Fine-Grain Time Division Multiplexing in FPGAs

Rosemary Francis
Computer Lab
University of Cambridge
Rosemary.Francis@cl.cam.ac.uk

Simon Moore
Computer Lab
University of Cambridge
Simon.Moore@cl.cam.ac.uk

Robert Mullins
Computer Lab
University of Cambridge
Robert.Mullins@cl.cam.ac.uk

Our investigation into Networks-on-Chip (NoCs) for Field-Programmable Gate Arrays (FPGAs) indicates that fine-grain time-division multiplexing over configurable wires can reduce chip area and allow the design of effective global interconnect.

I. Introduction

Increase in FPGA capacity combined with CMOS technology scaling has resulted in increased demands on the configurable wiring architecture. Modern FPGAs typically give over 50% of the reconfigurable area over to routing resources. Rent's Rule [1,2] implies that static FPGA wiring is not scalable and we are approaching the point where cross chip communication is impossible within a clock cycle.

In order to maintain clock speeds, communications will need to be pipelined; it is a small step to then send the data over a network. Separately, migrating commonly used structures such as processor cores and digital signal processing modules to hard-blocks is a technique employed by most FPGA manufacturers. This serves not only to cut down on power consumption, but also makes more effective use of the chip area. While hard blocks go a long way to accelerate the performance of mapped systems, the reconfigurable FPGA fabric still lags behind.

The increasing fault rates of new technologies, as well as other issues, are pushing up the cost of ASIC production. The market for FPGAs is only going to expand as it becomes less economically feasible to produce a hardcore custom solution. FPGAs need to adapt to meet this demand while maintaining the levels of abstraction needed for platform independence and ease of programming.

II. Network-on-Chip

A packet-switched mesh NoC [3] is a highly adaptive communication infrastructure that can handle both bursty and stream-based traffic. Low latency routers can be implemented efficiently while still achieving high performance [4]. While NoCs can be effective ASIC interconnect solutions, FPGAs place different demands on a scalable routing solution. For example, FPGA hard blocks run at speeds of around 400MHz, while modern soft-core IP blocks rarely achieve half that [5] and distribute the data bits over a wider area.

Some attempts [6] have been made to implement soft-core networks on FPGAs. While these maintain the flexibility of reconfigurable designs, the reconfigurable resources needed to implement the network cannot be justified. Even the simplest routers are comparable in size to many of the soft-core modules they endeavour to serve. Simpler approaches have focused on an alternative solution using more wires and less logic [7], but they are still extremely resource hungry and achieve very low data rates. We have been looking into a new architecture which allows the design of a flexible soft core NoC that is neither wiring nor logic hungry.

III. Time-Division Multiplexed Wiring

Time Division Multiplexing (TDM) has been used to improve resource usage on FPGAs in areas of research such as multi-context FPGAs [8], but we aim to multiplex over individual wire segments within the FPGA. We have developed a scheme whereby wires are used by multiple signals within a design clock cycle. Previously static paths are pipelined to achieve high data rates using fewer wires. A design clock cycle is divided into many time slots. Signals are scheduled onto shared wires by the allocation of time slots. The data is latched on each shared wired and the scheduling is performed at compile time.

To reduce the amount of wiring on the FPGA and increase the density of logic, we have been working on sharing the wires effectively without requiring very many time slots or significantly extending the critical path.

In order to evaluate the potential of this shared wiring we have written a scheduler to map designs from modern

TABLE I
TABLE SHOWING THE REDUCTION IN THE NUMBER OF WIRES REQUIRED PER CHANNEL BEFORE AND AFTER SCHEDULING ONTO TDM WIRING

Benchmarks	Num of LUTs	Max wires before	Max Wires after	Reduction
oc_wb_dma	3479	66	45	32%
oc_vga_lcd	2207	47	36	23%
oc_pci	2439	57	42	26%
oc_mem_ctrl	3972	49	35	29%
oc_des_perf_opt	5336	46	33	28%

conventional FPGAs onto new shared wiring FPGAs. Conflicts arise when more than one signal requires the same wire in a given time slot. In this case one of the wires needs to be moved onto non-shared wiring or buffered on previous wires. The scheduler makes these decisions in order to minimise the amount of wiring and number of time slots required.

We will also use the higher data rate enabled by the TDM architecture to design an effective global NoC infrastructure. Much of the arbitration can be performed at compile time thus saving on logic block utilisation. Network routers can be placed to make full use of the wire sharing capability. We will be able to implement a global interconnect network with the flexibility of a soft-core interconnect network, but the size and speed of a hard-core network.

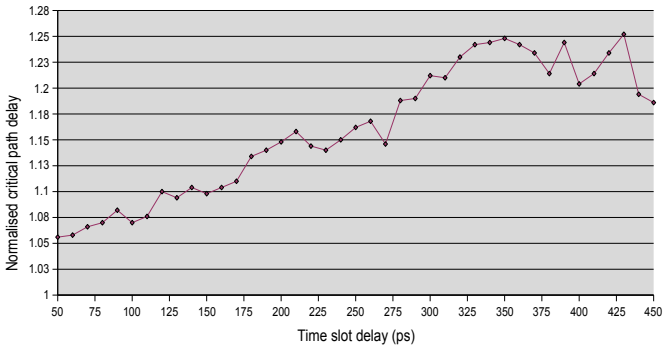

Fig. 1. Increase in critical path delay as the time slot length varies

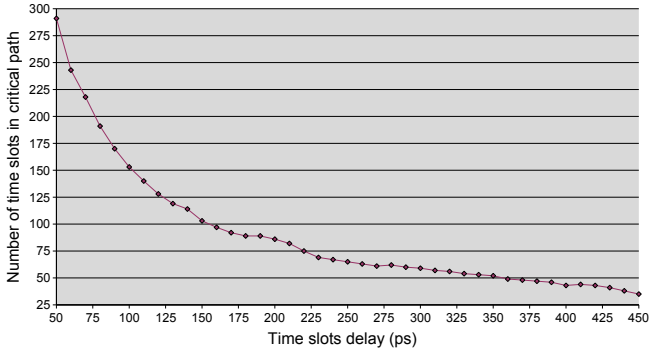

Fig. 2. Number of time slots required to schedule the design as the time slot length varies

IV. RESULTS

We selected the largest designs from the Altera university program to use as benchmarks and subjected them to a series of experiments after placing and routing them onto a Stratix FPGA. Ideally we would have rewritten the entire CAD chain to optimise the benchmarks for TDM wiring, but we have managed to achieve good results without moving the location of any of the wiring.

We have investigated how far we can reduce the amount of wiring on the FPGA in the first stage of our scheduling algorithm and what this does to the critical path. The table above shows the number of wires in the most utilised channel before and after wire sharing is applied.

The first set of experiments used a naive scheduling algorithm, which assumes infinite resources. We varied the time-slot length (the length of an interconnect clock cycle) and increased the number of time slots until it was possible to schedule all the wires. From this we know the minimum critical path penalty and the natural limit of wire sharing before any conflicts have been resolved. We have discovered that with fractional latency penalties, the wiring naturally schedules onto two thirds the wiring required for non-shared designs. This was uniform across different time-slot lengths.

Graphs 1 and 2 show how the number of slots required and the average timing penalty vary with the length of a time slot. The timing penalty is caused by constraining data to wait for the next clock edge to pass a wire and thus lengthening the critical path delay. Discontinuities in the timing penalty graph are caused by mismatches between the time-slot length and on-chip wire delays. A time slot that is much more than one wire delay, but less that two is inefficient. Clearly we can greatly reduce the number of time slots required without significantly affecting the timing. It is critical to keep the number of time slots down in order to minimise the silicon cost of extra configuration bits and high speed clock networks.

Following these results we have enabled subset allocation to measure the scheduling penalty. Subset allocation restricts connections within a switch box. Wires may only connect to others in the same subset. We reran the experiments with enough wiring, but the wiring indexed so that only those with the same index can connect to each other within the switch box. We have achieved good subset allocation with a fast algorithm resulting in few

scheduling conflicts. This means that TDM wiring can cope with the reduced population switch boxes key to implementing dense FPGA interconnect.

Requiring designs to map to fine grain wiring subsets and fewer wires causes timing conflicts. Having determined the optimum time-slot length for scheduling we will be using the scheduler to solve these timing conflicts and push the amount of wire sharing to its limit. As conflict resolution causes the critical path length to increase the length of a time slot will be critical to ensuring that the number of time slots remains reasonable.

We will be presenting these results in detail along with the effects of pushing the scheduling to the limit. We will be looking at the effect of using just one wire on each routing channel in the FPGA .

V. Conclusion

As static wiring fails to scale, TDM wiring presents an interesting dynamic alternative. We have demonstrated that dynamic routing can be used to reduce the amount of FPGA configurable wiring by around 30%. This will allow us to increase the density of the lookup tables as well as use the TDM wiring to share them. We believe that the TDM wiring can be used to harness the flexibility of soft core interconnect and combine it with the performance of hard-core NoCs.

VI. Further Work

This work is an introduction to more detail results to be presented in the Network-on-Chip Symposium 2008. In the full paper we investigate a wider range of benchmarks and use a scheduling algorithm to exploit flexability within the TDM wiring. While this leads to an increase in the latency of the critical paths by a factor of about four, it does allow us to decrease the ammount of wiring by a factor of five whilst maintaining fewer than 24 configuration bits per routing buffer.

We are currently working on a power and area model in order to measure the real benefits of using TDM wiring.

References

[1] B. S. Landman and R. L. Russo, On a Pin Versus Block Relationship For Partitions of Logic Graphs, IEEE Trans. on Comput., 1971.

[2] P. Christie and D. Stroobandt, The Interpretation and Application of Rent's Rule, IEEE Trans. on VLSI Systems, 2000.

[3] W. Dally and B. Towles, Route Packets, Not Wires: On-Chip Interconnection Networks, DAC 2001

[4] R. Mullins, A.West and S. Moore, The Design and Implementation of a Low-Latency On-Chip Network, ASPDAC 2006.

[5] R. Goering, FPGA users rank challenges, tasks, EETimes, 2006

[6] T. Marescaux, A. Bartic, D, Verkest, S. Vernald and R. Lauwereins, Interconnection Networks Enable Fine-Grain Dynamic Multi-Tasking on FPGAs, FPL 2002

[7] M. Saldana, L, Shannon, J, Shuo Yue, S, Bian, J, Craig, and P, Chow. Routability Prediction of Network Topologies in FPGAs. IEEE Transactions on VLSI Systems, August 2007.

[8] S Trimberger, D Carberry, A Johnson, J Wong, A Time-Multiplexed FPGA, IEEE Symposium on FPGAs for Custom Computing Machines, 1997