設 計 自 動 化 85－19
（1997. 10. 29）

ビットシリアルパイプラインデータパス合成

一色 剛　清水頭 武信　太田 章久　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 : isshiki@ss.titech.ac.jp

あらまし　チップ内部の回路規模がプロセス技術の進歩のもとに大きくなるにつれ、配線領域のチップ面積に示める割合は急速大きくなる。特にFPGAの場合は、配線資源の増加がシリコン利用率に与える影響はさらに大きい。本稿では、我々が開発した高速ビットシリアルパイプラインデータパス合成システムについて紹介する。このシステムは、配線性の極めて高いビットシリアル回路を自動生成することによって、高速で高密度な回路を効率良く実現することを可能にする。このシステムは、デザイン入力として差分方程式をC++言語の上で記述し、その後のパイプライン合成やレイアウト合成はすべて自動化されている。このシステムは、ほぼ100%の論理資源使用効率と高速クロック動作を保証する。またここでは、ビットシリアルパイプラインにおける資源の共有化や複製化についての考察を述べる。

キーワード　ビットシリアル, パイプライン合成, FPGA

High-Performance Bit-Serial Pipeline Datapath Synthesis

Tsuyoshi Isshiki, Takenobu Shimizugashira, Akihisa Ohta,
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 : isshiki@ss.titech.ac.jp

Abstract　As the circuit size inside the chip grows with the help of the advance of process technology, the device area only for wiring signals becomes significantly large. Especially in the case of FPGAs, this increase in the routing resource results in the decrease of silicon utilization, where the logic density if already a factor of magnitude lower than the full custom chips. In this paper, we present our work on the high-performance bit-serial pipeline datapath which addresses the problem of incorporating design automation while guaranteeing a highly efficient routability of the circuit, therefore increasing the silicon utilization and performance. Our system consists of algorithm-level design capture in terms of difference equations using C++, pipeline synthesis and layout synthesis. Our system guarantees a near 100% logic utilization and high speed clock operation. We also discuss the issues of resource sharing and resource duplications for bit-serial pipeline synthesis.

key words　bit-serial, pipeline synthesis, FPGA

1 Introduction

In recent years, we are witnessing a constant silicon process technology improvement which will be reaching 0.1μ in a several years. As the circuit size inside the chip grows, the device area devoted only for wiring signals becomes significantly large. Until now, this nature of the routing difficulty is more or less conceived as unavoidable for any useful circuits. 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. 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. However, its effectiveness is still questionable for its incapability to handle large problems and the quality of the produced results.

Our approach of using bit-serial circuits attempts to address these difficult problems of incorporating design automation while guaranteeing a highly efficient routability of the circuit, therefore increasing the silicon utilization and improving performance. There have been studies on bit-serial synthesis in the past [?] [?], however the efficiency of routability of bit-serial circuits has not been well investigated. In this paper, we present our work on the bit-serial pipeline synthesis system, where the algorithm-level design capture using difference equations is done on C++ language, and bit-serial circuit synthesis and layout synthesis is fully automated, while guaranteeing a maximum logic utilization and high speed clock operation. Our bit-serial pipeline synthesis system have generated a number of applications which present an unusually high routability, where 100% routability can be empirically guaranteed for all designs whose logic utilization reaches as high as 100%.

Also, we will also discuss some strategies in expanding our synthesis system, currently only able to produce a directly mapped pipeline datapath from the specified difference equations, to provide a variety of synthesis options by resource sharing or resource duplication. Although the fundamental problem formulation for resource sharing is identical to the conventional high-level synthesis paradigm, our optimization objectives, and as a result the approach to this problem, need to be tuned adequately to take advantage of the features of bit-serial circuits.

2 Bit-Serial Pipeline Synthesis System

We have developed an application development platform which targets FPGA-based configurable sys-

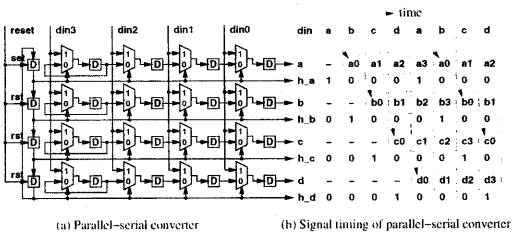


Figure 1: 4-input parallel-to-serial converter with 4-bit data.

tems where the hardware is configured for a specific application in order to accelerate the application normally running on software. The main goals of this development platform are :

1. 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.

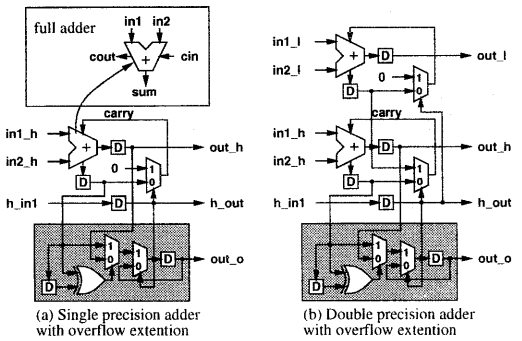
2. To guarantee accurate prediction of performance and hardware resource utilization before the lengthy layout synthesis. Totally eliminate the need for low-level manual intervention which requires not only expert digital system design skills but also the deep understanding about the target FPGA architecture.

2.1 Structure of Bit-Serial Circuits

In our bit-serial synthesis system, we provide numbers of bit-serial operators which are categorized by the number of fanouts and circuit size as follows:

1. *Parallel-serial interface circuits* : We provide parallel-to-serial and serial-to-parallel converters for interfacing with external bit-parallel devices. These circuits have some global signals whose fanouts are equal to the data word size (Fig.1).

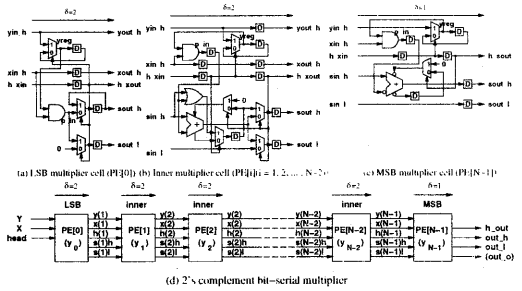
2. *Adders, subtracters, rounders* : For internal computation, we provide double precision numbers using two data lines mainly for output data of multipliers which produces double precision products. Also, in a situation where a large number of accumulations occur and overflow is unavoidable, we provide 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. There are a total of four types of adders and



(a) Single precision adder with overflow extention

(b) Double precision adder with overflow extention

Figure 2: Bit-serial adders. Hashed regions are the overflow extension circuits.



(d) 2's complement bit-serial multiplier

Figure 3: Bit-serial multiplier. It has a linear array structure.

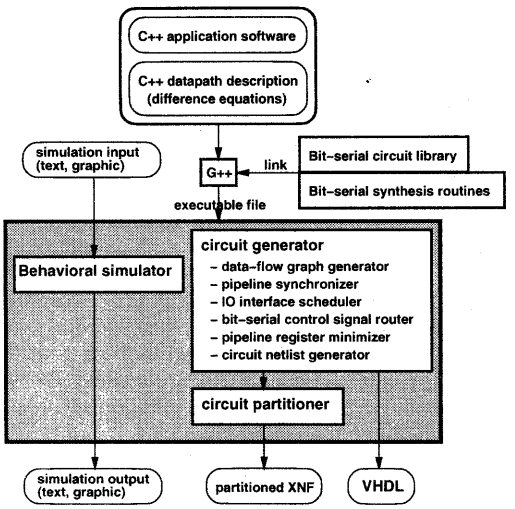


Figure 4: Design flow of our bit-serial synthesis system. The designer describes the algorithm in terms of difference equations on C++ using our dedicated class objects. The C++ program is compiled by a standard GNU C++ compiler (g++) which produces an executable file which performs behavioral simulation, circuit generation and circuit partitioning, and produced circuit-level netlist (partitioned XNF files, unpartitioned VHDL code).

subtracters for these different types of number representations (Fig.2). Rounders (round-to-nearest-integer, round-to-nearest-even) converts (extended) double precision numbers into (extended) single precision numbers. These circuits have a common circuit structure where the circuit size is independent of the data word size. There are no global signal and therefore the number of fanout is independent of N.

3. *Multipliers, saturater*: Multipliers takes two single precision inputs (or one input and a built-in constant) and generates double precision product. It consists of N multiplier cells for N-bit data where one multiplier cell is about twice the size as a double precision adder (Fig.3). Saturater converts extended single precision data into single precision data. It consists of one saturater cell and a shift-register with length $2N$. There are no global signal and therefore the number of fanout is independent of N.

In many of the applications we target, adders, subtracters and multipliers dominate the bit-serial pipeline, where multipliers are actually dominant in terms of circuit size. Bit-serial multiplier has a linear array structure, and therefore requires very little routing resource. This structural feature is actually the key in the efficiency of bit-serial circuit layout as we will discuss later.

2.2 C++ Design Capture

The design flow is illustrated in Fig. 4. First, the designer will enter the design of both the hardware and software using C++ [?]. 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*. 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 and Circuit Partitioning

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.

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 If there are loops in the datapath, the sampling period needs to be long enough so that all the loop paths have positive weights. This can be done by iteratively solving the *single-source longest paths problem* to find the minimum sampling period.

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. This pipeline scheduling optimization can be formulated as a linear programming problem and known to be efficiently solved by *simplex method* [?][?].

Bit-serial pipeline partitioning After the final bit-serial pipeline network has been synthesized, the network is partitioned into subcircuits in order to fit inside the target FPGA device. Based on Fiduccia and Mattheyses' bipartitioning [?], we formulated a K-way partitioning algorithm which maximizes logic utilization as well as minimizing the IO utilization.

3 Rent's Rule and Circuit Routability

In analyzing the circuit structure to evaluate the area required for VLSI layout, Rent's rule is commonly used [?][?][?]. This rule defines the relationship between the average number of pins and the average number of logic circuits in a subcircuit. It is expressed as

$$P = k \cdot G^{\gamma} \qquad (1)$$

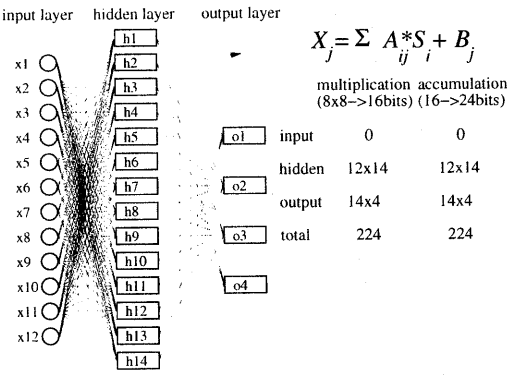


Figure 5: 12-14-4 digital neural network. Topology is a complete connection between adjacent layer nodes.

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. First discovered by E. F. Rent of IBM in the late 1960's, Donath [?] derived the same relationship from a stochastic model of hierarchical design process. From his model which assumes a two-dimensional layout, he has derived the average wire length r of the circuit using the Rent parameters as

$$r \sim \begin{cases} f(\gamma) & (\gamma < 0.5) \\ \log G & (\gamma = 0.5) \\ G^{\gamma - 0.5} & (\gamma > 0.5) \end{cases} \qquad (2)$$

where $f(\gamma)$ is a function independent of G. The intuitive explanation for this is that the case $\gamma = 0.5$ is the transition between planar and non-planar circuits, and the circuits whose Rent exponent is lower than 0.5 can be placed such that all connections essentially lie between nearest neighbors with an average wire length being independent of G. For circuits where $\gamma > 0.5$, the wire length grows to the power of $\gamma - 0.5$ with G. Therefore, the Rent exponent is a good indicator for estimating the amount of routing resources needed for the physical layout.

3.1 Routability Analysis of Bit-Serial Circuits

The bit-serial circuits which is used in this study are :

1. "N-12-14-4" : A 3-layer feedforward digital neural network with 12, 14, 4 nodes on each layer. It consists of 224 multipliers and 224 adders. Data precision is 8 bits for multipliers (product output is 16 bits) and 24 bits for adders for overflow

prevention. It also has external ROM interface for calculation of the activation function for each node. Fig. 5 shows the basic architecture of this digital neural network. The dense interconnection between layers makes the routing problem very challenging.

2. "N-8-8-4" : A 3-layer feedforward digital neural network with 8, 8, 4 nodes. There are 96 multipliers and adders.

3. "1D-FIR-30-II" : A 30-tap 1D FIR filter. Data precision is 16 bits with double precision (32-bit) addition.

4. "2D-FIR8x8" : An 8×8-tap 2D FIR filter. Data precision is 8 bits with double precision (16-bits) addition.

5. "1D-IIR20-III" : A 20-th order IIR filter with 2 sampling period lookahead on the feedback loop. Data precision is 16 bits with double precision addition.

6. "adapt10T-7" : A 10-tap 1D adaptive FIR filter where the coefficient updating cycle delay is 7 sampling periods. Data precision is 8 bits with double precision addition.

7. "IDCT-I" : An 8-point inverse-DCT. Data precision is 16 bits with double precision addition.

We have obtained the Rent's parameters for our 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. Here, CLB is the logic block of Xilinx 3100A FPGA which is our current target FPGA architecture. Note that Rent's rule on log-log scale forms a linear equation $\log P = \gamma \log G + \log k$. Fig.?? shows the plots for neural network designs. The Rent's parameters for other designs are summarized in Table ??. 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 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, which will be strongly confirmed in our physical layout results described in the next section.

Table 1: Rent's parameters for bit-serial circuits.

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

3.2 Physical Layout Results

We have used Xilinx 3100A FPGA architecture as the target FPGA to actually map our bit-serial designs and observe the logic resource utilization and performance. It has been pointed out that logic utilization over 80% for "real-life" circuits has little chance of being routed successfully [?]. In our case, *all* circuits were routable at the first run where most of the subcircuit had logic utilization of over 95% (in fact more than half were over 99%). This kind of high logic utilization has never been reported with the use of automated placer and router. This indicates how efficient it is to place and route our bit-serial circuits which is backed up by their low Rent exponent values. Although, unfortunately, because of the variance in the routing performance, ppr is not capable of producing stable circuit performance even for our bit-serial circuits. We have set our performance goal to 40MHz using the XC3100A FPGA series which is currently the fastest device available from Xilinx.

4 Pipeline Synthesis with Resource Sharing and Resource Duplication

So far, we have discussed our works on bit-serial pipeline synthesis system and its advantages on routability. In our current system, however, we are only able to produce a directly-mapped pipeline network which is described in terms of difference equations. In other words, for each set of difference equations, only one pipeline network solution is derived. There are, of course, some freedom in the allocation of

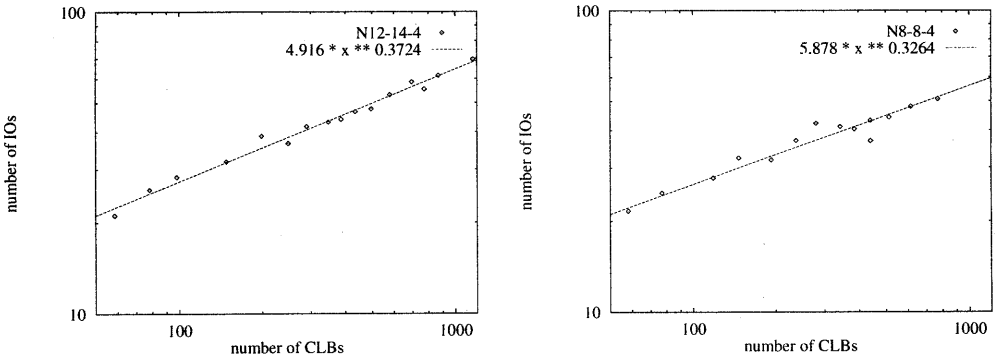


Figure 6: Rent's exponent approximation for N12-14-4 and N8-8-4. The two axes are log-scale.

Table 2: Synthesis results of bit-serial pipeline networks. "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	signal precision	# CLBs	# gates	# chips (XC3164A)	CLB util.	IO util.	critical delay	sampling frequency (@40MHz)
N-12-14-4	8	6964	127962	32	97.2%	30.3%	24.6ns	5.0MHz
N-8-8-4	8	3082	56465	14	98.3%	30.7%	24.6ns	5.0MHz
1D-FIR30-II	16	2004	35813	9	99.4%	17.4%	20.3ns	2.5MHz
2D-FIR8x8	8	2272	39957	11	92.2%	21.1%	25.0ns	5.0MHz
1D-IIR20-III	16	2132	41165	10	98.1%	22.9%	23.4ns	2.5MHz
adapt10T-7	8	871	16036	4	97.5%	27.3%	22.8ns	5.0MHz
IDCT-I	16	1038	15909	5	90.4%	23.7%	23.2ns	2.5MHz

the pipeline retiming registers. However, this register allocation can be optimized using simplex method on linear programming, which is done in our bit-serial pipeline synthesis system. We are currently working on expanding the synthesis system to provide wider choices of synthesis strategies by allowing resource sharing or resource duplication. In this section, we discuss some of the issues regarding resource sharing and resource duplication.

There are several situations where resource sharing and resource duplication are very useful (Fig.??.

1. Throughput of the bit-serial pipeline datapath is determined by the data word length and the critical loop length, if any exists. Data word length gives the lower bound of the sampling period, i.e., if data is N bits, it takes N clock cycles to transmit one data. If there is a critical loop whose length T is longer than N, $T - N$ clock cycles would be idle. If $T \geq 2N$, then there is an opportunity to share the hardware resource to make the datapath more efficient in terms of circuit size versus performance. In other cases where the application can afford to be slower in exchange for smaller hardware, this resource sharing can be effective.

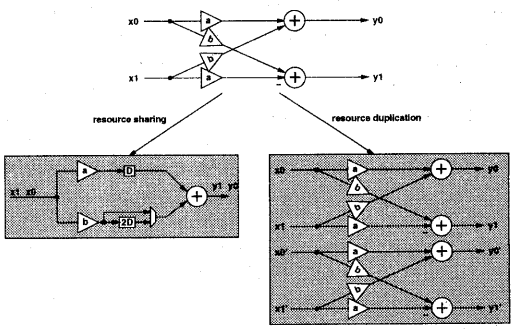


Figure 7: Resource sharing and resource duplication.

2. Resource duplication can be used if the applica-

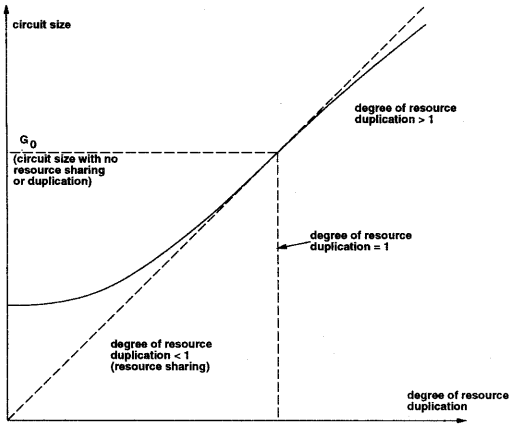


Figure 8: Relationship of resource sharing and resource duplication. Horizontal axis represent the degree of resource duplication. The region where the resource duplication is below 1 corresponds to resource sharing.

tion requires high throughput. If the hardware is duplicated by K, the sampling period becomes $\lfloor N/K \rfloor$. If there is a critical loop whose length is T, then the inequality $\lfloor N/K \rfloor \geq T$ needs to be satified. It is, therefore, essential in this situation to design the application with care so that the critical loop length T becomes as small as possible. Theoretically, if the application consists of acyclic computations only, then the degree of duplication K can be any positive number.

3. In complex applications where the data rate differs between locations in the data-flow, such as in multi-rate signal processing applications, throughput needs to be adjusted throughout the pipeline network. This can be achieved by either resource sharing on the low data rate signals, or resource duplication on the high data rate signals, or combination of both.

The relation of resource sharing and resource duplication against circuit size can be illustrated as in Fig.??. Horizontal axis represents the degree of resource duplication (D_{rd}). The region where the resource duplication is below 1 corresponds to resource sharing. Degree of resource sharing D_{rs} on this region is given as $D_{rs} = 1/R_{rd}$. Vertical axis represents the circuit size. When resource sharing is applied, the hardware size decrease but also introduces overhead circuitry for multiplexing the data and controlling the data-flow. The number of pipeline retiming registers may also increase in the case where the sampling period increases as the result of resource sharing. Therefore, when the degree of resource sharing is D_{rs}, the circuit size will become larger than G_0/D_{rs}. Also, as D_{rs} increases, the overhead also increases as well. On the other hand, when resource duplication is applied, if allowed, then the hardware size grows in return for higher performance. There are virtually no overhead circuits for realizing resource duplication. Moreover, the number of pipeline retiming registers in this case may *decrease* since the sampling period decreases as well. Therefore, when the degree of resource duplication is D_{rd}, the circuit size may actually become smaller than G_0/D_{rd}.

There are another important aspect of this relationship of resource sharing and resource duplication. As we have discussed earlier, high routability is a very important feature of bit-serial pipeline datapath, especially when implemented on FPGAs. Resource sharing involves multiplexing the data-flow which has a large impact on the locality of interconnections in bit-serial pipeline circuits. As the degree of resource sharing increases, the bit-serial circuits tend to be more globally connected, which increases the Rent exponent. As a result, this will diminish the very advantage of using bit-serial circuits. We are investigating heuristic approaches to handle the scheduling problem of resource sharing which attempts to minimize the Rent exponent in order to increase the routability.

5 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. Where 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. We believe that our work provides some of the key solutions in overcoming the problem of increasing routing cost with the growth of circuit size.

In particular, FPGA devices which are forced to handle such hard-to-route real-life circuits have been increasing the routing resource as the device size grows, and therefore continues to decrease the silicon density even further. The implications of the low Rent exponent value of bit-serial circuits are:

1. The average wiring length is expected to be independent of the circuit size. Therefore, we do not need to increase the routing resource or the IO pin count as the chip size increases in order to

utilize 100% of the hardware resource. This situation is clearly different from the current trend of FPGA architecture development where additional long-lines are added for larger chips as in Xilinx XC4000 series and XC6200 series.

2. Because of the high routability, hardware size estimation and performance prediction can be made accurate before the time-consuming layout synthesis. This facilitates the designers to do designs on high-level and effectively explore the design space and yet achieve very efficient designs.

We are in the process of expanding our bit-serial synthesis system to handle resource sharing and resource duplication in order to handle multi-rate signal processing applications and other complex applications such as 3D graphics.

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] R. Jain, F. Catthoor, J. Vanhoof, B. J. S. De Loore, G. Goossens, N. F. Goncalvez, L. J. M. Claesen, J. K. J. Van Ginderdeurn, J. Vandewalle and H. J. De Man, "Custom Design of a VLSI PCM-FDM Transmultiplexer from System Specifications to Circuit Layout Using a Computer-Aided Design System," *IEEE Jo. Solid-State Circuits*, pp.73-84, 1986.

[4] A. F. Murray and P. B. Denyer, "A CMOS Design Strategy for Bit-Serial Signal Processing," *IEEE Jo. Solid-State Circuits*, pp.746-753, 1985.

[5] 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.

[6] S. Kumar, K. Forward and M. Palaniswami, "A Fast-Multiplier Generator for FPGAs," *Proc. IEEE Int. Conference on VLSI Design*, pp.53-56, 1995.

[7] C. E. Leiserson and J. B. Saxe, "Optimizing Synchronous Systems," *J. VLSI and Comput. Syst.*, vol. 1, no. 1, pp. 41-67. Oct. 1983.

[8] X. Hu, S. C. Bass and R. G. Harber, "Minimizing the Number of Delay Buffers in the Synchronization of Pipelined Systems," *IEEE Trans. Computer Aided Design*, pp. 1441-1449, Dec. 1994.

[9] C. M. Fiduccia and R. M. Mattheyses, "A Lenear-Time Heuristic for Improving Network Partitions," *Proc. 19th ACM/IEEE Design Automation Conference*, pp.241-247, 1982.

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

[11] Abba A. El Gamal, "Two-Dimensional Stochastic Model for Interconnections in Master Slice Integrated Circuits," *IEEE Trans. Circuits and Systems*, pp.127-138, Feb. 1981.

[12] M. Feuer, "Connectivity of Random Logic," *IEEE Trans. Comp.*, Vol. C-31, pp.29-33, Jan. 1982.

[13] Steve M. Trimberger, "Field-Programmable Gate Array Technology," *Kluwer Academic Publishers*, 1994.

[14] M. Schlag, J. Kong and P. K. Chan, "Routability-Driven Technology Mapping for Lookup Table-Based FPGA's," *IEEE Trans. Computer-Aided Design*, pp.13-26, Jan. 1994.

[15] "The Programmable Gate Array Data Book," Xilinx, 1994.