システムＬＳＩ設計技術　94－10
（2000. 1. 11）

〔招待論文〕ビットシリアルFPGAシステムの開発

一色剛　太田章久　國枝博昭

東京工業大学 電気・電子工学科

〒 152-8552 目黒区大岡山 2-12-1

E-mail: {isshiki,aohta,kunieda}@ss.titech.ac.jp

あらまし　FPGA の設計において，設計自動化ツールは必須であるが，これらのツールが生成する回路は多くの場合，配線性に乏しいため論理使用効率の低下や回路速度の低下を招いているのが現状である．我々は，この設計自動化と論理使用効率・パフォーマンスの向上の両立という目的のもと，ビットシリアル回路に基づいた FPGA 上での設計について研究を進めている．本論文では，我々が現在まで行って来たビットシリアル回路設計，アプリケーション設計環境の構築，新たなビットシリアル用 FPGA アーキテクチャの開発，等についてまとめ，さらに今現在進行している研究について述べる．

キーワード　FPGA，ビットシリアル，高パフォーマンス，論理使用効率，高配線性

Development of Bit-Serial FPGA System

Tsuyoshi Isshiki　Akihisa Ohta　Hiroaki Kunieda

Dept. of Electrical and Electronic Engineering

2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan

E-mail: {isshiki,aohta,kunieda}@ss.titech.ac.jp

Abstract　Design automation is essential in FPGA designs. However, the circuits generated by these synthesis tools tends to have poor circuit routability, resulting in low logic utilization and/or low circuit speed. We have been investigating the use of bit-serial circuits in FPGA environment to address this difficult issues of incorporating design automation while guaranteeing a highly efficient routability of the circuit, therefore increasing the silicon utilization and improving performance. Our research stems widely from bit-serial circuit designs, development of application design environment, development of new FPGA architectures, as well as new FPGA applications. In this paper, we summarize our work on these issues as well as some of the current on-going researches.

Keywords　FPGA, Bit-serial, High performance, Logic utilization, High routability

1 Introduction

Silicon process technology is constantly keeping up the pace of improvement in the last few decades, which will soon reach 0.1μ. As the circuit size inside the chip grows, the device area devoted only for wiring signals becomes significantly large. Especially in the case of FPGAs, this increase in the routing resource results in the decrease in the silicon utilization of the actual user-logic resource, where the logic density is already a factor of magnitude lower than the full custom chips. The focus of the current state-of-the-art FPGA architectures is directed on increasing the routing capacity as well as the logic capacity. This is a great dilemma for the FPGA architects, where insufficient routing resource results in low logic utilization, but increasing the routing resource also results in low silicon utilization dedicated to logic implementation.

Also, by the many works on the high-level synthesis community and layout CAD community, fully automated design systems from the behavioral synthesis to automatic placement and routing already exist for FPGA environment. However, its effectiveness is still questionable for its incapability to handle large problems and the quality of the produced results.

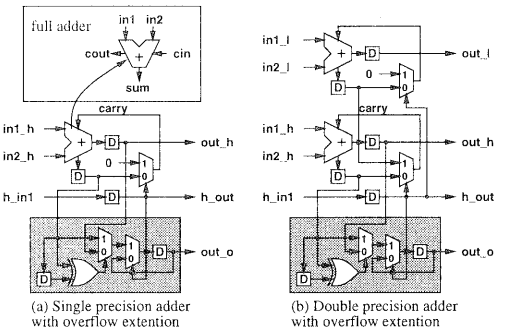
Our research group has been investigating the use of bit-serial circuits in FPGA environment to address these difficult issues of incorporating design automation while guaranteeing a highly efficient routability of the circuit, therefore increasing the silicon utilization and improving performance. Our research stems widely from bit-serial circuit designs, development of application design environment, development of new FPGA architectures, as well as new FPGA applications. In this paper, we summarize our work on these issues as well as some of the current on-going researches.

2 Bit-Serial Pipeline Synthesis System

We have developed an application design environment which targets FPGA-based configurable systems where the hardware is configured for a specific application in order to accelerate the application normally running on software. The main goals of this design environment are :

- To simplify the design task so as to be acceptable by software designers oriented towards theory and application. This includes eliminating the need for HDL coding, enabling simultaneous hardware/software design verification, and fully automating the circuit synthesis and layout tasks.

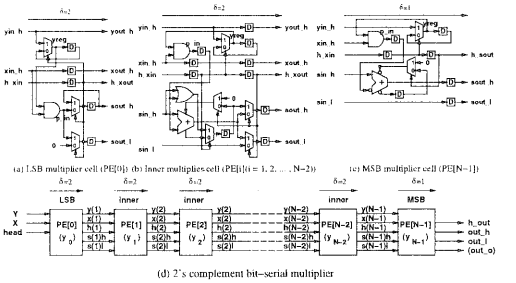
- To guarantee accurate prediction of performance and hardware resource utilization before the lengthy layout synthesis. Totally eliminate the need for low-level manual intervention which requires not only expert digital system design skills but also the deep understanding about the target FPGA architecture.



(a) Single precision adder with overflow extention

(b) Double precision adder with overflow extention

Figure 1: Bit-serial adders. Hashed regions are the overflow extension circuits.



(d) 2's complement bit-serial multiplier

Figure 2: Bit-serial multiplier. It has a linear array structure.

2.1 Structure of Bit-Serial Circuits

In our bit-serial synthesis system, we provide a numbers of bit-serial operators essential in designing various applications such as adders, multipliers, scalers, rounders, saturaters, parallel-to-serial and serial-to-parallel converters, etc., which are efficiently implemented on LUT-based FPGA architectures. Fig.1 shows the bit-serial adder, and Fig.2 shows the bit-serial multiplier. For internal computation, we provide double precision numbers using two data lines mainly for output data of multipliers which produces double precision products. Also, in a situation where a large number of accumulations occur and overflow is unavoidable, we provide additional data line for overflow prevention. For N-bit single precision data, *extended* single precision data has $2N$ bits, and *extended* double precision data has $3N$ bits. Various types of rounders and saturaters are provided to convert these data types.

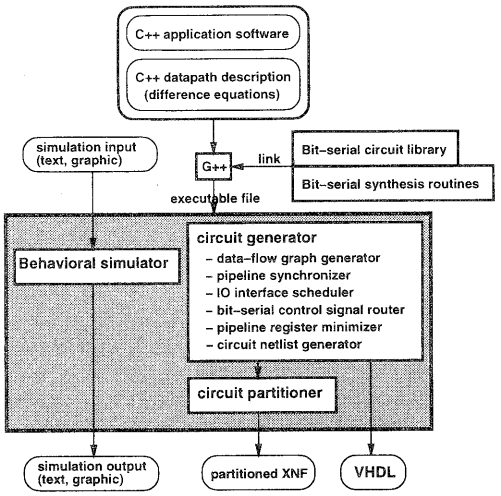


Figure 3: Design flow of our bit-serial synthesis system. The designer describes the algorithm in terms of difference equations on C++ using our dedicated class objects. The C++ program is compiled by a standard GNU C++ compiler (g++) which produces an executable file which performs behavioral simulation, circuit generation and circuit partitioning, and produced circuit-level netlist (partitioned XNF files, unpartitioned VHDL code).

2.2 C++ Design Capture

The design flow is illustrated in Fig. 3. First, the designer will enter the design of both the hardware and software using C++ [3]. Software design description is allowed to use the full capability of C++, whereas the hardware description is done by *difference equation* formulation on C++ using a set of *hardware class variables* (Fig.4). This C++ file is compiled by a general C++ compiler where the produced executable file becomes both the behavioral simulator for the total application (software and hardware) and also the circuit generator for the hardware design. The software description can also include breaking points and text or graphical outputs for debugging purpose. Any C++ debugger tool can also be used for this purpose as well.

2.3 Bit-Serial Pipeline Synthesis

Our bit-serial pipeline synthesis methodology consists of capturing the difference equations and converting them into data-flow graph, converting the data-flow graph into primary circuit netlist by direct mapping of the arithmetic operators into bit-serial hardware modules, pipeline scheduling/optimization, and circuit partitioning.

```
1D_FIR(int &int, int &out, Interface &bus,
       Single x[8], Single &y, Double a[8],
       double coef[8])
{
  set_signal_precision(16);
  // setting the signal word length to 16 bits

  for(int j=0; j < 8; j++){
    if(j==0){
      x[j] = bus.read(in);  // x[] is the input data
      a[j] = x[j] * coef[j];
    }
    else{
      x[j] = delay(x[j-1]);
      // x[j] is the delayed by one sampling period
      // from x[j-1]
      a[j] = x[j] * coef[j] + a[j-1];
    }
  }
  y = round(a[7]);   // convert from double to single
  bus.wire(out) = y; // y is the output data
}
```

Figure 4: 1D FIR filter C++ description

Circuit Generation After the design is captured by C++ by means of difference equations, bit-serial circuit modules are created for each distinct arithmetic operation by calling the bit-serial circuit library and the primary circuit netlist is constructed.

Minimum Sampling Period Calculation The minimum sampling period is calculated in the case where there are loops in the pipeline datapath.

Pipeline Scheduling Optimization After the minimum sampling period is determined, the pipeline network is synchronized by inserting retiming registers in order to equalize the latency on every path in the network. At the same time, the number of retiming registers are minimized in order to reduce the hardware overhead.

2.4 Routability of Bit-Serial Circuits

In analyzing the circuit structure to evaluate the area required for VLSI layout, Rent's rule is commonly used [2] [6] [1]. This rule defines the relationship between the average number of pins and the average number of logic circuits in a subcircuit. which is expressed as $P = k \cdot G^{\gamma}$, where P is the average number of external pins in a subcircuit, and G is the average number of modules in a subcircuit. k is the *Rent constant* which has empirically been found to correspond to the average number of pins per module. γ is the *Rent exponent* which ranges between 0 and 1. It is well known that for circuits whose Rent exponent γ is over 0.5 (this is found to be the case for most real-life circuits [2] [1]), the average wiring length grows exponentially with the circuit size G [6]. This im-

plies that the routing resource (routing channel width) needs to be increased as the chip size grows. This rule is in fact forming the basis of current FPGA architectures, where larger chips are provided with increased number of routing channels.

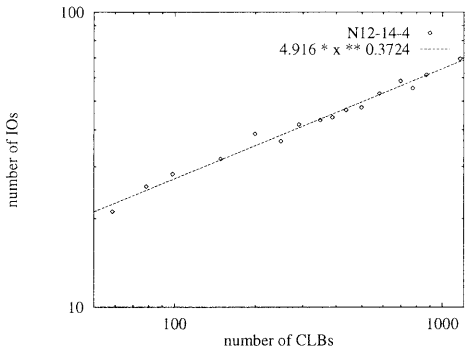


Figure 5: Rent's exponent approximation for N12-14-4. The two axes are log-scale.

We have obtained the Rent's parameters for our various bit-serial designs by applying multi-way partitioning with different size constraints. The method for deriving Rent's parameters is to plot the average CLB counts and the average IO counts per partition on the log-log scale and use linear regression to estimate the slope and the intercept [4]. Here, CLB is the logic block of Xilinx 3100A FPGA. Fig.5 shows the plots for bit-serial neural network circuit. The Rent's parameters for other designs are summarized in Table 1. Compared to Landman and Russo's work where they reported the Rent's exponent to be between 0.47 and 0.75, our bit-serial circuits have a significantly lower Rent's exponent between 0.22 and 0.37. This implies that our bit-serial circuits are very highly routable circuits on two-dimensional plane. This was caused mainly by the dominant bit-serial multiplier circuits. These multipliers whose Rent exponent is 0 (a linear array) absorbed the circuit complexity in terms of Rent exponent at the high fanout nodes. There have not been any studies on logic circuits which reveal such a low Rent's exponent for *real* applications. Our bit-serial circuit design and our bit-serial pipeline network synthesis strategy led to such an extremely routing-efficient circuit structure. Since FPGAs are tuned to implement "hard-to-route" real circuits having high Rent exponent of over 0.5, we can expect that lack of routing resource on FPGAs for our bit-serial circuits is unlikely to cause any problem.

Table 1: Rent's parameters for bit-serial circuits. "N-12-14-4" is a digital neural network with 12, 14, 4 nodes on each layer. "N-8-8-4" is a digital neural network with 8, 8, 4 nodes on each layer. "1D-FIR-30-II" is a 30-tap 1D FIR filter. "2D-FIR8x8" is an 8×8-tap 2D FIR filter. "1D-IIR20-III" is a 20-tap recursive filter with 2 sampling period lookahead on the feedback loop. "adapt10T-7" is a 10-tap 1D adaptive FIR filter where the coefficient updating cycle delay is 7 sampling periods. "IDCT-I" is an Inverse-DCT circuit.

design	γ	k
N-12-14-4	0.3724	4.916
N-8-8-4	0.3264	5.878
1D-FIR30-II	0.2218	6.018
2D-FIR8x8	0.3245	4.665
1D-IIR20-III	0.3264	5.878
adapt10T-7	0.3735	5.150
IDCT-I	0.3394	5.493

3 A New Bit-Serial FPGA Architecture

Based on these studies, we have designed our own FPGA architecture which specifically targets bit-serial circuits. Our bit-serial FPGA contains logic blocks, routing blocks and I/O blocks as same as popular FPGAs. Our FPGA chip floorplan is shown in Fig.6. Logic block (L) implementing combinational and sequential logic is placed in a regular array on the chip. Routing blocks (S and C) provide interconnect of each logic block. I/O block (IO) surround the core.

3.1 Logic Block Core Architecture

The logic block core is a primitive block in bit-serial FPGA. One of the distinctive characteristics of bit-serial circuits is that the connectivity inside the cell is dense, while the connectivity between bit-serial cells is sparse. 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.7 shows the logic block core. The design was based on Xilinx 3000 series FPGAs, since our bit-serial datapath synthesis system was originally applied to this architectures in which we had much success. The modifications we employed to this architecture are:

1. The number of 4-input LUTs is increase to 4, while a pair of LUTs share the same inputs. By doubling the number of LUTs, more nets connecting the circuit elements inside the bit-serial cell are buried

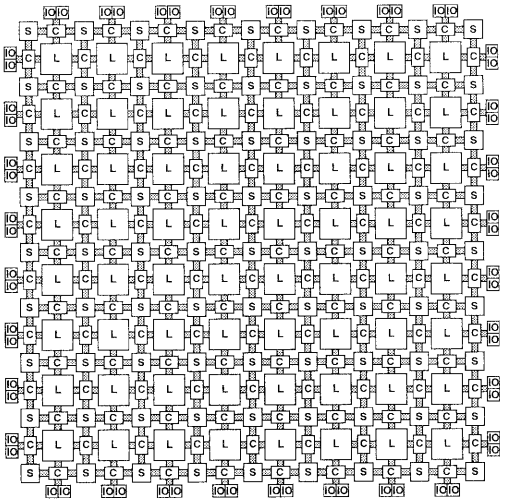


Figure 6: Bit-serial FPGA chip floorplan.

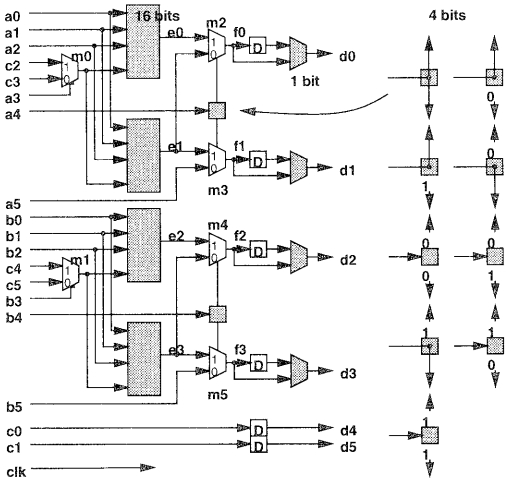


Figure 7: Logic block core architecture of bit-serial FPGA.

inside the logic block, which results in savings of inter-block routing resource.

2. There are 6 multiplexers at the inputs and outputs of each LUT which are inserted specifically to efficiently implement bit-serial cell. The 2 multiplexers at the inputs $(m0, m1)$ are for initializing the carry-bit when the bit-serial data cross the word boundary. $c2, c3, c4, c5$ are the feedback signals from the FF outputs $d0 \sim d6$. These four signals can also be connected to 0 or 1.

3. Two additional FFs are added which simply latches $c1$ and $c2$. They are inserted to provide shift register elements within the bit-serial pipeline datapath.

3.2 Two-Level Routing Architecture

The number of inputs and outputs on the logic block is rather large (24 pins), although the number of signals coming into or outgoing from the logic block in the actual bit-serial design is small (typically below 10 pins). 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 developed a *two-level routing architecture* for our bit-serial FPGA. Fig.8 and Fig.9 illustrate our two-level routing architecture. The routing is divided into two-levels : *external block routing* and *internal block routing.*

3.2.1 Internal Block Routing

Signal routed to one of the logic block pins is buffered at the boundary of the block, and then connected to yet another routing network. 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. On the other hand, 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.

3.2.2 External Block Routing

External-block routing is implemented with S-blocks and C-blocks. S-blocks are connected to 8-track single-length routing segments for four directions. S-blocks provide connectivity with the four routing segments using 8 programmable switches. This programmable switch realizes 15 types of connections as shown in Fig.9. This routing architecture using 8 signal track segment and S-block is based on the Xilinx XC4000 architecture for its simplicity in constructing the routing program.

C-blocks provide connectivity between routing segments and logic blocks using 4 programmable switches. This programmable switch realizes 4 types of connection as shown in Fig.9. Each logic block contains four bidirectional pins on all four directions (16 pins total), each of which is connected to a C-block. This simplification of C-block is one of a distinct feature of our two-level routing architecture.

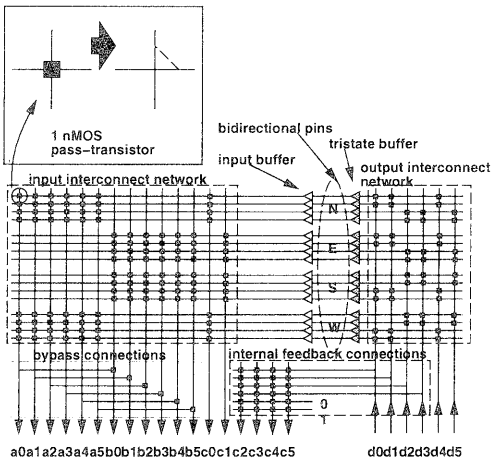


Figure 8: Internal block routing architecture of bit-serial FPGA.

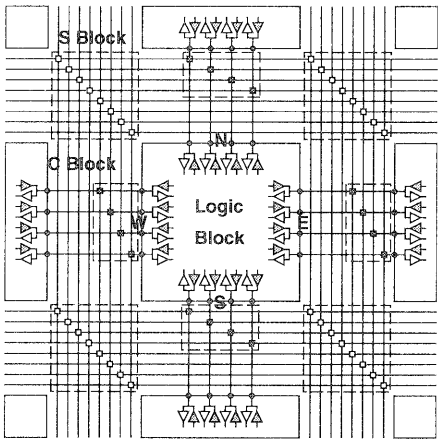


Figure 9: External block routing architecture of bit-serial FPGA.

Table 2: Component size for bit-serial FPGA.

Unit	Size
Transistor count	200k
Die	$4.8mm \times 4.8mm$
Core	$3.5mm \times 3.5mm$
L-Block	$332\mu m \times 320\mu m$
S-Block	$138\mu m \times 37\mu m$
C-Block(Type I)	$43\mu m \times 37\mu m$
C-Block(Type II)	$8.5\mu m \times 150\mu m$
L + S + C ×2	$385\mu m \times 407\mu m$
Configuration block	$350\mu m \times 80\mu m$

Table 3: Estimated performance of our bit-serial FPGA chip.

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

3.3 VLSI Implementation

The bit-serial FPGA architecture was first transformed into a transistor-level description on Verilog (1 months × 2 persons), and full-custom mask layout was performed (4 months × 2 persons). The process technology used in this design was $0.5\mu m$ (gate length = $0.6\mu m$) 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 $385\mu m \times 407\mu m$ area. With further rework on the layout, and by using 3-metal process which is common for current technology, we should be able to minimizing the empty space surrounding the S-block and C-block cells and also the wiring areas of the 8-track routing segments. If this can be achieved, the area of $385\mu m \times 407\mu m$ is expected to shrink to within $350\mu m \times 350\mu m$ (the raw area sum of L-block, S-block and 2 C-blocks are less than $(335\mu m)^2$). Inside the $3.5 \times 3.5mm$ chip area (excluding I/O area), there are 8×8 logic blocks and 64 I/O blocks (Fig.6). Various data are shown in Table 3. 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.11 shows the result of layout.

Total delay from flip-flop output to flip-flop input with 4 manhattan distance routing is 6.4 ns, which results in 156MHz clock operation. Since the longest wiring length (critical path) is shorter than parallel structure, we obtain higher speed. This is one of distinctive features in bit-serial circuits. Another factor of high speed is to divide routing resource into two parts. Since we put buffer between internal and external routing resource, we can also reduce critical path delay.

Table 4: Mapping results I : 2D FIR filter (5×5-tap, 8-bit), wave filter (5th order, 16-bit), adaptive FIR filter (10-tap, 16-bit) and IDCT (8-point, 16-bit)

Application	Chip size	Gate Count	LUT count	Logic util.	Ave. routing dist.	Max routing dist.
2D FIR filter	14×14	6791	518	96.4%	1.77	5
IDCT	19×19	13369	898	94.2%	1.92	6
Wave filter	21×21	15846	530	95.0%	1.93	7
Adaptive filter	35×35	48510	2397	96.5%	1.86	9

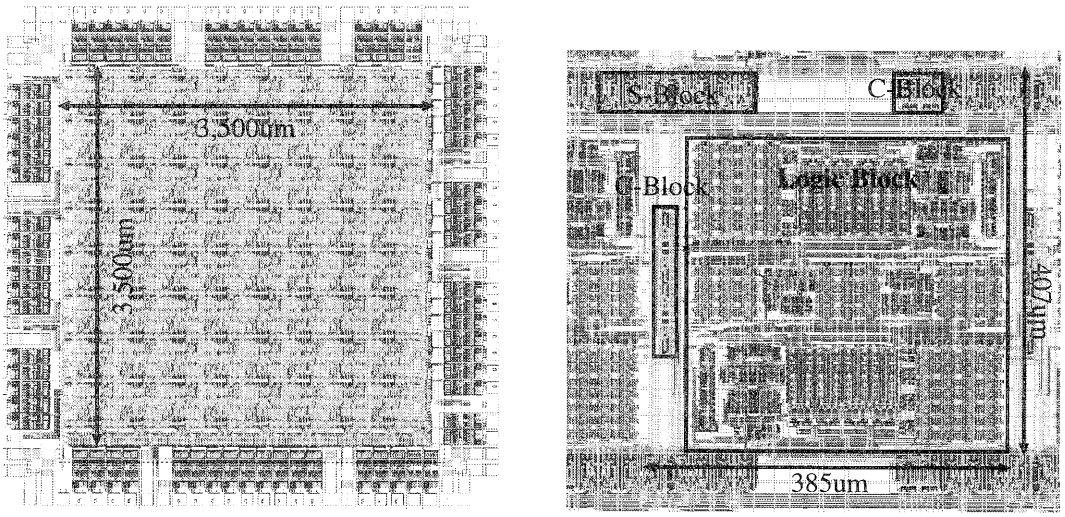


Figure 11: Layout of bit-serial FPGA chip (left), and layout of logic block (right).

4 Design Examples

We have developed a placement tool based on simulated anealing and a routing tool based on maze router for our bit-serial FPGA. Based on these dedicated FPGA layout tools together with our bit-serial datapath synthesis system described earlier, we have done experiments on mapping several applications. Fig.10 shows a mapping result of IDCT on a chip with 19×19 logic blocks. Here, the chip size is set to $n \times n$ such that the total number of logic blocks is larger than the required number of logic blocks for each circuit.

Table 4 shows mapping results generated from our placement and router tool. One of the distinctive feature is that all applications have very high logic utilization near 100% and very small average distance. These applications can be implemented as dense as possible and the routing between logic blocks are very localized. These results show that our bit-serial FPGA architecture can scale efficiently while guaranteeing near 100% utilization without adding extra routing resource.

Chip scalability in terms of performance is largely affected by the quality of place-route tools. While the average routing distance is low regardless of the chip size, maximum routing distance, unfortunately, increases with the chip size. This is *solely due to the placement algorithm we employed with was routability-driven.* Our routing tool was able to guarantee minimum distance routing of the longest nets. The primary goal for routability-driven placement is to minimize the average routing distance, and therefore minimizing the distance of longest nets is given a lower priority. Since our bit-serial circuits are extremely routing efficient, we are currently seeking ways to develop a performance-driven placement which can trade off speed (longest net distance) and routability (average net distance) so as to guarantee high clock speed operation as well as high logic utilization regardless of the chip size.

5 Current Research Topics

We are now expanding our research on a next generation bit-serial FPGA architecture and new applications.

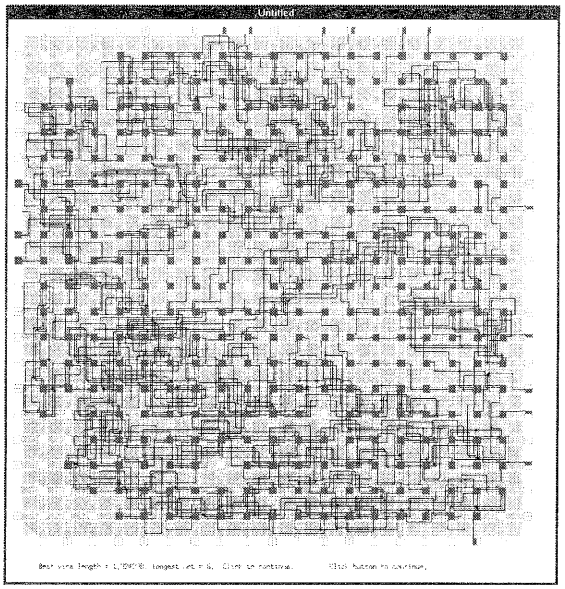


Figure 10: Mapping result of IDCT

- We have completed a new logic block architecture design which efficiently implements shift registers using LUT [5]. Shift registers compose a large portion of the bit-serial datapath, more than 60% of the gate count in some cases. With a modification on the LUT circuitry, we are now able to implement 16-bit shift registers in each 4-input LUT, which results in a significant savings of logic block resource. Synthesis tool which handles this new logic block architecture is also completed.

- We have set the main target for our next generation bit-serial FPGA on image/video processing and 3D graphics. We are now exploring ways to incorporate new packaging technologies to provide a high data bandwidth between large external memory. One key technology is to utilize "area-IO" where the IO blocks are located inside the chip core, drastically increasing the number of IOs. Using this area-IO, the FPGA dice (unpackaged) is mounted on a large DRAM dice (also unpackaged) by the Chip-On-Chip assembly technology, resulting in a single "substrate-less" Multi-Chip Module. This technology, unlike System-On-A-Chip, enables to utilize the latest process technologies individually tuned for DRAM and logic, and achieves a significant increase in data bandwith which is needed for video and 3D graphics applications. Design of area-IO architecture, new DRAM architecture, and new bit-serial circuit libraries for these applications are now in progress.

6 Conclusion

In this paper, we have introduced our work on bit-serial pipeline synthesis system which can empirically guarantee high performance and high logic utilization. The significant factor of this system's capability is the high routability of our bit-serial circuits. We have used Rent's rule to compare the expected wiring length of the real-life circuits observed in the past and our bit-serial circuits. While the average wiring length of most of those real-life circuits grows with the circuit size, the average wiring length of our bit-serial circuits are expected to remain constant with different circuit size.

Also, based on this observation, we have developed our own bit-serial FPGA architecture and implemented on VLSI. Our design examples based on our bit-serial datapath synthesis system and our taylor-made placement and routing tools show that our bit-serial FPGA can guarantee high utilization and high performance for a number of applications.

We are currently in the phase of a next generation bit-serial FPGA architecture utilizing a new system packaging technology which targets image/video processing and 3D graphics applicatinos.

Acknowledgement

Authors would like to thank the members of CAD21 Research Body of Tokyo Institute of Technology and members of FPMCM project of University of California at Santa Cruz for their suggestion and cooperations.

References

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

[2] 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.

[3] 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.

[4] Tsuyoshi Isshiki, Wayne Wei-Ming Dai, Hiroaki Kunieda, "Routability Analysis of Bit-Serial Pipeline Datapaths", *IEICE Trans. Fundamentals*, E80-A, pp.1861-1870, 1997.

[5] Akihisa Ohta, Tsuyoshi Isshiki, Hiroaki Kunieda, "New FPGA Architecture for Bit-Serial Pipeline Datapath", *Proc. of IEEE Symp. FPGAs for Custom Computing Machines*, 1998.

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