An Area-Efficient Asynchronous FPGA Architecture for Handshake-Component-Based Design

Yoshiya Komatsu, Masanori Hariyama, and Michitaka Kameyama
Graduate School of Information Sciences, Tohoku University
Aoba 6-6-05, Aramaki, Aoba, Sendai, Miyagi, 980-8579, Japan

Abstract— *This paper presents an area-efficient FPGA architecture for handshake-component-based design. The handshake-component-based design is suitable for large-scale, complex asynchronous circuit because of its understandability. However, conventional FPGA architecture for handshake-component-based design is not area-efficient because of its complex logic blocks. This paper proposes an area-efficient FPGA architecture that combines complex logic blocks (LBs) and simple LBs. Complex LBs implement handshake components that implement data path controller, and simple LBs implement handshake component that implement data path. The FPGA based on the proposed architecture is implemented in a 65nm process. Its evaluation results show that the proposed FPGA can implement asynchronous circuits efficiently.*

Keywords: FPGA, Reconfigurable LSI, Self-timed circuit, Asynchronous circuit

1. Introduction

Field-programmable gate arrays (FPGAs) are widely used to implement special-purpose processors. FPGAs are cost-effective for small-lot production because functions and interconnections of logic resources can be directly programmed by end users. Despite their design cost advantage, FPGAs impose large power consumption overhead compared to custom silicon alternatives [1]. The overhead increases packaging costs and limits integrations of FPGAs into portable devices. In FPGAs, the power consumption of clock distribution is a serious problem because it has an enormously large number of registers than custom VLSIs. To cut the clock distribution power, some asynchronous FPGAs has been proposed [2], [3], [4], [5], [6]. However, the problem is that it is difficult to design asynchronous circuits and few CAD tools or design flow for asynchronous FPGAs have been introduced. To solve the problem, we proposed an FPGA architecture for handshake-component-based asynchronous circuit design (HCFPGA) [7]. In handshake-component-based design, asynchronous circuits are designed by connecting handshake components. Since various handshake components such as for data processing and data path control are defined, it is easy to design asynchronous data path and its controller. Besides, there are hardware description languages and circuit synthesis tools for handshake-component-based design [8], [9]. Therefore, handshake-component-based design is suitable for complex large-scale asynchronous circuits. However, the problem of the previous HCFPGA is its large transistor count because each FPGA cell is complex to support various handshake components.

This paper proposes an area-efficient HCFPGA architecture that combines complex LBs and simple LBs. As the proposed architecture implements handshake components efficiently, CAD tools such as Balsa [9] are utilized to design asynchronous applications. Data path and its controller are implemented by simple LBs and complex LBs respectively. Therefore, the proposed architecture can implement applications efficiently.

2. Architecture

2.1 Handshake-component-based design methodology

In asynchronous circuits, the handshake protocol is used for synchronization instead of using the clock. Figure 1 shows a four-phase handshake sequence. First, active port sets the request wire to "1" as shown in Fig. 1(a). Second, passive port sets the acknowledge wire to "1" as shown in Fig. 1(b). Third, active port sets the request wire to "0" as shown in Fig. 1(c). Finally, passive sets the acknowledge wire to "0" as shown in Fig. 1(d) and wire values return to initial state. Data signals are sent along with request signals or acknowledge signals.

Handshake components were proposed for use in the synthesis of the language Tangram [8] created by Philips Research. An asynchronous functional element such as a binary operator is denoted by a handshake component. There are 46 handshake components [10] and each handshake component is used for data processing or data path control. Figure 2 shows handshake components. Handshake components constitute a handshake circuit. Figure 3 shows an example of a handshake circuit. Each handshake component has ports and is connected to another handshake component through a channel. Communication between handshake components is done by sending request signal from the "active" port and acknowledge signal from the "passive" port. Depending on the kind of handshake components, data signals are sent along with request signals or acknowledge signals. The number of

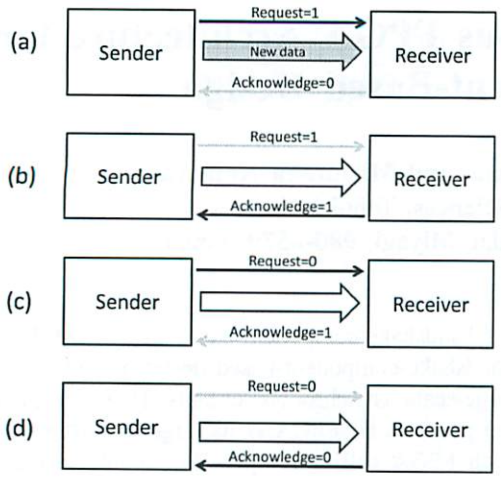

Fig. 1: A four-phase handshake sequence.

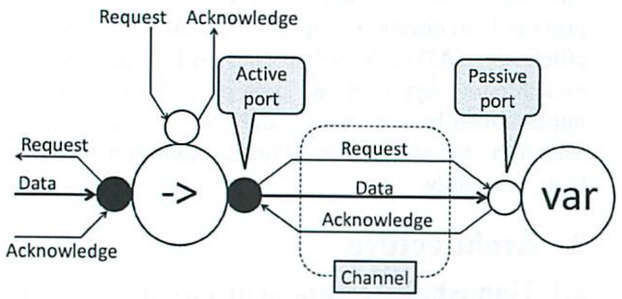

Fig. 2: Handshake components and channels.

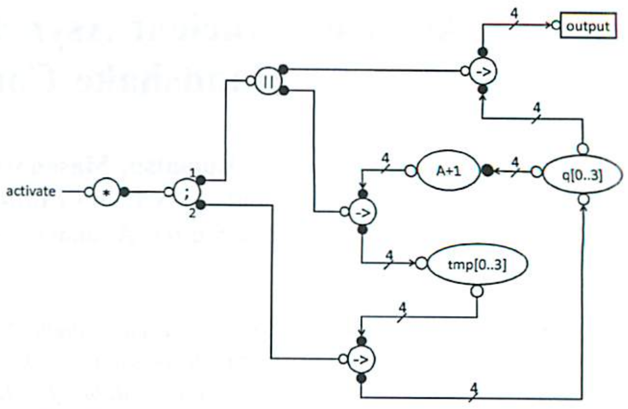

Fig. 3: A simple handshake circuit (4 bit counter).

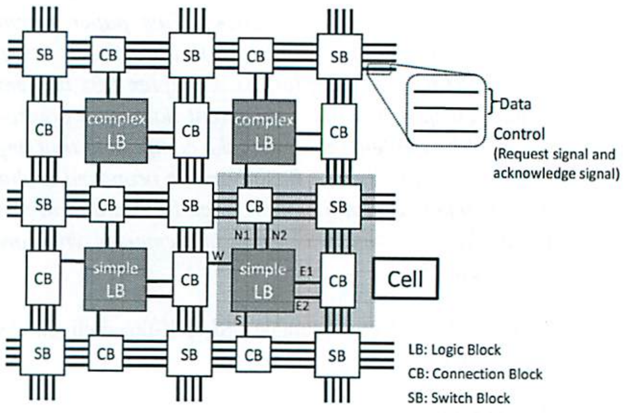

Fig. 4: Overall architecture.

ports of a handshake component and the width of data signal can be varied. Each handshake components execute complex handshake sequences through channels. However, handshake circuits are easily understandable and manageable because a function of each handshake component is clear and each handshake is symbolized by a channel and ports. Also, there are tools that translate high-level circuit description into handshake circuit to synthesize asynchronous circuit. Thus, handshake-component-based design is suitable for complex and large-scale asynchronous circuits. Asynchronous circuit synthesis is done by replacing each handshake component with corresponding circuit.

2.2 Overall architecture

As mentioned in preceding section, circuit synthesis is done by replacing each handshake component with corresponding circuit. Thus, asynchronous circuits can be implemented by replacing each handshake component with a combination of LBs. Figure 4 shows the overall architecture of the proposed FPGA. The FPGA consists of mesh-connected cells like conventional FPGAs. Each cell includes an LB, two Connection Blocks (CBs) and a Switch Block (SB). There are two types of LBs. One is complex LB and the other is simple LB. The upper CB connects SBs to N1, N2 and S terminals of two LBs, and the bottom CB connects SBs to E1, E2 and W terminals. In the proposed architecture, each LB includes dedicated circuits for implementing handshake components. Therefore, the proposed architecture can implement handshake circuits efficiently. The proposed architecture can implement 39 out of 46 handshake components defined in Balsa manual [10]. Handshake components that have multiple ports or wide data path can be implemented using several LBs. In the proposed FPGA architecture, the Four-Phase Dual-Rail (FPDR) encoding is employed for asynchronous data encoding. The FPDR encoding encodes a bit and a request signal onto two wires. Table 1 shows the code table of the FPDR encoding. The main feature is that the sender sends a spacer and a valid data alternately as shown in Fig. 5. FPDR circuits are robust to the delay variation. Hence, the FPDR encoding is the ideal one for FPGAs in which the data path is programmable. Because the FPDR encoding is employed, three wires are required for a data bit. Two wires are used for the data encoded in FPDR encoding, and one wire for the acknowledge signal.

Table 1: Code table of the FPDR encoding.

	Code word (T, F)
Data 0	(0,1)
Data 1	(1,0)
Spacer	(0,0)

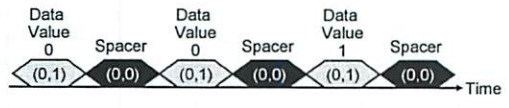

Fig. 5: Example of the FPDR encoding.

2.3 Logic block structure

As mentioned in 2.2, there are complex LB and simple LB. Figure 6 and 7 show the structures of a complex LB and simple LB. Complex LB consists of a BinaryFunction module, a Variable module, a Sequence module, a CallMUX module, a Case module, an Encode module, an Input switch box and an Output switch box. Simple LB consists of a BinaryFunction module, a Variable module, a C-element, an Input switch box and an Output switch box. An Input switch box and an Output switch box connect modules to CBs. Each module is used to implement a handshake component. Table 2 shows correspondence relation between modules and handshake components. Complex LB can supports 39 handshake components because it has all the modules. On the other hand, simple cell can implement 22 handshake components including Variable component and BinaryFunction component. Therefore, complex LB is suitable for implementing data path controller and simple LB can implement data path efficiently.

3. Evaluation

The proposed FPGA is implemented in e-Shuttle 65nm CMOS process with 1.2V supply. The circuits are evaluated using HSPICE simulation. Table 3 shows the comparison

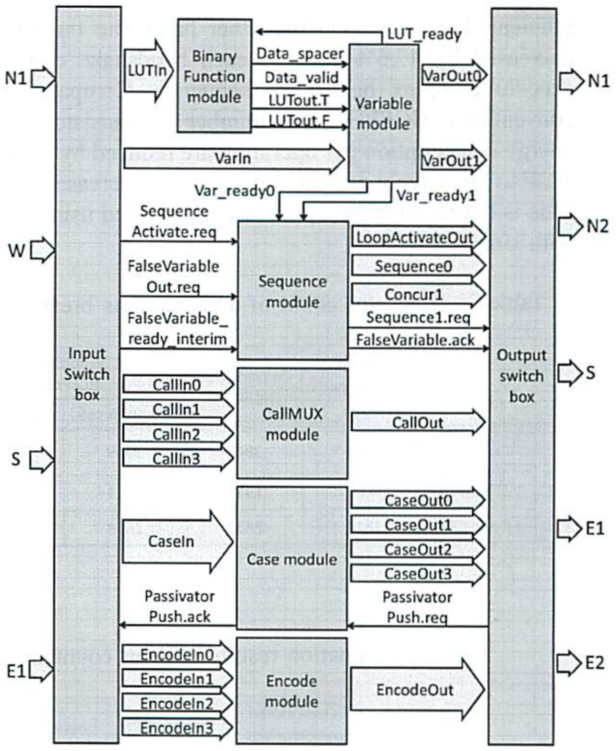

Fig. 6: Structure of a complex LB.

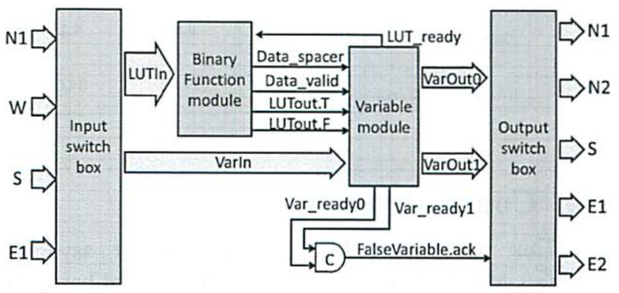

Fig. 7: Structure of a simple LB.

Table 2: Handshake components and its corresponding resources.

Module	Handshake component
Variable	BuiltinVariable, Variable
Sequence	Concur, Loop, Sequence, While
CallMUX	Call, CallMUX, Continue, ContinuePush
Case	CallDEMUX, Case, CaseFetch, DecisionWait, PassivatorPush, SynchPush
Encode	Encode
BinaryFunction and Variable	BinaryFunc, BinaryFuncConstR, UnaryFunc
Variable and Sequence	FalseVariable, ActiveEagerFalseVariable, PassiveEagerFalseVariable
Programmable Interconnect resources	Adapt, Combine, CombineEqual, Constant, Fetch, Fork, ForkPush, Halt, HaltPush, Passivator, Slice, Split, SplitEqual, Synch, SynchPull, WireFork

result of cells of the conventional asynchronous FPGA, the conventional HCFPGA and the proposed HCFPGA. Compared to the conventional asynchronous FPGA cell, the transistor count of the complex cell is increased by 63.0% because the complex cell is the same as the conventional HCFPGA cell. The transistor count of the simple cell is reduced by 31.0% compared to the complex cell.

The next evaluation shows the implementation results of a 4-bit counter. Table 4 shows the comparison of transistor counts, energy consumptions per operation and throughputs. Compared to the conventional asynchronous FPGA, the number of transistors and the energy consumption per operation are reduced by 4.4% and 19.8% respectively. This is because handshake-component-based design method is suitable for designing not only controllers but also area-efficient data paths. On the other hand, the throughput is decreased by 47.6% because each handshake components execute complex handshake sequence. Compared to the conventional HCFPGA, the number of transistors and the energy consumption per operation are reduced by 25.3% and 11.8% respectively and the throughput is increased by 7.9%. This is because the data path is implemented using the cells with simple LB.

Table 3: Transistor count of a cell and its breakdown.

	Conventional Asynchronous FPGA	Conventional HCFPGA	Proposed HCFPGA	
			Complex cell	Simple cell
Cell	2423	3949	3949	2726
LB	611	1311	1311	611
SB and CBs	1812	2638	2638	2115

Table 4: Evaluation results of 4-bit counter.

	Conventional Asynchronous FPGA	Conventional HCFPGA	Proposed HCFPGA
Number of transistors	33922	43439	32432
Energy Consumption [pJ]	5.14	4.68	4.13
Throughput [M operations/sec]	160.63	77.93	84.09

4. Conclusions

This paper presented an area-efficient asynchronous FPGA architecture for handshake-component-based design. In the proposed HCFPGA architecture, simple LB and complex LB are used to implement a data path and its controller respectively. Therefore, the proposed architecture implements applications efficiently. As a future work, we are evaluating the proposed FPGA architecture on some practical benchmarks.

Acknowledgment

This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with STARC, e-Shuttle, Inc., Fujitsu Ltd., Cadence Design Systems Inc. and Synopsys Inc.

References

[1] V. George H. Zhang. and J. Rabaey, "The design of a low energy FPGA," in *Proceedings of 1999 International Symposium on Low Power Electronics and Design*, California, USA, Aug 1999, pp. 188–193.

[2] J. Teifel and R. Manohar, "An asynchronous dataflow FPGA architecture," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1376–1392, 2004.

[3] R. Manohar, "Reconfigurable Asynchronous Logic," in *Proceedings of IEEE Custom Integrated Circuits Conference*, Sep. 2006, pp. 13–20.

[4] M. Hariyama, S. Ishihara, and M. Kameyama, "Evaluation of a Field-Programmable VLSI Based on an Asynchronous Bit- Serial Architecture," *IEICE Trans. Electron*, vol. E91-C, no. 9, pp. 1419–1426, 2008.

[5] M. Hariyama, S. Ishihara, , and M. Kameyama, "A Low-Power Field-Programmable VLSI Based on a Fine-Grained Power-Gating Scheme," in *Proceedings of IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, Knoxville(USA), Aug 2008, pp. 430–433.

[6] S. Ishihara, Y. Komatsu, M. Hariyama and M. Kameyama, "An Asynchronous Field-Programmable VLSI Using LEDR/4-Phase-Dual-Rail Protocol Converters," in *Proceedings of The International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, Las Vegas(USA), Jul 2009, pp. 145–150.

[7] Y. Komatsu, M. Hariyama and M. Kameyama, "Architecture of an Asynchronous FPGA for Handshake-Component-Based Design," *Proc. International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, pp. 133-136, July 2012

[8] K. van Berkel, J. Kessels, M. Roncken, R. Saeijs, and F. Schalij, " The VLSI-programming language Tangram and its translation into handshake circuits, " in *Proc. EDAC*, 1991, pp. 384—389.

[9] A. Bardsley, "Implementing Balsa Handshake Circuits," Ph.D. thesis, Dept. of Computer Science, University of Manchester, 2000.

[10] Doug Edwards and Andrew Bardsley and Lilian Janin and Luis Plana and Will Toms, "Balsa: A Tutorial Guide", 2006.