SoC Emulation in Multiple FPGA using Bus Splitter

Wooseung Yang[1], Seung-Jong Lee[2], Ando Ki[2] and Chong-Min Kyung[1]
[1]Dept. of EECS, KAIST,
woosee@duo.kaist.ac.kr, kyung@ee.kaist.ac.kr
[2]R&D Center, Dynalith Systems Co. Ltd.
{sjlee,adki}@dynalith.com

Abstract

This paper proposes an emulation environment for SoC designs using small number of large gate-count FPGA's and a PC system. To overcome the pin limitation problem in partitioning the design when the design size overwhelms the FPGA gate count, we use bus splitter modules that replicate on-chip bus signals in one FPGA to arbitrary number of other FPGA's with minimal pin count.

The proposed scheme is applied to the emulation of 2 million gate multimedia processing chip using two Xilinx Viretex-2 6000 FPGA devices in 6.6MHz operating frequency. An ARM core, memories, camera and LCD display are modeled in software using dual 2GHz Pentium-III processors. This scheme can be utilized for more than 2 FPGA's in the same ways as two FPGA case without losing emulation speed.

1. Introduction

Most current SoC designs are composed of more than one processor units, several memory blocks and large size of logic design blocks. Since the design and verification of the SoC is very difficult, it is a dominant trend to re-use as many pre-verified components as possible [6, 11] and to adopt well-defined on-chip bus specifications [1–4].

In manufacturing point-of-view, the NRE cost of the silicon fabrication becomes higher and higher and the design cycle time becomes shorter and shorter. So full-chip scale emulation of the design or rapid system prototyping is unavoidable to increase the possibility to make the first silicon work and enable hardware-software co-verification [9].

Most of large-scale emulation equipments have lots of FPGA's connected in regular interconnection networks composed of dedicated routing resources or programmable cross-bar switches [7, 10]. These approaches usually have the problem of partitioning the design into small clusters satisfying the computing resource constraints and the interconnection resource constraints. Another weak point is that FPGA's of most recent technology can not be utilized since it takes lots of time and cost to build the whole emulator system composed of many FPGA's and even more complex interconnections.

In the other extreme, one or a few leading-edge FPGA components of over-hundreds of million gates and better operating speed are used to emulate the whole design. This approach is much faster in operating speed by the help of advanced process technology and the simplified interconnection among FPGA's. Also, though using very expensive FPGA components, they are relatively low cost because of reduced complexity in the interconnection logics.

But these approaches usually transfer the burden of partitioning the design into several FPGA's to users, so they require manual partitioning and are not scalable to the number of used FPGA's. This lack in scalability prohibits the application in large size SoC design.

In this paper, we propose a novel scalable partitioning method on the on-chip bus boundary. The remaining parts of this paper composed of the following sections. The section 2 describes the SoC design we are targeting to emulate and surveys possible partitioning strategies. In section 3, we propose a bus-splitting method based on the properties of the on-chip bus standards. The section 4 shows the design of the proposed bus-splitter and its application in the emulation of multimedia processing chip and conclusions are made in the section 5.

2. Emulation target and partitioning strategies

Figure 1 shows a block diagram of typical SoC design we are targeting to emulate which is based on a on-chip bus standard; AMBA AHB bus system [1]. It is a multimedia data processing chip composed of one ARM9 processor core, several AHB IP's related to the MPEG4 video encoding/decoding, SDRAM memory controller and etc.

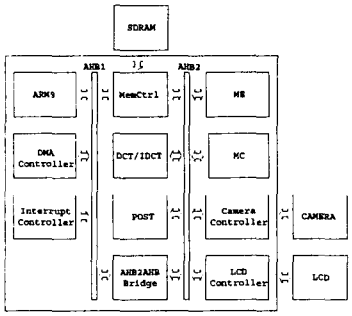

Figure 1. Example of SoC design using on-chip bus standard

To emulate this kind of design, i.e. designs containing a processor core or large memory modules that are not mapped to FPGA devices, most hardware emulation equipments adopt various co-verification methods connecting the hardware part running in FPGA's with an instruction set simulator or a software model running in host computer. Instead of an instruction set simulator, a bus functional model described in test bench description languages such as Vera or TestBuilder may be used because instruction set simulators require C or assembly coding to be completed and the simulation speed is also relatively slow.

Figure 2 shows our test environment utilizing PC system with PCI cards mounting FPGA devices. We translate the bus functional models for the ARM processor described in VERA into C codes that call pre-defined API functions to access FPGA. The transactors in the FPGA's receive the requests generated by API functions and create suitable interface signals for AHB bus, SDRAM controller, Camera controller and LCD controller.

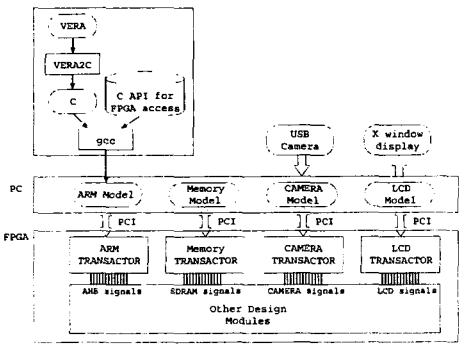

Figure 2. Emulation environment

Other parts of the design are mapped to FPGA's. The whole design cannot be mapped to the biggest FPGA commercially available now.

In the next section, we will focus on partitioning on the bus boundary.

3. Bus splitter

To send and receive bus signals in one FPGA to/from other FPGA's and utilize the limited pin resources efficiently, we designed bus splitter modules. The basic idea of the bus splitter is to utilize the properties of on-chip bus standards.

In the following subsections we will describe the bus splitter design focusing on AMBA AHB bus system. But the principles can be applied to other on-chip bus standards.

3.1 Mastership

The first property is that, at any time instance, there is only one bus master that initiate the bus transaction and one bus slave that respond to the transaction. If we know the mastership of the bus easily, we can save lines that are not used in the current transaction. Figure 3 shows this situation. A design composed of three bus masters($M1, M2, M3$) and three bus slaves($S1, S2, S3$) are partitioned into two partitions($F1 = \{M1, S1, S2\}, F2 = \{M2, M3, S3\}$). For convenience, we assume that arbiter and decoder logics belong to $F1$.

Since $M2$ and $M3$ can not be active at the same time, we only need to allocate one data bus and multiplex the data line from $M2$ and $M3$ inside the FPGA, to transmit data from masters in $F2$ to slaves in $F1$.

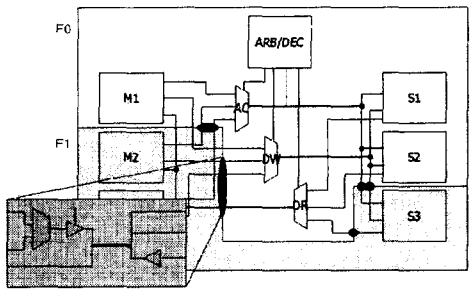

Figure 3. Splitting AMBA AHB bus system

3.2 Transaction Type

The second common property of on-chip bus standard is that read-transaction and write-transaction can not occur at the same time. So the read bus and write bus in on-chip bus can be shared in one physical data bus when interconnecting them off-chip. The tri-state buffers in Figure 3 controls the

direction of data transfer according to the transaction type.

The number of pins required to bisect the system as shown in Figure 3 can be represented as the following equation.

$$\begin{aligned} P_{direct} &= a+ \\ &\quad (b+c+2W)*(M-M_0+1)+ \\ &\quad (d+e+W)*(S-S_0+1) \\ P_{bussplit} &= a+ \\ &\quad b*(M-M_0+1)+ \\ &\quad d*(S-S_0+1) \\ &\quad c+e+2W \end{aligned} \tag{1}$$

where W = bus bit width
M = # of masters in the system
M_0 = # of masters in the FPGA1(containing arbiter/decoder)
S = # of slaves in the system
S_0 = # of slaves in the FPGA1(containing arbiter/decoder)
a = # of signals for global control
b = # of unsharable control signals from master
c = # of sharable control signals from master
d = # of unsharable control signals from slave
e = # of sharable control signals from slave

P_{direct} is the pin count when all the pins are directly connected and $P_{bussplit}$ is the pin count when the bus splitter is used. Parameters, $a \sim e$ depends on the bus standard, and for the case of AMBA AHB, $a = 6$, $b = 2$, $c = 13$, $d = 1$ and $e = 3$. Other parameters depend on the properties of the target design. For the typical system where $W = 32$ and $M - M_0 = S - S_0 = 4$, total pin counts reduction is 82% ($P_{direct} = 581, P_{bussplit} = 101$).

Notice that since parameter b and d is small enough, $P_{bussplit}$ is insensitive to the increase in the number of IP's in the design, whereas P_{direct} is rapidly increased when the design uses more IP's.

3.3 Pipelined Bus

To further reduce the required pin counts, we use time-shared multiplexing technique similar with virtual wire system [5, 8]. When transferring many logical signals with fewer physical lines, it is very important to schedule the sequence of signal transfer according to the evaluation sequence of the combinational logic path. But, since most on-chip bus standards are pipelined so that the bus operations(e.g. address and data phase) are divided into several clock cycles, we don't bother much about the dependency in allocating the bus signals to physical lines.

Figure 4 shows one example of packing 32-bit version of AMBA AHB bus signals into 32-bit data line. The rectangles marked A, D, M and S mean signals from arbiter, decoder, current bus master and current bus slave. In the first micro-cycle of the transfer, information on the current bus master and bus request signals for the next transaction from all master IP's are passed. 32-bit address signals and other control signals for the current transaction are driven by the current bus master in the next two micro-cycles. The response signals for the previous bus transaction are also driven by the slave in the third micro-cycle. In the last micro-cycle, the data is transferred from master to slave (write-transaction) or from slave to master(read-transaction).

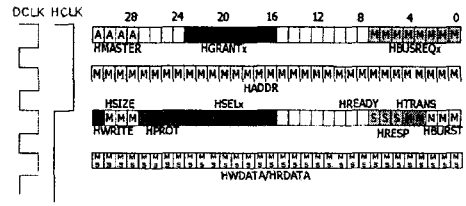

Figure 4. Packing AMBA AHB bus signals into four 32-bit structures

The packing width and depth can be varied according to the limit of the physical data channels connecting FPGA's and the required operating frequency as far as the intra-cycle combinational path dependency is satisfied. The only combinational path spanning two modules in AMBA AHB is $HADDR \rightarrow HSEL$ path.

4. Emulation Example

Figure 5 shows suggested emulation setup for the example design in Figure 1. The ARM processor, two SDRAM modules, camera and LCD display are modeled in software as described in section 2. Other logic parts are mapped into two FPGA devices mounted in PCI cards. The FPGA's are automatically configured by the software in PC whenever the emulation is started. Three major modules in AHB1 bus are mapped in FPGA1 and use 97% of the slice resources in the FPGA. Other modules and transactors to communicate with PC are mapped to FPGA0 and use 70% of the slices.

The bus splitter is composed of two stages as shown in Figure 6 based on the packing structure in Figure 4. The first stage selects signals from currently active bus master and slave. The next stage selectively drives the signal line in time-shared manner only at required micro-cycle according to the packing structure. Signals, $e1$, $e2$, $e3$ and $e4$ indicate micro-cycle $1 \sim 4$.

Figure 7 shows three PCI cards mounting Xilinx Virtex2 6000 FPGA devices, installed in 33MHz PCI slots of Pen-

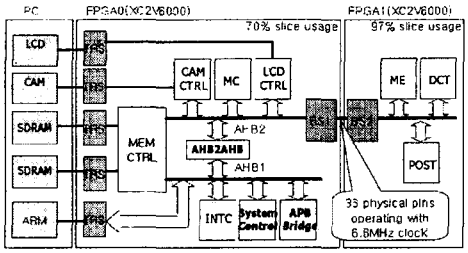

Figure 5. Emulation environment for ARM-based Multimedia SoC

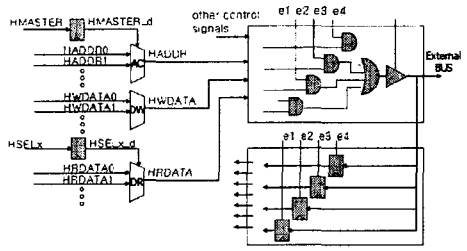

Figure 6. Bus splitter implementation

tium PC. Two of them are used in the emulation of design in Figure 5. 80-pin SCSI cable is used to interconnect the PCI cards. 32 signal lines are used to deliver the packed signals. Additional four signal lines are used to exchange clock and reset signals and other signal lines are used as ground.

In the emulation, MPEG4 coding/decoding test code written in VERA code was used. Actual images captured from the USB camera or stored in the hard disk of the PC were fed to the FPGA and the decoded result was shown in the PC screen. The design mapped in the FPGA's operated at 6.6MHz and the bus splitter modules transferred data using 33MHz PCI clock.

5. Conclusion

In this paper, we proposed an emulation environment for SoC using small number of large gate-count FPGA's and a PC system. To overcome the partitioning problem when the design size overwhelms the FPGA gate count, we used bus splitter modules that replicate on-chip bus signals in one FPGA to arbitrary number of other FPGA's with minimal pin count.

The proposed scheme was applied to the emulation of 2 million gate multimedia processing chip using two Xilinx Viretex2 6000 FPGA devices in 6.6MHz operating frequency. The ARM core, memories, camera and LCD display modules are modeled in software using dual 2GHz

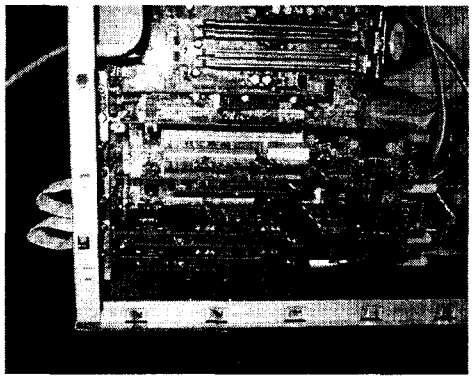

Figure 7. Three PCI cards mounting FPGA's are installed in 33MHz PCI slots of Pentium PC. 80-pin SCSI cable is used to interconnect three cards.

Pentium-III processors.

The proposed scheme can be used for other on-chip bus standards and for more number of FPGA's without losing emulation speed.

References

[1] ARM Limited. *AMBA Specification(Rev 2.0)*, 1999.

[2] M. H. G. M. A. M. Henry Chang, Larry Cooke and L. Todd. *Surviving the SOC Revolution; A Guide to Platform-Based Design.*

[3] IBM Cooperation, Research Triangle Park, NC. *On-Chip Peripheral Bus; Architecture Specifiations*, 1999.

[4] IBM Cooperation, Research Triangle Park, NC. *Processor Local Bus; Architecture Specifiations*, 1999.

[5] M. D. S. Z. H. D. M. H. Jonathan Babb, Russel Tessier and A. Agarwal. Logic emulation with virtual wires. *IEEE Trans. on Computer-Aided Design*, 16(7), July 1997.

[6] M. Keating and P. Bricaud. *Reuse Methodology Manual for System-on-a-Chip Designs*. Kluwer Academic Publishers, Boston, MA, 1999.

[7] M. Khalid. Routing architecture and layout synthesis for multi-fpga systems. 1999.

[8] S.-y. Y. Kyung-soo Oh and S.-I. Chae. Emulator environment based on an fpga prototyping board. *Proceedings of 11th International Workshop on Rapid System Proceedings*, 2000.

[9] P. P. Prakash Rashinkar and L. Singh. *System-on-a-chip Verification; Methodology and Techniques*. Kluwer Academic Publishers, Boston, MA, 2000.

[10] G. B. Scott Hauck and C. Ebeling. Mesh routing topologies for multi-fpga systems. *IEEE Trans. on Very Large Scale Integration Systems*, 6(3):400–408, September 1998.

[11] R. Seepold and A. Kunzmann. *Reuse Techniques for VLSI Design*. Kluwer Academic Publishers, Boston, MA, 1999.