Automatic Generation of Communication Paths between Multiple FPGA Boards

Hiroaki Suzuki[1,a)] Kensuke Iizuka[1] Hideharu Amano[1] Wataru Takahashi[2,b)]
Kazutoshi Wakabayashi[3]

Abstract: Programming in a multi-FPGA system requires the programmers to make tables for setting communication paths between boards, and there is a problem that the communication paths may change depending on the board used. In this paper, we have improved the flow of the multi-FPGA board design environment for the M-KUBOS cluster by dividing applications and automatically generating communication channel configuration tables for use in actual implementation using CyberWorkBench (CWB), a high-level synthesis tool with extended functions and SystemC.

Keywords: FPGA system, Design environment, CyberWorkBench

1. Introduction

Multi-access Edge Computing (MEC) attracts attention. This is because, with the recent spread of 5G technology, processing that was conventionally performed in the cloud will now be performed by a computing system installed in a base station [1].

Multi-FPGA system is advantageous for MEC because of its scalability and low energy consumption. However, it is difficult and requires time and effort to develop a program for multi-FPGA systems because it is usually necessary to split the application so that it will fit on each board. Even after splitting, it is also necessary for a human to write a table to set the communication paths between boards. This table must be set manually each time the communication path changes depending on the number of boards or links used, and board assignment. There is a position dependency that the communication path changes depending on which group of boards is used among multiple boards.

In this paper, we have divided applications on the tool side using the high-level synthesis tool CyberWorkBench (CWB)[2] and SystemC, a hardware modeling language, for the multi-FPGA system M-KUBOS cluster[3]. FPGA boards in this cluster communicate with each other via Static Time Division Multiplexing (STDM) switches and these switches are required to set communication path setting tables in advance. The design flow of the multi-FPGA design environment was also improved by automatically generating the communication path setting table that is required when actually implementing the system in a multi-FPGA system.

2. Overall design flow

The overall design flow with the proposed design tools is shown in Figure 2. First, the user describes the application to be implemented, module by module, using SystemC. Next, CWB generates RTL files according to the user-described modules and split points. The flow table file which shows the flow structure of the application is also generated. Generated RTL files are used for implementation with vendor's tools to generate configuration bitstream files for each FPGA board. Now, the allocation of the FPGA board is done manually. The communication path setting tools described here generate slot tables and virtualization tables from the flow table and topology file which shows the current connection structure detail.

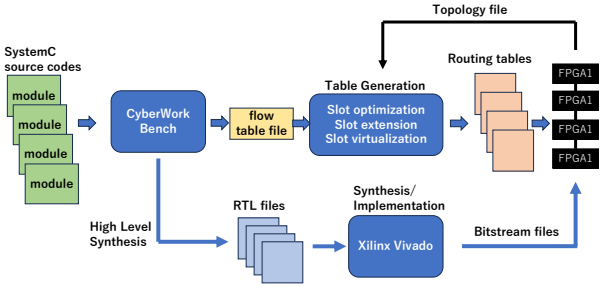

Fig. 1 Overall design flow

3. Automatic generation of communication path setting file

We propose a method to reduce the effort of multi-FPGA implementation by automatically generating communication channel configuration files after high-level synthesis.

3.1 Generating the flow table on the CWB

Figure 2 describes the flow table generation process in the example of dividing the LeNet into three parts. Since it is possible to grasp what kind of communication occurs between boards at the time of division, a flow number is assigned to the division between the boards. The result is outputted as a flow table in a traffic file. The flow table is represented by three columns: the

[1] Keio University, Yokohama, Kanagawa 223-8522, Japan
[2] NEC, Minato, Tokyo 108-8001, Japan
[3] The University of Tokyo, Bunkyo, Tokyo, 113-0032, Kapan
[a)] fic@am.ics.keio.ac.jp
[b)] nec_crest@am.ics.keio.ac.jp

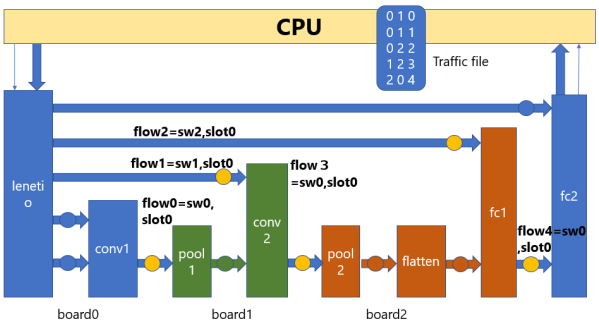

Fig. 2 Flow table generation

first column is the source board, the second column is the destination board, and the third column is the unique flow number (flow-id). In the example in the figure, line 1, 0 1 0, there is a communication path from board0 to board1, designated as flow0. Even in the same application, the flow table generated will differ depending on the user's division method, and a communication path configuration file is automatically generated based on this flow table.

3.2 The tool to generate the communication channel configuration file

Our tool generates the communication channel configuration file from this flow table generated on the CWB. The proposed tool consists of three parts: the slot optimization tool based on the algorithm proposed in [4], the extending multi-switch tool, and the tool to generate a virtualization table to convert flow-id to slot number and vice versa. The slot optimization tool routes STDM slots appropriately based on the flow table. However, this tool is limited to only one switch. The extending multi-switch tool extends this restriction of one switch to four for M-KUBOS. The last tool generates a virtualization table to convert the flow-id assigned to the upper bits in the transmitted data packets to the slot number on STDM switches assigned by the slot optimization tool. The receiving board can also convert the slot number to the flow-id in the reverse direction using the virtualization table.

A virtualization table exists on both the sender and receiver sides. The flow-id is set to the upper bits of the data packet to be sent, and the sender side automatically assigns slots according to this flow-id value and virtualization table. Conversely, the receiver side inverts the converted value to the original flow-id, and on the receiver side, each module runs its processing according to the flow-id. For example, in the case of Figure 2, when the flow-id is 0 (flow0) in the data from board0, this data is sent to the pool1 module on board1, and when the flow-id is 1 (flow1), the data is sent to the conv2 module on board1. Here, the flow-id must be embedded in the transmitted data packet, done by the applications generated by the CWB, which adds the flow-id to the upper bits of the data transmitted between boards based on the partitioning structure in CWB.

The virtualization table files and the communication channel configuration files are set to the FPGA continuously. They are mapped into the same address space in the hardware. Eventually, this tool can minimize the number of slots for applications and achieve the same communication performance when setting manually.

Table 1 Execution time when using manual partitioning and partitioning functions

	manual partitioning[ms]	partitioning function[ms]
1	78.917	78.886
2	78.903	78.887
3	78.857	78.896
4	78.900	78.864
5	78.869	78.92
average	78.890	78.892

4. Evaluation

In this evaluation, we measured the execution time of two cases: one in which LeNet was divided manually on CWB, and the other in which CWB division was used and the communication path setting table was generated automatically. The target multi-FPGA system is M-KKUBOS cluster[3]. Cases with two boards and three boards were tried. Even in the case of using CWB splitting, evaluations were taken for the case where multiple switches were used and the case where only one switch was used and multiple STDM slots were used. For the evaluation, the start of the operation was set to 0 sec, and the time until one piece was judged was measured, which was done for five images.

Table 1 shows that there is almost no difference in execution time between manual splitting and using the split function. The fact that there is no performance degradation using this design flow indicates that only the time and effort required to develop a multi-FPGA system can be reduced while maintaining the same performance as the conventional design method.

5. Conclusion

In this paper, we improved the multi-FPGA board design environment for the multi-FPGA system M-KUBOS cluster. We added a partitioning function to CWB, and developed the automatic communication channel configuration table generation tool. By using them, we were able to reduce the time and effort in multi-FPGA development. We evaluated the LeNeT execution time for both manual partitioning and flow-based implementation, and the results show that almost the same performance is obtained. Future work includes further improvement of the splitting function and, in the future, fully automatic splitting of modules without user specification.

Acknowledgments This work was supported by JST, CREST Grant Number JPMJCR19K1, Japan.

References

[1] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher and Valerie Young: Mobile Edge Computing: A key technology towards 5G, ETSI White Paper No.11 (2015).

[2] NEC: CyberWorkBench: Products — NEC, https://www.nec.com/en/global/prod/cwb/index.html (accessed 2023-09-04).

[3] Takumi INAGE and Kazuei HIRONAKA and Kensuke Iizuka and Kohei ITO and Yasuyu FUKUSHIMA and Mitaro NAMIKI and Hideharu AMANO: M-KUBOS/PYNQ Cluster for multi-access edge computing, *The Ninth International Symposium on Computing and Networking (CANDAR 2021)*, pp. 95–101 (2021).

[4] HU, Y. and KOIBUCHI, M.: Optimizing Slot Utilization and Network Topology for Communication Pattern on Circuit-Switched Parallel Computing Systems, *IEICE Transactions on Information and Systems*, Vol. E102.D, No. 2, pp. 247–260 (online), DOI: 10.1587/transinf.2018EDP7225 (2019).