Abstract—This paper presents a novel architecture of an asynchronous FPGA for handshake-component-based design. The handshake-component-based design is suitable for large-scale, complex asynchronous circuit because of its understandability. This paper proposes an area-efficient architecture of an FPGA that is suitable for handshake-component-based asynchronous circuit. Moreover, the Four-Phase Dual-Rail encoding is employed to construct circuits robust to delay variation because the data paths are programmable in FPGA. The FPGA based on the proposed architecture is implemented in a 65nm process. Its evaluation results show that the proposed FPGA can implement handshake components 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 have been proposed [2], [3], [4], [5], [6].

In asynchronous FPGAs, CAD tools that is different from ones for synchronous FPGAs is necessary to implement applications. Although, few CAD tools or design flow for asynchronous FPGAs have been introduced. As the design methods for asynchronous circuits, some method uses the Signal Transition Graph[8] and another method employs handshake components[9][10]. Handshake-component-based design is easy to understand and easy to construct datapath. Besides, Balsa[10] is proposed as a design methodology that uses handshake components. Balsa is a hardware description language and it allows circuit designers not to pay attention to low-level details such as control of handshake. Thus, it is suitable for designing complex large-scale circuits such as a DMA controller[10] and a microprocessor[11]. In Balsa, 46 handshake components are defined and complex asynchronous circuits are synthesized by combining them. Moreover, there are synthesis tools that generate a handshake circuit that consists of handshake components and a netlist consists of standard cells. Therefore, Balsa is desirable as a inputs of CAD tools for asynchronous FPGAs.

This paper proposes an area-efficient architecture of an FPGA that is suitable for handshake-component-based asynchronous circuit. The proposed architecture implements handshake components that is defined in Balsa efficiently. Small frequently-used handshake components are implemented on a logic block (LB), and other handshake components are implemented using more than one LB. As handshake components can be mapped directly on the proposed architecture, circuit designers can utilize existing CAD tools that generate a netlist of handshake components. Therefore, a design method for the proposed FPGA is established.

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, the sender sets the request wire to 1 as shown in Fig. 1(a). Second, the receiver sets the acknowledge wire to 1 as shown in Fig. 1(b). Third, the sender sets the request wire to 0 as shown in Fig. 1(c). Finally, the receiver sets the acknowledge wire to 0 as shown in Fig. 1(d) and wire values return to initial state.

A asynchronous functional element such as a binary operator is denoted by a handshake component. Figure 2 shows handshake components. 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 ports and the width of data signal can be varied. Each function of handshake component is simple and clear. Furthermore, handshaking that consists of request signal and acknowledge signal is symbolized as a channel. Therefore, handshake circuits are easily understandable and manageable.

Handshake components constitute a handshake circuit. Figure 3 shows an example of a handshake circuit. Circuit synthesis is done by replacing each handshake component with corresponding asynchronous circuit.

**2.2 FPGA architecture for Handshake-component-based design**

As mentioned in preceding section, circuit synthesis is done by replacing each handshake component with corresponding asynchronous circuit. Thus, asynchronous circuits can be implemented on a conventional FPGA by replacing each handshake component with a combination of LBs. However, because it is difficult to implement the C-element that is frequently used in asynchronous circuit area-efficiently, hardware cost of a handshake component becomes large. 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 37 out of 46 handshake components defined in Balsa. Handshake components that have multiple ports or wide datapath can be implemented using several LBs.

**2.3 Overall architecture**

Figure 4 shows the overall architecture of the proposed FPGA. The FPGA consists of a mesh-connected cellular array like conventional FPGAs. 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.
### 2.4 Logic block structure

Figure 6 shows a LB of the proposed architecture. The proposed FPGA architecture can implement 37 handshake components. The LB consists of an LUT, a Variable module, a While module, a Call module, a Case module, an Encode module, multiplexers and a demultiplexer. The detailed circuits of modules are shown in Fig. 7, 8, 9 and 10. As shown in Table 2, each module implements several handshake components. In addition, several handshake components are implemented by employing programmable interconnection resources or combining two modules as shown in Table 3. The number of the transistors of the proposed FPGA is small because of resource sharing.

### 3. Evaluation

The proposed FPGA is implemented in a 65nm CMOS process. Table 4 shows the comparison result of the cells of the proposed architecture and the conventional architecture. The number of transistors of the proposed architecture is 2.5 times larger than the conventional one. Table 5 shows the implementation result of the Case handshake component that has four output ports. Compared to conventional architecture, the number of the transistors is reduced by 37%. This is because the proposed architecture requires one cell for implementing the four-output Case handshake component while the conventional architecture requires four cells.
4. Conclusions

This paper presents an architecture of an asynchronous FPGA for handshake-component-based design. The proposed FPGA architecture implements handshake components efficiently. Therefore, the proposed architecture is suitable for the synthesis tools that generate netlists of handshake components, such as Balsa. 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 STARIC, 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] J. Sparsø and S. Furber, Principles of Asynchronous Circuit Design: A Systems Perspective, Kluwer Academic Publishers, 2001.

[8] J. Cortadella, M. Kishinevsky, A. Kondratiev, L. Lavagno, A. Yakovlev, Logic Synthesis of Asynchronous Controllers and Interfaces, Springer, 2002.

[9] 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.

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

[11] Q. Zhang, G. Theodoropoulos, “Modelling SAMIPS: A Synthesizable Asynchronous MIPS Processor,” Proc. of the 37th Annual Simulation Symposium, pp. 205-212, 2004

Table 2: Handshake components and its corresponding modules.

| Module | Handshake component           |
|--------|------------------------------|
| Variable | Variable, FalseVariable, ActiveEagerFalseVariable |
| While   | While, Loop, Sequence        |
| Call    | Call, CallMux, ContinuePush  |
| Case    | Case, CaseFetch, PassivatorPush, SynchronPush, Calldemux, DecisionWait |
| Encode  | Encode                       |

Table 3: Other handshake components and its corresponding resources.

| Module       | Handshake component      |
|--------------|--------------------------|
| LUT, Variable| BinaryFunc, BinaryFuncConstR, UnaryFunc |
| Variable, While | Concur               |
| Programmable interconnect resources | Adapt, Combine, CombineEqual, Constant, Continue, Fetch, Fork, ForkPush, Halt, Haltpush, Slice, Split, SplitEqual, Synchron, SynchronPull, WireFork |

Table 4: Comparison of cells of the conventional architecture and the proposed architecture.

|                         | Conventional architecture | Proposed architecture |
|-------------------------|--------------------------|-----------------------|
| Number of transistors   | 1344                     | 3372                  |

Table 5: Comparison of cells that implement Case handshake components.

|                         | Conventional architecture | Proposed architecture |
|-------------------------|--------------------------|-----------------------|
| Number of transistors   | 5376                     | 3372                  |