each LUT can implement any 4-input function controlled by inputs a_0, a_1, a_2, a_3, or b_0, b_1, b_2, b_3.

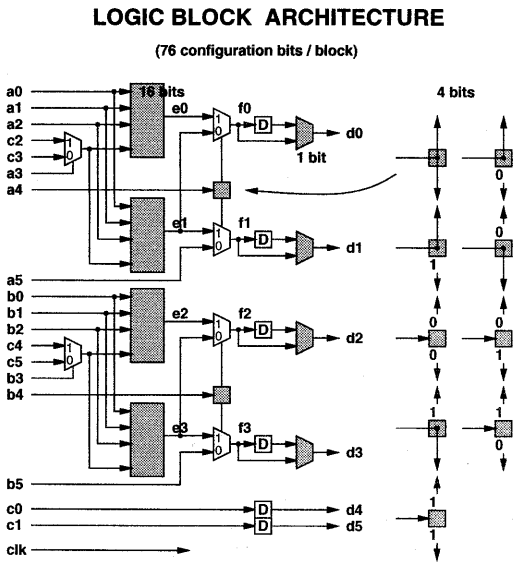

Figure 1: Logic block architecture of bit-serial FPGA.

5. Programmable switches connected to inputs a_4 and b_4 controls the functionality of the four multiplexers at the output of LUTs. As a result, 2 LUTs can implement any 5-input functions.

6. The final outputs d_0, d_1, d_2, d_3 can either be the direct outputs from the multiplexers or the outputs from flip-flops. All bit-serial operators use the outputs from flip-flops, therefore the attached programmable switches are actually unnecessary. They are only present in order to make our FPGA capable of implementing any logic functions other than bit-serial datapath circuits.

7. Two flip-flops are added (inputs c_0 and c_1) to implement shift registers which are frequently used in bit-serial operations.

3 Two-Level Routing Architecture

As we have described, the number of inputs and outputs on the logic block is rather large (24 pins), although the number of signals coming into or from the logic block in the actual bit-serial design is small (typically below 10 pins). Large portion of the output signals are only used inside the same logic block in bit-serial designs. Xilinx 4000 FPGA do not anticipate these feedback signals, and all feedback routing

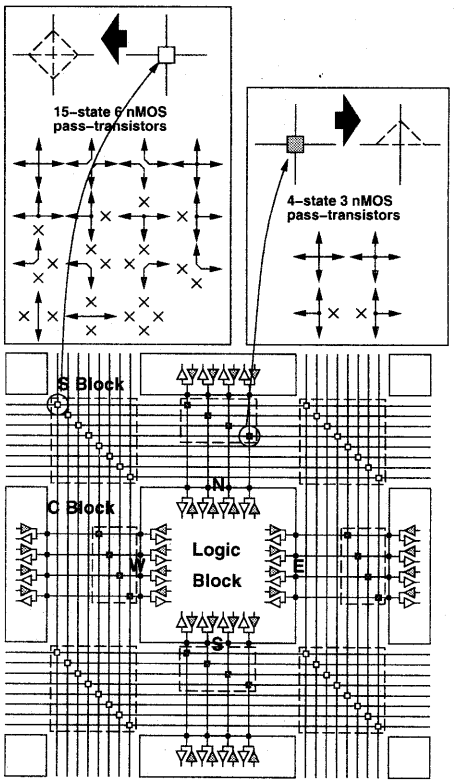

Figure 2: External block routing architecture of bit-serial FPGA.

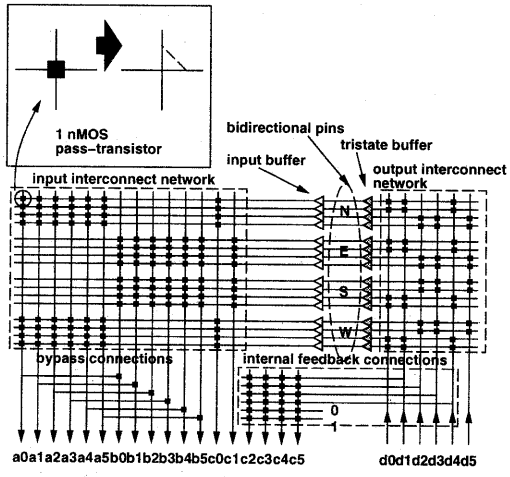

Figure 3: Internal block routing architecture of bit-serial FPGA.

is implemented on the inter-block routing resource. Since our aim in making the logic block large is to absorb the feedback routing inside the logic block, we need to provide a rich feedback routing resource inside the logic block. To achieve this, we develop a *two-level routing architecture* for our bit-serial FPGA. Fig.2 and Fig.3 illustrate our two-level routing architecture. The routing architecture is divided into two-levels : *external block routing* and *internal block routing*. The main features of this routing scheme are:

External block routing Inter-block routing is implemented with S-blocks and C-blocks. S-blocks are connected to 8-track single-length routing segments on four directions. S-blocks provide connectivity with the four routing segments using 8 programmable switches. C-blocks provide connectivity between routing segments and logic blocks using 4 programmable switches. Each logic block contains four bidirectional pins on all four directions (16 pins total), which are each connected to a C-block.

Internal block routing Signal routed to one of the logic block pins is buffered at the input or output, and then connected to yet another routing network to provide routing flexibility. North and west pins have connectivity to inputs $a_0, a_1, a_2, a_3, a_4, a_5, c_0$ and all outputs. East and south pins have connectivity to inputs $b_0, b_1, b_2, b_3, b_4, b_5, c_1$ and all outputs. When any of the input pairs (a_0, b_0), (a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4), (a_5, b_5) share the same signal, they can be accessed from all four directions using bypass connections. Dedicated feedback connections for the inputs c_2, c_3, c_4, c_5 from the outputs d_0, d_1, d_2, d_3 are also provided.

Advantages of our two-level routing architectures can be summarized as follows :

1. The large number of input and output signals of the logic block would create a significant capacitive load due to the drain capacitance of the pass-transistors on the routing segments. This routing scheme is seen in Xilinx 4000 FPGAs (Fig.4(a)). In their scheme, the input and output pins are connected directly onto the routing segments via pass-transistors. By our routing scheme, the intermediate routing resource inside the logic block (hence the *two-level routing*) enabled us to insert buffers at the logic block pins which effectively isolates the capacitive load of the drain capacitance of pass-transistors from the routing segments. The routing delay through the single-length routing segment is greatly reduced. This fact also leads to power reduction.

2. All logic block outputs can be routed back to itself without consuming any external routing resource. Also, connections between adjacent logic blocks which frequently occurs in bit-serial circuits is implemented via C-blocks without consuming any external routing resource as well. These two features are also not seen in Xilinx 4000 FPGA.

3. The connectivity of inputs and outputs are made symmetric with a high degree of flexibility in four directions. This increases the routability and also makes the automatic routing very easy to develop.

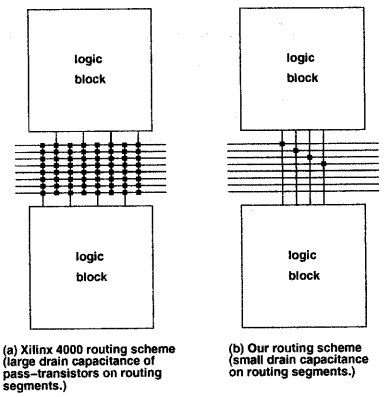

Figure 4: Comparison of C-block structure between Xilinx 4000 FPGA and our bit-serial FPGA.

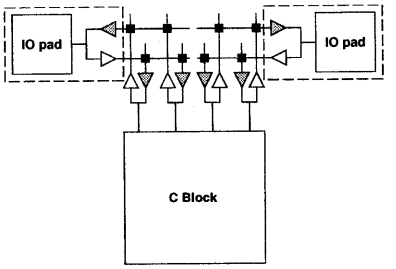

Figure 5: IO block architecture for bit-serial FPGA.

4 IO Block

IO blocks are connected to C-Blocks located on the chip edges. IO blocks are designed with minimum

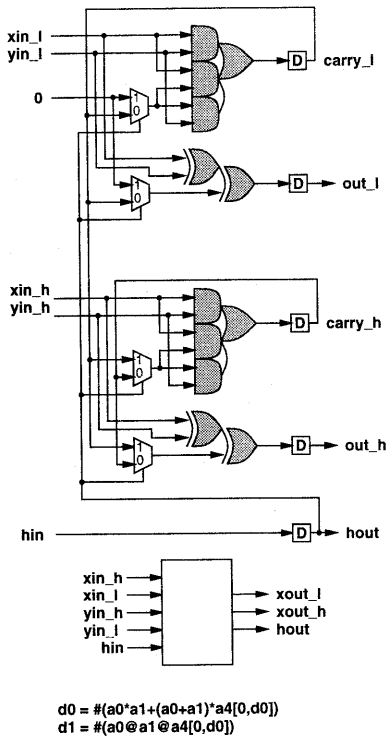

Figure 6: Circuit implementation of bit-serial adder cell.

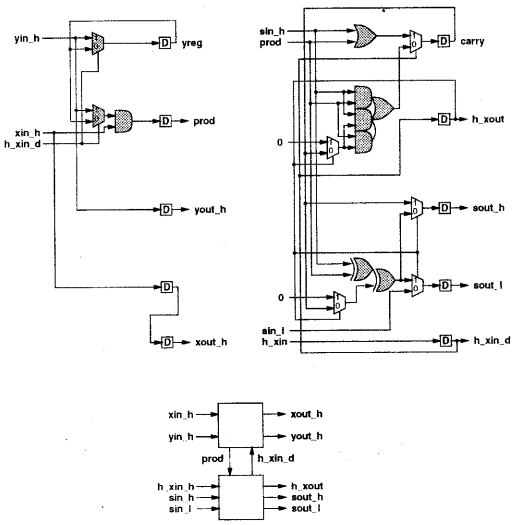

Figure 7: Circuit implementation of bit-serial multiplier cell.

functionality, merely to provide connectivity between logic blocks and IO pads (Fig.5). Since we have provided sufficient number of flip-flops inside the logic block, we assume that the signals are latched once routed inside the logic block.

5 VLSI Implementation

The bit-serial FPGA architecture was first transformed into a transistor-level description on Verilog (1 months × 2 persons), and mask layout was done on full-custom (4 months × 2 persons). The process technology used in this design was 0.5μ (gate length = 0.6μ) 2-metal process. Due to limited manpower and time, we could not spend enough time on the floorplanning and transistor size optimization. Therefore the layout result leaves room for further improvement on area and speed. Component size for our design chip is summarized in Table.2. One logic block, one S-block and two C-block consumed $385\mu \times 407\mu$ area.

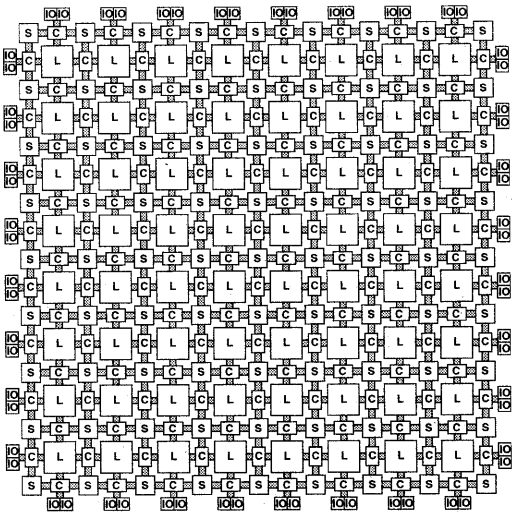

Figure 8: Bit-serial FPGA chip floorplan.

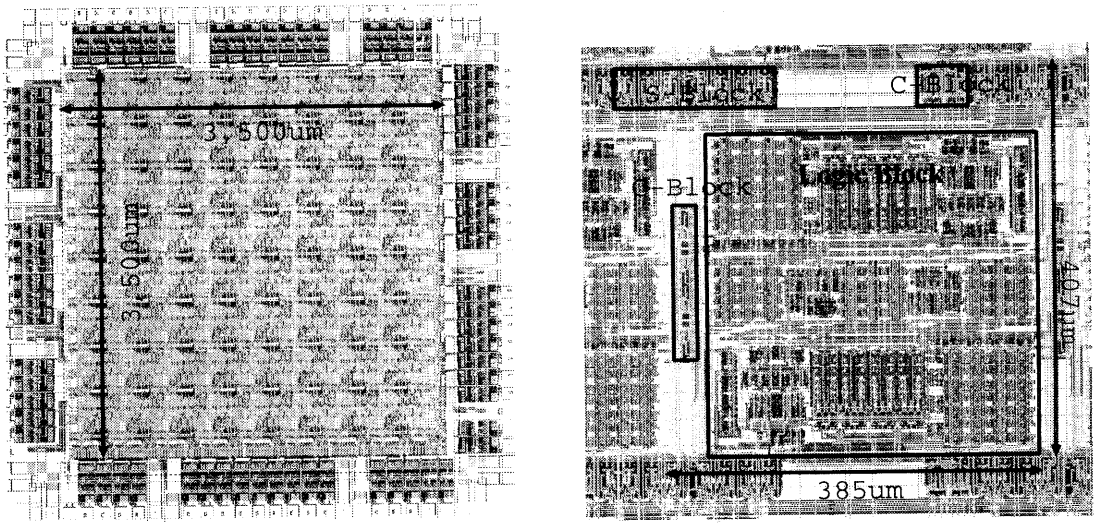

Figure 10: Layout of bit-serial FPGA chip (left), and layout of logic block (right).

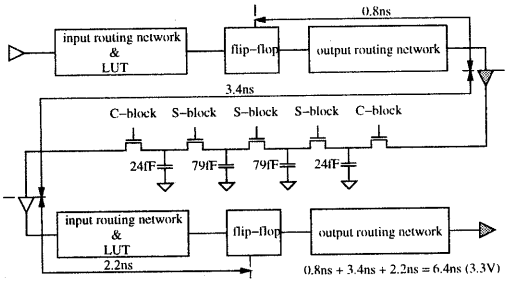

Figure 9: Delay model of bit-serial FPGA (assumes 4 manhattan distance routing).

With further rework on the layout, we would expect the area to shrink within $350\mu \times 350\mu$. If we were to use a 3-metal process, we expect the area would further reduce by half. Inside the $3.5 \times 3.5mm$ chip area (excluding IO area), there are 8×8 logic blocks and 64 IO blocks (Fig.8). Various data are shown in Table 1. In one chip, we can fit either two 16-bit multipliers, four 8-bit multipliers, or 64 double-precision adders (independent of word size).

Fig.9 shows the delay model for our bit-serial FPGA. Total delay from flip-flop output to flip-flop input with 4 manhattan distance routing is 6.4 ns.

Table 1: Estimated performance of our bit-serial FPGA chip

area (LB, SB, CB×2)	$385 \times 407\mu^2$
area (total)	$3,500 \times 3,500\mu^2$
transistor count	200k transistors
max. gate/block	~70 gates
max. gate/chip	~4500 gates
clock frequency	156 MHz
(assume 4 manhattan distance routing)	
16-bit multiplication (×2)	19.5 MOPS
8-bit multiplication (×4)	78 MOPS
16-bit addition (×64)	624.64 MOPS
8-bit addition (×64)	1.25 GOPS

Table 2: Component size for bit-serial FPGA

Unit	Size	Composition
Die	$4.8mm \times 4.8mm$	Core with pads
Core	$3.5mm \times 3.5mm$	All internal logic except pads
L–Block	$332\mu m \times 320\mu m$	Logic Block
S–Block	$138\mu m \times 37\mu m$	Switching Block
C–Block(Type I)	$43\mu m \times 37\mu m$	Connection Block
C–Block(Type II)	$8.5\mu m \times 150\mu m$	Connection Block
L + S + C ×2	$385\mu m \times 407\mu m$	
Configuration block	$350\mu m \times 80\mu m$	Shift register and some control logic

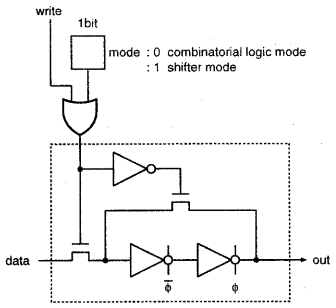

Figure 11: New LUT RAM cell

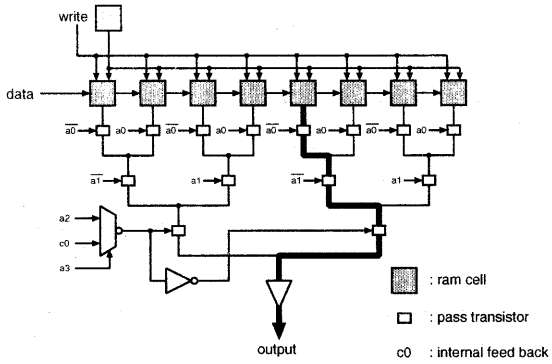

Figure 12: Bit–serial 3–input LUT

6 New lookup table architecture

Next, as our extension, we propose new lookup table architecture for improving our current bit–serial FPGA. Our design bit–serial FPGA has some problem as follows.

1. For configuration of RAM cells throughout the FPGA, a large number of *data* lines and *write* lines are needed, which caused difficulties in routing these signals.

2. Configuration block need many shift registers.

3. Bit–serial operation must have many shift registers for pipleline synchronization. Although we provided two extra flip-flops for each L-block, this is still not enough for many of the bit-serial applications.

As a solution to this problem, we present a new lookup table architecture which can be configured as shift register or conventional logic.

Fig.11 shows the new LUT RAM cell. Data is stored in clocked inverter loop when *write* signal is high. When this LUT is used as a shift register, RAM cells are connected in series to form a dynamic D flip-flop. Control signal *write* is generated from configuration block and it is shared with same row L–blocks. One LUT has 1–bit memory which indicate LUT mode (combinatorial logic or shifter). With this new RAM cell, either a chain of shifters or a lookup table can be implemented. Both of shifter and configuration bit inputs are transmitted in bit–serial.

Fig.12 shows a 3–input LUT using new RAM cell. New LUT is similar to our previous LUT except RAM cell. a_0, a_1, a_2 and a_3 is input signal from input interconnect network and c_0 is internal feedback signal. 3–input LUT requires 8 bit memory. 8 to 1 decoder has a tree structure. The node of the tree is composed of one pass transistor, so decoder is small and fast. a_0, a_1 and a_2 is LUT's input and decide N (shift number). When $a_0 = a_1 = 0$ and $a_2 = 1$, this is 5

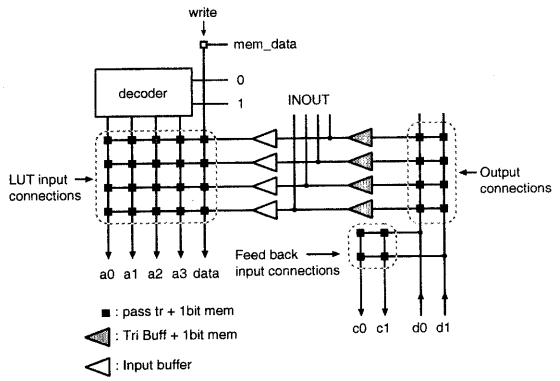

Figure 13: New internal routing architecture

bit shifter. Signal *data* is LUT's RAM cell input and shifter input. When LUT is in write mode (configuration bit store), signal *write* is high and *data* is written to LUT's RAM cell per clock.

To realize shifter mode, a_0, a_1, a_2 and a_3 are fixed to '0' or '1' while LUT is this mode. So we improve internal routing architecture as illustrated Fig.13. Alteration point is adding decoder to keep '0' or '1' state. Consequently shifter mode is realized and '0', '1' input can share initialize input which used carry set or reset. Furthermore since internal feed back connections is simple, it is expected that L–block is small.

This improved LUT is more attractive for bit–serial FPGA in terms of area size and routing problem. But power consumption is large because LUT's RAM cell is driven by clock. We examine this problem in points that clock is not supplied when LUT is not shifter mode. Another problem is how many number of LUT's input. To minimize silicon area, the number configuration bits may be small. It is prefer that the number of LUT's inputs is smaller. In bit–serial operation it is satisfactory that the number of LUT's input is three. However, some application like IDCT which requires many shift registers cause insufficiency of shift register resources and complex of placement and routing problems. To solve this trade–off is future work.

7 Summary

In this paper, we presented our new FPGA architecture and VLSI implementation for high-performance bit-serial pipeline datapaths. The main features of the new architecture are the large logic block consisting of four 4-input LUTs and 6 FFs with dedicated carry-save logic for bit-serial operations, and also the two-level routing architecture aimed to make the interblock routing simple with short delays while providing another layer of routing inside the logic block to provide pin flexibility in order to increase routability. We are currently developing a placement and routing tool while evaluating the current architecture.

As for VLSI implementation, circuit area is $3.5 \times 3.5mm$ using 0.5μ 2-metal process. The chip consists of 64 logic blocks and 64 IO blocks.

Moreover we will describe disadvantages of our previous architecture and propose new lookup table architecture as improvement of performance.

ACKNOWLEDGMENT

Authors would like to thank the members of CAD21 Research Body of Tokyo Institute of Technology, VDEC which provides us an opportunity to implement VLSI, and members of Kunieda Laboratory for their suggestion and cooperations.

References

[1] Tsuyoshi Isshiki and Wayne Wei-Ming Dai, "Bit-Serial Pipeline Synthesis for Multi-FPGA Systems with C++ Design Capture," *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, April 1996. T. Isshiki and W. W. -M. Dai,

[2] B. Landman and R. Russo, "On a Pin Versus Block Relationship for Partitioning Logic Graphs," *IEEE Trans. Computers*, pp.1469-1479, 1971.

[3] T. Isshiki, W. W. -M. Dai, Hiroaki Kunieda "Routability Analysis of Bit-Serial Pipeline Datapaths," *To appear on IEICE Trans. Fundamentals*, Oct. 1997.

[4] Wilm E. Donath, "Placement and Average Interconnection Lengths of Computer Logic," *IEEE Trans. Circuits and Systems*, pp.272-277, April 1979.

[5] L. Hagen, A. B. Kahng, F. J. Kurdahi, C. Ramachandran, "On the Intrinsic Rent Parameter and Spectra-Based Partitioning Methodologies," *IEEE Trans. Computer-Aided Design*, pp.27-37, Jan. 1994.