SCATOMi : Scheduling Driven Circuit Partitioning Algorithm for Multiple FPGAs using Time-multiplexed, Off-chip, Multicasting Interconnection Architecture

Young-Su kwon and Chong-Min Kyung
VLSI Systems Lab., CHiPS, KAIST
GuSong-Dong, YuSong-Gu
Taejon, Korea
yskwon@vslab.kaist.ac.kr

Abstract

FPGA-based logic emulator with large gate capacity generally comprises a large number of FPGAs connected in mesh or crossbar topology. However, gate utilization of FPGAs and speed of emulation are limited by the number of signal pins among FPGAs and the interconnection architecture of the logic emulator. The time-multiplexing of interconnection wires is required for multi-FPGA system incorporating several state-of-the-art FPGAs. This paper proposes a circuit partitioning algorithm called SCATOMi(SCheduling driven Algorithm for TOMi) for multi-FPGA system incorporating four to eight FPGAs where FPGAs are interconnected through TOMi(Time-multiplexed, Off-chip, Multicasting interconnection). SCATOMi improves the performance of TOMi architecture by limiting the number of inter-FPGA signal transfers on the critical path and considering the scheduling of inter-FPGA signal transfers. The performance of the partitioning result of SCATOMi is 5.5 times faster than traditional partitioning algorithms. Architecture comparison show that the pin count is reduced to 15.2%-81.3% while the critical path delay is reduced to 46.1%-67.6% compared to traditional architectures.

1 Introduction

In designing system-on-chip(SoC) functional verification represents over 48% of the whole design process [1]. Simulation accelerators and logic emulators are usually based on FPGAs and widely used for functional verification of complex logic designs at the speed 100-1000 times faster than software simulation. An essential limitation of FPGA comes from the fact that its gate capacity lags behind that of the contemporary application-specific integrated circuits(ASIC's) by a factor of five to ten. The multi-FPGA logic emulation system is, therefore, the only viable solution for providing gate-level emulation with a large gate capacity by combining multiple FPGAs through hard-wired interconnections.

A problem with these traditional interconnection architectures is that the logic mapping into FPGA is easily pin-limited. That is, only a fraction of available FPGA gates can be utilized and the circuit partitioning algorithm sometimes fails in routing inter-FPGA nets due to the shortage of IO pins. This is because the number of pins in each FPGA is not increased as fast as the gate count of FPGA is increased.

The traditional partitioning algorithms have mostly concentrated on minimizing the number of cuts (inter-FPGA nets) between logic partitions utilizing the locality of logic designs. The performance driven partitioning algorithms concentrate on reducing the number of inter-FPGA nets on the critical path because the delay of off-chip signal transfer between FPGAs dominates the critical path delay[5][9][10]. In this paper, we propose a novel circuit partitioning algorithm called SCATOMi(SCheduling driven Algorithm for TOMi) for an interconnection architecture called TOMi(Time-multiplexed, Off-chip, Multicasting interconnection) which uses time-multiplexed interconnection wires between FPGAs exploiting multicasting of signals to solve the pin limitation problem.

This paper is organized as follows. In section 2, the TOMi architecture is presented. SCATOMi is described in section 3. Experimental results are shown in section 4.

2 Multi-FPGA Architecture with TOMi

A multi-FPGA system based on TOMi consists of several FPGAs connected via shared inter-FPGA interconnections as shown in Fig. 1. The interconnection among FPGAs consists of wires for "*emulation clock*", "μclk" and

TOMi, respectively. μclk, which is of higher frequency than *emulation clock*, controls micro-operations for the signal transfer between consecutive edges of *emulation clock*. TOMi is composed of time-multiplexed interconnection wires that transfer logic signals from one FPGA to another according to μclk. Each bit line of TOMi shared by all FPGAs transfers a logic signal driven by one of FPGAs in one μclk cycle.

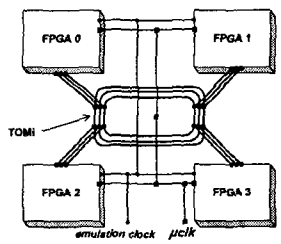

Figure 1. A block diagram of a system comprising four FPGAs interconnected using TOMi.

In Fig. 2, an enlarged view of two FPGAs comprising the TOMi architecture and the operation of TOMi are shown. Each circle indicates an instance of a combinational logic and arrows indicate nets between logics. The direction of a net indicates dependency relation. For example, N_3 can not be transferred until N_0 and N_1 are transferred. The signal transfer scheduling is also shown in the figure. We have assumed that the inter-FPGA signal transfer takes two μclk cycles and the evaluation of combination logics in one FPGA takes one μclk cycle. The bit width(number of bit lines) of TOMi is assumed to be two for simplicity. After N_0 and N_1 are transferred during the time interval $\delta t = (t+2, t+4)$, the combinational logics in FPGA 0 are evaluated during $(t+4, t+5)$. N_3 is transferred during $(t+4, t+5)$ because N_3 is influenced by N_0 and N_1.

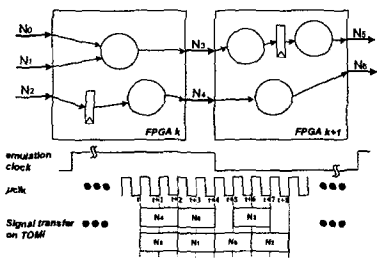

Figure 2. An operation of TOMi architecture.

3 SCATOMi Circuit Partitioning Algorithm for TOMi Architecture

The design is partitioned by min-cut partitioning algorithm at first. In this paper, the improved FM algorithm implemented in [4] is used. After min-cut partitioning, all B-nets are removed to reduce the number of inter-FPGA nets on the critical path. By removing all B-nets, all inter-FPGA nets become F-nets or S-nets. Therefore, in the worst case, the critical path is composed of one S-net and $(n(F)-1)$ F-nets where $n(F)$ is the number of FPGAs. In Fig. 3, the

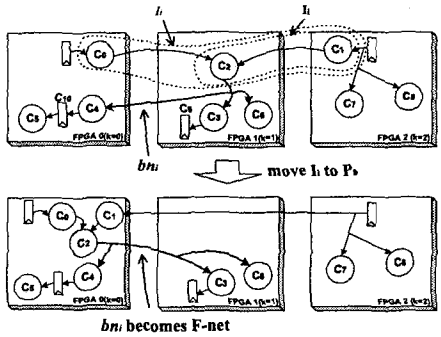

Figure 3. The removal of bn_i by moving $\mathbf{I}_i$. $\mathbf{I}_i$ consists of C_2 and C_1. bn_i becomes F-net by moving $\mathbf{I}_i$ to P_0 while no F-net or I-net becomes B-net.

source instance of bn_i is C_2 and destination instances are C_3, C_4 and C_6. $k_{d,min}$ is 0 because the smallest partition index of all partitions covering C_3, C_4 and C_6 is 0. I_i consists of C_0, C_1 and C_2 while $\mathbf{I}_i$ consists of C_1 and C_2 as their partition indices are larger than $k_{d,min}$(=0). bn_i becomes F-net by moving $\mathbf{I}_i$ to FPGA 0. The second option(D-option) is to move the destination instances of bn_i to the partition with index k_s. A subset of D_i denoted as $\mathbf{D}_i$ is defined such that $\mathbf{D}_i$ includes all the instances located in the partitions with index smaller than k_s. All the instances belonging to $\mathbf{D}_i$ are then moved to the partition k_s to remove B-nets. After the initial partitioning, the number of inter-FPGA signal transfers on the critical path is reduced by considering the signal scheduling result. The balance measure is also considered in the optimization procedure. No B-net is created in this procedure because instances are moved in unit of I_i or D_i. Initially, the dependency graph for inter-FPGA nets is created. In dependency graph, nodes represent the inter-FPGA net and edges represent the precedence relation between inter-FPGA nets. The priority, the distance from the node to the sync node, is computed for each node. The number of resources is W_{TOM_i}, the number of bit lines of

TOMi. The list scheduling algorithm routine denoted as $List\_Scheduling()$ schedules the dependency graph to determine the time interval in which each inter-FPGA signal is transferred. After scheduling, I_i^c and D_i^c for n_i^c(inter-FPGA net on the critical path) are found as shown in Fig. 4. In Fig. 4, the critical path after list scheduling of inter-FPGA net is shown as a bold line. Two inter-FPGA nets, n_0 and n_1 are on the critical path. $\mathbf{I_0^c}(\mathbf{D_0^c})$ is the subset of $I_0^c(D_0^c)$ where instances are in FPGA 0(FPGA 1). In Fig. 4, $\mathbf{I_0^c}$={C_1}, $\mathbf{D_0^c}$={C_2, C_3}, $\mathbf{I_1^c}$={C_2, C_3, C_4} and $\mathbf{D_1^c}$={C_5, C_7, C_8}.

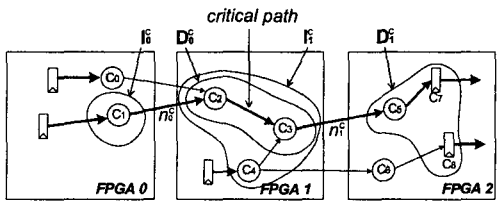

Figure 4. $\mathbf{I^c}$ and $\mathbf{D^c}$.

If $\mathbf{D_0^c}$ is moved to FPGA 2, the number of inter-FPGA signal transfer on the critical path can be decreased by one. Similarly, $\mathbf{I_1^c}$ can be moved to FPGA 0 to reduce the number of inter-FPGA signal transfers on the critical path. No additional B-net is created when $\mathbf{D_0^c}$ or $\mathbf{I_1^c}$ is moved.

4 Experiments

SCATOMi described in the previous section was implemented in C. SCATOMi accepts EDIF netlist and generates partitioned EDIF designs. The Partitioning93 XNF(Xilinx Netlist Format) netlists are selected for the experiment. The EDIF netlists for Partitioning93 circuits were generated from XNF file. The excellence of the proposed partitioning algorithm is shown by comparing the number of inter-FPGA nets and performance results between the proposed partitioning algorithm and other partitioning algorithms. We have applied several traditional partitioning algorithms to partition benchmark circuits on the TOMi architecture. The number of μclk cycles for an inter-FPGA net transfer is two and that for one FPGA evaluation is one. The bit width of TOMi, $W_{TOMi} = 256$. The TOMi architecture comprises four Virtex2 FPGAs. DDR(Double Data Rate) registers in Virtex2 FPGA are used to transfer inter-FPGA nets at high speed where the frequency of μclk is 133MHz(7.5ns period). We compared the results of SCATOMi against those of multi-level KL [7], PFM [4] and hMetis [8]. The numbers of inter-FPGA nets and the performance in the number of μclk cycles per *emulation clock* cycle (denoted as T_e) when the circuits are partitioned into four FPGAs are shown in Table 1.

The μclk count per *emulation clock* cycle of SCATOMi shows a significant reduction compared to others although the number of cut nets is larger than other algorithms. In SCATOMi, all B-nets are completely removed while there are many B-nets in the results of traditional partitioning algorithms. The removal of B-nets reduced the dependencies between partitions and scheduling driven optimization have greatly improved the performance. The operation speed of SCATOMi is 5.5 times faster than other algorithms on the average.

Table 2. The required number of pins for several architectures to implement benchmark circuits.

Circuit	NTT	BORG	VWire	TOMi
s5378	1242	1824	776	676
s9234	1880	2690	868	848
s13207	2536	3748	1174	1024
s15850	3132	4556	1394	1024
s35932	9488	13932	906	1024
s38417	7248	10340	1836	1024
sum	25526	37090	6954	5620
ratio	4.54	6.60	1.23	1.0

The results of implementing benchmark circuits into four FPGAs on several architectures such as NTT mesh [6], BORG partial crossbar [3], VirtualWire(denoted as VWire) [2] and TOMi are shown in Table 2 and Table 3. The number of required pins for TOMi is about 22.0% of NTT, 15.2% of BORG and 81.3% of VirtualWire. The ratio of number of pins to 1.0 for TOMi architecture shows an advantage of TOMi.

Table 3. The critical path delay of benchmark circuits for several interconnection architectures with PFM algorithm and TOMi architecture with SCATOMi algorithm. The unit of measurement is one μclk period, 7.5ns.

Circuit	NTT	BORG	VWire	TOMi
s5378	4.92	11.93	11.50	7.00
s9234	14.16	17.97	20.50	7.00
s13207	18.16	25.34	23.00	11.00
s15850	22.28	28.19	31.00	13.00
s35932	9.95	11.64	14.00	11.00
s38417	13.31	18.02	22.00	7.00
sum	82.78	113.09	122.00	56.0
ratio	1.48	2.01	2.17	1.0

In Table 3, the critical path delays of benchmark circuits

Table 1. A comparison of 4-way partitioning results in terms of the distribution of inter-FPGA nets(# F-nets/# B-nets/# inter-FPGA S-nets) denoted as "cut nets" and the number of μclk cycles per *emulation clock* period(T_e) denoted as "#μ/T_e" for various partitioning algorithms including the proposed algorithm, SCATOMi.

			Multi-KL		PFM		hMetis		SCATOMi	
Circuit	#nets	#inst.	cut nets	#μ/T_e	cut nets	#μ/T_e	cut nets	#μ/T_e	cut nets	#μ/T_e
s5378	3224	3082	207/254/42	24	109/113/44	28	44/65/0	17	142/0/78	7
s9234	6097	5890	441/411/83	37	197/105/3	37	100/36/10	11	272/0/63	7
s13207	9444	8808	510/918/295	70	123/173/60	40	108/38/3	34	311/0/112	11
s15850	11070	10489	966/516/367	82	144/236/83	50	84/49/15	27	148/0/202	13
s35932	19879	18818	1554/1631/272	149	441/344/587	88	32/106/35	8	213/0/86	11
s38417	25588	23982	2151/1391/858	147	274/322/400	69	54/120/16	11	123/0/62	7
sum				509		312		108		56
ratio				9.1		5.6		1.9		1.0

for several interconnection architectures are shown. Although the time-multiplexing feature of TOMi degrades the performance and multicasting increases the capacitive load, the critical path delay of TOMi is 67.6% of NTT mesh, 49.8% for BORG partial crossbar and 46.1% for Virtual-Wire architecture. The TOMi architecture with SCATOMi shows the comparable performance compared to other traditional architectures while it requires small number of FPGA pins using time-multiplexed shared wires.

5 Conclusion

In this paper, we present a novel multi-FPGA partitioning algorithm called SCATOMi for TOMi architecture with the time-multiplexed, off-chip, multicasting interconnection wires. SCATOMi partitions the circuit to minimize the number of off-chip signal transfer and considers the scheduling result of inter-FPGA signals for performance improvement. SCATOMi is suitable for multi-FPGA system with a small number of FPGAs because of capacitive load of shared wires. Salient features of SCATOMi include no pin limitation problem, no routing failure and expansibility. The performance of TOMi architecture with SCATOMi shows a significant performance improvement for benchmark circuits while the required pin number is small.

References

[1] International technology roadmap for semiconductors. Technical report, Silicon Industry Association, 2001.

[2] J. Babb, R. Tessier, M. Dahl, S. Hanono, D. Hoki, and A. Agarwal. Logic emulation with virtual wires. *IEEE Transactions on CAD*, 16(6):609–626, Jun. 1997.

[3] P. Chan and M. Schlag. Architectural tradeoffs in field programmable device based computing systems. In *IEEE Symposium on FPGAs for Custom Computing Machines*, pages 152–161, 1993.

[4] A. Dasdan and C. Aykanat. Two Novel Multiway Circuit Partitioning Algorithms Using Relaxed Locking. *IEEE Transactions on CAD*, 16(2):169–178, Feb. 1997.

[5] W.-J. Fang and A.-H. Wu. Performance-driven multi-FPGA partitioning using functional clustering and replication. In *IEEE/ACM Design Automation Conference*, pages 283–286, 1998.

[6] S. Hauck, G. Borriello, and C. Ebeling. Mesh routing topologies for multi-FPGA systems. *IEEE Transactions on VLSI Systems*, 6(3):400–408, Sep. 1998.

[7] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. Technical report, Sandia National Laboratories, 1993. Technical report SAND93-1301.

[8] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel Hypergraph Partitioning:Applications in VLSI Domain. *IEEE Transactions on CAD*, 16(9):956–964, Sep. 1997.

[9] C. Kim and H. Shin. A performance-driven logic emulation system:FPGA network design and performance-driven partitioning. *IEEE Transactions on CAD*, 15(5):560–568, May. 1996.

[10] P. Sawkar and D. Thomas. Multi-way partitioning for minimum delay for look-up table based FPGAs. In *IEEE/ACM Design Automation Conference*, pages 647–652, 2001.