A Novel and Efficient Routing Architecture for Multi-FPGA Systems

Mohammed A. S. Khalid
Quickturn Design Systems, Inc.
55 West Trimble Road
San Jose, CA 95131-1013
mkhalid@quickturn.com

Jonathan Rose
Dept. of Electrical and Computer Engineering
University of Toronto, Toronto
Ontario, CANADA M5S 3G4
jayar@eecg.toronto.edu

Abstract

Multi-FPGA systems (MFSs) are used as custom computing machines, logic emulators and rapid prototyping vehicles. A key aspect of these systems is their programmable routing architecture which is the manner in which wires, FPGAs and Field-Programmable Interconnect Devices (FPIDs) are connected. Several routing architectures for MFSs have been proposed [Arno92] [Butt92] [Hauc94] [Apti96] [Vuil96] [Babb97] and previous research has shown that the partial crossbar is one of the best existing architectures [Kim96] [Khal97]. In this paper we propose a new routing architecture, called the **H**ybrid **C**omplete-**G**raph and **P**artial-Crossbar (HCGP) which has superior speed and cost compared to a partial crossbar. The new architecture uses both hard-wired and programmable connections between the FPGAs. We compare the performance and cost of the HCGP and partial crossbar architectures experimentally, by mapping a set of 15 large benchmark circuits into each architecture. A customized set of partitioning and inter-chip routing tools were developed, with particular attention paid to architecture-appropriate inter-chip routing algorithms. We show that the cost of the partial crossbar (as measured by the number of pins on all FPGAs and FPIDs required to fit a design), is on average 20% more than the new HCGP architecture and as much as 25% more. Furthermore, the critical path delay for designs implemented on the partial crossbar were on average 20% more than the HCGP architecture and up to 43% more. Using our experimental approach, we also explore a key architecture parameter associated with the HCGP architecture: the proportion of hard-wired connections versus programmable connections, to determine its best value.

1 Introduction

Field-Programmable Gate Arrays (FPGAs) are widely used for implementing digital circuits because they offer moderately high levels of integration and rapid turnaround time [Brow92]. Multi-FPGA systems (MFSs), which are collections of FPGAs and memory joined by programmable connections as illustrated in Figure 1, are used when the logic capacity of a single FPGA is insufficient, and when a quickly re-programmed system is desired. The typical uses are for logic emulation [Apti96] [Quic96] [Babb97], rapid prototyping [Van92] [Gall94] [Alte94] [Lewi97] and reconfigurable custom computing machines [Arno92] [Cass93] [Dray95] [Lewi97] [Vuil96] [Lewi97].

The routing architecture of an MFS is the way in which the FPGAs, fixed wires, and program-

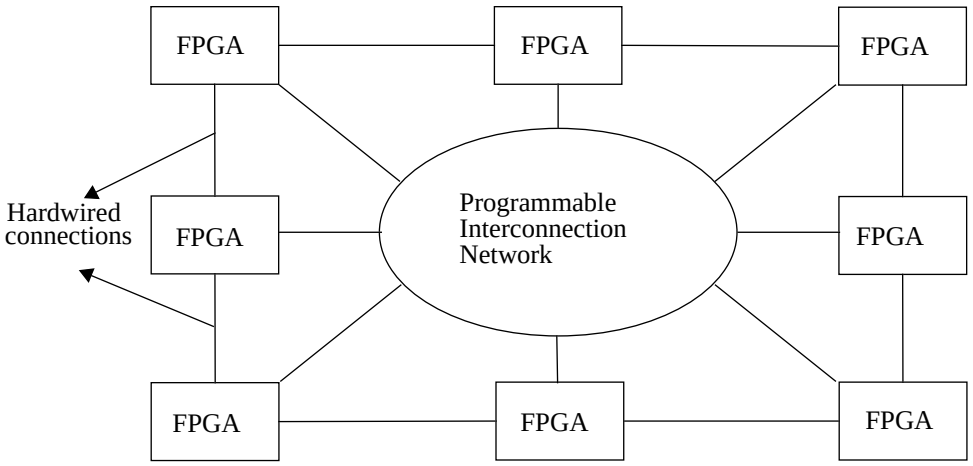

Figure 1 - A Generic Multi-FPGA System

mable interconnect chips are connected. The routing architecture has a strong effect on the speed, cost and routability of the system. Many architectures have been proposed and built [FCCM] [Butt92] [Van92] [Apti96] [Babb97] [Lewi97] and some research work has been done to empirically evaluate and compare different architectures [Kim96] [Khal97]. These studies have shown that the partial crossbar is one of the best existing MFS architectures. In this paper we present a new routing architecture for MFSs that uses both hardwired and programmable connections to reduce cost and increase speed. We evaluate and compare the HCGP architecture and the partial crossbar architecture using an empirical approach. In particular we compare architectures on the basis of pin cost and speed.

The speed comparisons are based on post inter-chip routing critical path delay of real benchmark circuits, which, to our knowledge, is the first time such detailed timing information has been used in the study of board-level MFS architectures.

We focus on single-board MFS routing architectures that use no more than about 25 FPGAs. This is for two reasons: First, the complete graph topology used in the HCGP architecture does not scale well for a large number of FPGAs. It becomes infeasible to connect each FPGA to every other FPGA because such a scheme would likely result in severe routability problems. In such cases, hierarchical architectures would be more effective. We believe that the HCGP architecture could form the basis of a hierarchical architecture, with the root architecture being an HCGP, and groups of HCGPs connecting in a next-level HCGP and so on. Second, we did not have huge circuits and the CAD tools required for mapping such circuits to hierarchical architectures.

Previous work has been done evaluating mesh [Hauc94] and other architectures [Chan93]. In [Hauc94], several constructs (1-hop interconnections, Superpins, and Permutations) were proposed to improve the basic 4-way mesh. However, synthetic netlists (not real circuits) were used to evaluate different mesh topologies. In [Chan93] architectural trade-offs in the design of folded Clos network (partial crossbar) were investigated and an optimal algorithm for routing two-terminal nets was presented. Although this work provides some theoretical insight into these architectures, empirical studies that evaluate the implementation of real circuits on different architectures provide a more clear picture of the ‘goodness’ of each architecture relative to the others [Kim96] [Khal97]. Our own previous research has shown that partial crossbar is vastly superior to the best mesh architecture [Khal97]. In [Kim96], several MCNC circuits were mapped to seven different architectures, including the partial crossbar architecture. Each circuit was mapped to a fixed size

MFS (containing 30 FPGAs). The size of the FPGA was varied depending upon the circuit size. Each architecture was evaluated on the basis of total number of CLBs needed across all circuits (where fewer CLBs used implies better architecture), the type of FPGA chips used (smallest FPGAs implies better architecture), and maximum number of hops needed across all inter-FPGA nets (as a metric for speed). A *hop* is defined as a chip-to-chip connection, i.e. a wire segment that connects two different chips on a board. It was shown that one of the proposed architectures, FPGAs connected together as a tri-partite graph, gave the best results (slightly better than partial crossbar). In this work, relatively few large circuits were used that would have really 'stressed' the architectures, as only three reasonably large circuits (>2000 CLBs) were employed. Also, for the speed estimate only the worst case *net* delay in terms of the number of hops was considered; which is not as representative of the true delay as post-routing *critical path* delay.

An early version of the present work appeared in [Khal98]. The present work includes key enhancements, particularly timing-driven inter-chip routing for HCGP and an exploration of the effects of a key parameter P_p (to be defined later) on the speed of the HCGP architecture. This paper is organized as follows: In Section 2 we describe the experimental evaluation procedure and the evaluation metrics used, and give details on the suite of large benchmark circuits used in this experimental work. In Section 3 we cover the architectural issues and assumptions that arise when mapping real circuits to the HCGP and partial crossbar architectures. We also briefly describe architecture-specific inter-chip routing algorithms for these architectures. Experimental results and their analysis is presented in Section 4, and we conclude in Section 5.

2 Experimental Overview

To evaluate the two routing architectures considered in this paper, we used the experimental procedure illustrated in Figure 2. Each benchmark circuit was partitioned, placed and routed into each architecture. Section 2.1 describes the general toolset used in this flow. The cost and delay metrics that we use to evaluate architectures are described in Section 2.2. A description of the 15 benchmark circuits used is given in Section 2.3.

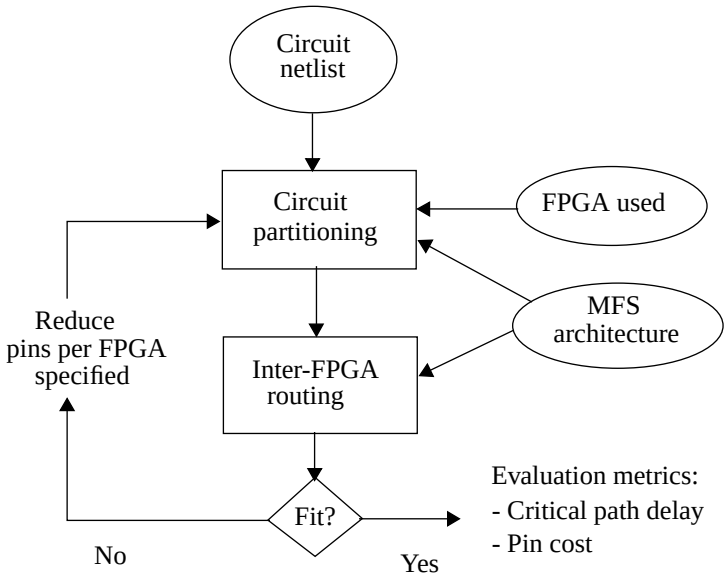

Figure 2 - Experimental Evaluation Procedure for Multi-FPGA Systems

2.1 General CAD Flow

As illustrated in Figure 2, we start with a (technology mapped) netlist of 4-LUTs and flip flops of the circuit. The circuit is partitioned into a minimum number of sub-circuits using a multi-way partitioning tool which accepts as constraints the specific FPGA logic capacity and pin count. For all the experiments presented in this paper we used a Xilinx 4013E-1 FPGA, which consists of 1152 4-LUTs, 1152 flip flops, and 192 usable I/O pins [Xili97]. Multi-way partitioning is accomplished using a recursive bi-partitioning procedure. The partitioning tool used is called 'part' and was originally developed for the Transmogrifier-1 rapid prototyping system [Gall94]. It is based on the Fiduccia and Mattheyses partitioning algorithm [Fidu82] with an extension for timing-driven pre-clustering [Shih92]. The output of the partitioning step is a netlist of connections between the FPGAs that contain the circuit.

Given the chip-level interconnection netlist, the next step is to route each inter-FPGA net using the most suitable routing path. The routing path chosen should be the shortest path (use the minimum number of hops) and it should cause the least possible congestion for subsequent nets to be routed. Depending on the architecture, the routing resources available in an MFS could be wires that are direct connections between FPGAs, or wires that connect FPGAs and FPIDs.

If the routing attempt fails, the partitioning step is repeated after reducing the number of I/O pins per FPGA specified to the partitioner. This usually increases the number of FPGAs needed, and helps routability by decreasing the pin demand from each FPGA, and providing more "route-through" pins in the FPGAs which facilitate routing.

Note that in an actual MFS, the inter-FPGA routing step is followed by pin assignment, placement and routing within individual FPGAs. We need not perform these tasks because we are only interested in knowing the MFS size needed to fit the circuit. Our previous research has shown that we can afford to assign pins randomly for each FPGA without jeopardizing routability and speed [Khal95]. During recursive bi-partitioning, we restrict the logic utilization of each FPGA to be at most 70% to avoid placement and routability problems within individual FPGAs. Thus we ensure that if an inter-FPGA routing attempt succeeds, it is almost guaranteed that the subsequent pin assignment, placement, and routing steps will be successful for each FPGA in the MFS.

Notice that the above-mentioned claims about I/O pin-constrained placement and routing are not applicable to the older Xilinx FPGAs (XC3000) and the older Xilinx tool set (the APR tool set, [Xili92]). However, in our research we assume that the Xilinx XC4013 FPGA and the XACT tool set [Xili94] is used, which give excellent results under I/O pin-constrained placement and routing, as shown in [Khal95]. Therefore, our assumption that pin locking on a Xilinx XC4013 FPGA will not have an unduly adverse impact on its routability and speed is valid.

Another important point is that the 70% cap on FPGA logic utilization (that we imposed), could be increased further (whenever possible for specific FPGAs) if we perform placement and routing for individual FPGAs. We did not perform individual FPGA placement and routing because it would have involved a huge amount of experimental effort and time, and probably would not change our architectural conclusions in any significant way. Also, in most cases, very high FPGA logic utilization (say > 85%) after partitioning is rare because of FPGA pin limitations. In fact, the average post-partitioning logic utilization is less than 50%. The 70% cap on logic utilization is based on a conservative estimate. From our previous research study [Khal95] and from anecdotal evidence provided by other FPGA users, we found that in almost all circuits, restricting logic utilization to 70% or less leads to routing completion in the Xilinx XC4000 series of FPGAs.

We have developed a specific router for each of the architectures compared. (We had attempted to create a generic router but found that it had major problems with different aspects of each architecture [Khal99].)

2.2 Evaluation Metrics

To compare the two routing architectures we implement benchmark circuits on each and contrast the pin cost and post-routing critical path delay, as described below.

2.2.1 Pin Cost

The cost of an MFS is likely a direct function of the number of FPGAs and FPIDs: If the routing architecture is inefficient, it will require more FPGAs and FPIDs to implement the same amount of logic as a more efficient MFS. While it is difficult to calculate the price of specific FPIDs and FPGAs, we assume that the total cost is proportional to the total number of pins on all of these devices. Since the exact number of FPGAs and FPIDs varies for each circuit implementation (in our procedure above, we allow the MFS to grow until routing is successful), we calculate, for each architecture, the total number of pins required to implement each circuit. We refer to this as the *pin cost* metric for the architecture.

2.2.2 Post Routing Critical Path Delay

The speed of an MFS, for a given circuit, is determined by the critical path delay obtained after a circuit has been placed and routed at the inter-chip level. We call this the *post-routing critical path delay.* We have developed an MFS static timing analysis tool (MTA) for calculating the post routing critical path delay for a given circuit and MFS architecture.

The operation and modeling used in the MTA are described briefly as follows: It first calculates the critical path delay of the un-partitioned design using a widely used method called the *block oriented technique* [Joup87]. It then reads the inter-FPGA netlist and the routing path for each inter-FPGA net, as provided by the inter-chip router, and the MFS architecture description. From this information the circuit is annotated with the inter-chip delays, from which the post-routing critical path delay can be calculated.

In the delay annotation step, the delay values given in Table 1 (obtained from data sheets [Xili97] and [ICub97] and some design experience) are used.

Item	Delay (ns)
Intra-FPGA CLB-to-CLB routing delay	2.5
FPGA input pad delay	1.4
FPGA output pad delay	3.2
CLB delay (without using H-LUT)	1.3
CLB delay (via H-LUT)	2.2
FPID crossing delay (including pad delays)	10
PCB trace delay	3
FPGA route through delay	10

Table 1: Delays Used in Timing Analyzer Model

Note that since we do not perform individual FPGA place and route, we approximate the CLB-to-CLB delay as a constant. The value of 2.5 ns for CLB-to-CLB routing delay is roughly half the delay on a long line for XC4013E-1 FPGA. This is a pessimistic estimate. Although using a single delay value is somewhat inaccurate, it still gives us a good estimate of the post-routing critical path delay of an MFS because it is dominated by off-chip delay values.

Circuit	Size	Function	Source, Synthesis tool used (if applicable)
s35932	4374 LUTs, 1728 FFs, 357 I/Os	Sequential circuit	MCNC
s38417	6097 LUTs, 1463 FFs, 134 I/Os	Sequential circuit	MCNC
s38584	4396 LUTs 1451 FFs, 292 I/Os	Sequential circuit	MCNC
mips64	2900 LUTs 440 FFs, 260 I/Os	Scaled down version of MIPS R4000	PREP, Verilog model synthesized using Exemplar
spla	3423 LUTs 0 FFs, 62 I/Os	Combinational Circuit	MCNC
cspla	2039 LUTs 0 FFs, 62 I/Os	Clone of spla	UofT, Generated using GEN[Hutt96]
mac64	2560 LUTs 64 FFs, 133 I/Os	64-bit multiply-accumulate ckt.	UofT, Verilog model synthesized using Synopsys
sort8	1540 LUTs 200 FFs, 20 I/Os	8-bit HW sort engine	UofT, Verilog model synthesized using Synopsys
fir16	5366 LUTs 1040 FFs, 60 I/Os	16-bit, 8-stage FIR filter	UofT, Verilog model synthesized using Synopsys
gra	2494 LUTs 1156 FFs, 144 I/Os	Graphics acceleration circuit	UofT, circuit generated using tmcc[Gall95]
fpsdes	3484 LUTs 1008 FFs, 69 I/Os	Fastest pseudo DES circuit	UofT, Verilog model synthesized using Synopsys
spsdes	2452 LUTs 982 FFs, 69 I/Os	Smallest pseudo DES circuit	UofT, Verilog model synthesized using Synopsys
ochip64	3617 LUTs 5810 FFs, 84 I/Os	Output chip for ATM switching chip set	UofT, VHDL model synthesized using Exemplar
ralu32	2553 LUTs 584 FFs, 98 I/Os	32-bit register file, ALU, and control logic	PREP, VHDL model synthesized using Synopsys
iir16	3149 LUTs 522 FFs, 52 I/Os	16-bit IIR filter	UofT, VHDL model synthesized using Synopsys

Table 2: Benchmark Circuits

2.3 Benchmark Circuits

A total of fifteen large benchmark circuits were used in our experimental work. An extensive effort was expended to collect this suite of large benchmark circuits. The details of each benchmark circuit are shown in Table 2 which provides the circuit name, size (in 4-LUTs, D flip flops, and I/O count), rough description of the functionality, the source of the circuit and the manner in

which it was synthesized. Four circuits were obtained from MCNC [Yang91], two from FPGA synthesis benchmarks [Prep96], and the remaining nine were developed at the University of Toronto (UofT). The circuits from MCNC were available in the XNF [Xili97] gate-level netlist format required by our front end tools. All the circuits from [Prep96] and UofT were originally available as VHDL or Verilog HDL models and were synthesized into XNF netlists using Exemplar [Exem94] and Synopsys Behavioral Compiler [Knap96] and/or Design Compiler [Syno97] synthesis tools. We show these details of the benchmark circuits because we feel that the MCNC circuits that have been used so far in MFS architecture studies are insufficient in terms of size and variety to 'stress' different architectures and the mapping tools used. Specifically, we found that they are easier to partition and map compared to the other real circuits that we use in this work.

3 Routing Architecture Description and Routing Algorithms

In this Section we describe the partial crossbar and HCGP architectures. For each architecture, we briefly describe an architecture-specific inter-chip router.

3.1 Architectural Description and Routing for the Partial Crossbar

The partial crossbar architecture [Butt91] [Butt92] [Varg93] is used in logic emulators produced by Quickturn Design Systems [Quic96]. A partial crossbar using four FPGAs and three FPIDs is shown in Figure 3. The pins in each FPGA are divided into N subsets, where N is the number of FPIDs in the architecture. All the pins belonging to the same subset number in different FPGAs are connected to a single FPID. Note that any circuit I/Os will have to go through FPIDs to reach FPGA pins. For this purpose, a certain number of pins per FPID (50) are reserved for circuit I/Os. Notice that we could have reserved the number of pins per FPID required for I/O signals based on circuit requirements. For this scheme, the number of reserved pins per FPID (for I/O signals) would be variable across different circuits. We did not use this scheme because it will not make any significant difference in architectural comparison results. Also, our present scheme is easier to implement. As for the number of pins per FPID (50) reserved for circuit I/O signals, we used this number to meet the maximum I/O requirement among the circuits in our benchmark suite.

The number of pins per subset (P_t) is a key architectural parameter that determines the number of FPIDs needed and the pin count of each FPID. The extremes of the partial crossbar architecture can be illustrated by considering a system with four FPGAs, and assuming 192 usable I/O pins per FPGA: a P_t value of 192 will require a single 768-pin FPID that acts as a full crossbar. A P_t value of 1 will require 192 4-pin FPIDs. Both of these cases are impractical.

A good value of P_t should require low cost, low pin count FPIDs. For the above example, a P_t value of 12 will require 16 48-pin FPIDs. When we consider FPID pins required for circuit I/Os we will need to use 64 or 96-pin FPIDs that are commercially available [ICub97]. When choosing a value of P_t, we must ensure that number of usable I/Os per FPGA is evenly divisible by P_t or at least the remainder should be a very small number so that we can use such pins for routing high fanout inter-FPGA nets. In this work we set $P_t = 17$ which leaves five pins per FPGA to be used as global lines in the partial crossbar architecture. These global lines are used for routing global nets like *reset, clock* and other very high fanout nets in the circuit. Our previous research [Khal97] has shown that, for real circuits, the routability and speed of the partial crossbar is not affected by the value of P_t used. But this is contingent upon using an intelligent inter-chip router that understands the architecture and routes each inter-FPGA net using only two hops to minimize the routing delay. However, a practical constraint is that we should avoid using P_t values that require expensive or even unavailable high pin count FPIDs.

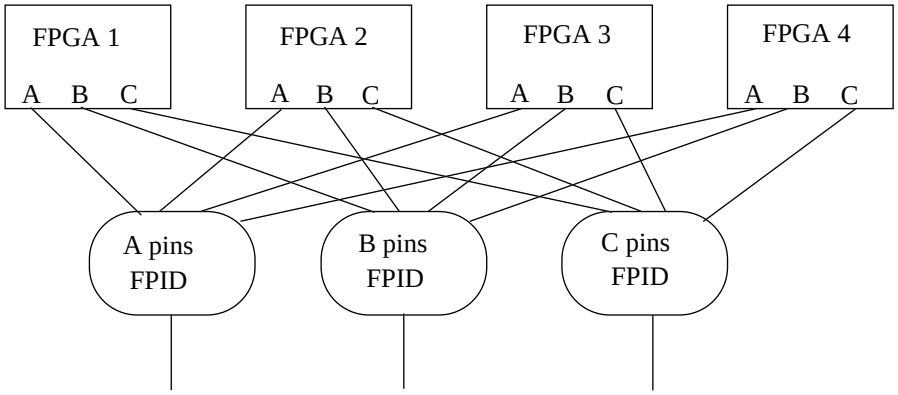

Figure 3 - The Partial Crossbar Architecture

3.1.1 Routing Algorithm for the Partial Crossbar

For any MFS architecture in general and for the partial crossbar in particular, it is important to use a routing algorithm that exploits architecture-specific features in order to obtain good results.

We have developed a routing tool, PCROUTE, for the partial crossbar architecture that gives excellent routability and speed results for all of our benchmark circuits. Irrespective of the value of P_t, it achieves 100% routing completion and produces two-hop routing for all the nets in almost all circuits. For only two circuits, for the specific case of $P_t = 4$, it produced multi-hop routing paths for a negligible number of nets (1 out of 991 nets for the first circuit and 3 out of 645 nets for the second). In practical terms, this means it gives almost optimal results for all of our benchmark circuits.

The PCROUTE algorithm works as follows: for each net (irrespective of fanout), it evaluates potential routing paths through all available FPIDs. It uses a suitable cost function to choose an FPID that will guarantee balanced usage of FPIDs and will preserve the most options for two-hop routing of subsequent nets to be routed. Consider a partial crossbar that consists of *n* FPGAs and *m* FPIDs. Consider an *N-terminal* net called *M*. Let *F* denote the set of FPGAs belonging to *M*, i.e. $\{f_1, f_2, ..., f_N\}$.

Let A_{ik} denote the number of available wires between FPGA *i* and FPID *k*. The routing cost of the net *M* through FPID *k*, *C*(*M, k*), is given by:

$$C(M, k) = \sum_{i = f_1}^{f_N} \frac{P_t}{A_{ik}}$$

An FPID that has the lowest routing cost for the net *M* is chosen for routing that net.

We show in [Khal99] that PCROUTE is equivalent in quality to other partial crossbar routers that have been proposed so far [Kim96] [Mak97a] [Lin97]. PCROUTE is better than [Mak97b] in terms of both speed and routability because that algorithm splits each multi-terminal into a set of two-terminal nets and routes them independently, leading to multiple hops and even possible routing failures.

3.2 Architectural Description and Routing for HCGP

The HCGP architecture for four FPGAs and three FPIDs is illustrated in Figure 4. The I/O

pins in each FPGA are divided into two groups: hardwired connections and programmable connections. The pins in the first group connect to other FPGAs and the pins in the second group connect to FPIDs. The FPGAs are directly connected to each other using a complete graph topology, i.e. each FPGA is connected to every other FPGA. The connections between FPGAs are evenly distributed, i.e. the number of wires between every pair of FPGAs is the same. The FPGAs and FPIDs are connected in exactly the same manner as in a partial crossbar. As in the partial crossbar, any circuit I/Os will have to go through FPIDs to reach FPGA pins. For this purpose, a certain number of pins per FPID (50) are reserved for circuit I/Os.

The direct connections between FPGAs can be exploited to obtain reduced cost and better speed. For example, consider a net that connects FPGA 1 to FPGA 3 in Figure 4. If there were no direct connections as in the partial crossbar, we would have used an FPID to connect the two FPGAs. This will cost extra delay and two extra FPID pins. A natural question to ask is: why not dispense with FPIDs and just use FPGAs connected as a completely connected graph as investigated in [Kim96]? The answer is that routing multi-terminal nets in an FPGA-only architecture is expensive in terms of routability because in such an architecture a multi-terminal net requires many extra pins in the source FPGA, as illustrated in Figure 5(a). In Figure 5(a) two extra FPGA pins are used for routing a fanout 3 multi-terminal net. Since extra pins are scarce on an FPGA this has an adverse effect on the routability of FPGA-only architectures. On the other hand, if we use an FPID for routing the same multi-terminal net, we do not need even a single extra FPGA pin, other than the FPGA pins needed to access the source and sinks of the net as shown in Figure 5(b).

A key architectural parameter in the HCGP architecture is the percentage of programmable connections, P_p. It is defined as the percentage of each FPGA's pins that are connected to FPIDs (the remainder are connected to other FPGAs). If P_p is too high it will lead to increased pin cost, if it is too low it will adversely affect routability. If P_p is 0% the HCGP architecture degrades to a completely connected graph of FPGAs with no FPIDs used. If P_p is 100% the HCGP architecture degrades to a standard partial crossbar. A key issue we address later is the best value of P_p for obtaining minimum cost and good routability.

3.2.1 Routing Algorithm for HCGP

The inter-chip routing algorithm for HCGP is similar to the partial crossbar routing algorithm

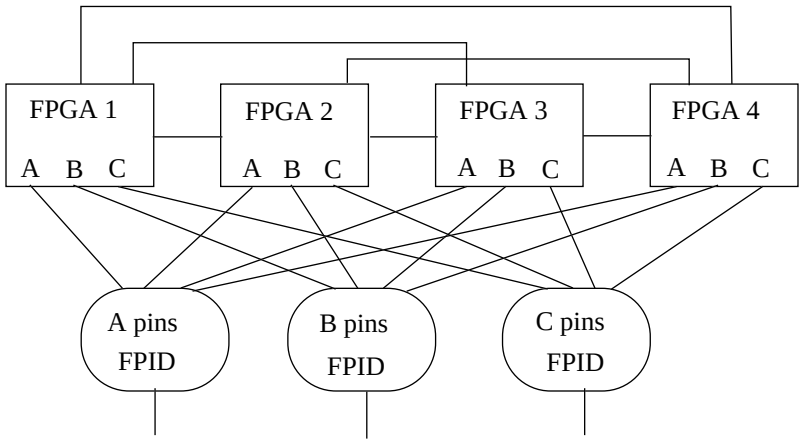

Figure 4 - The HCGP Architecture

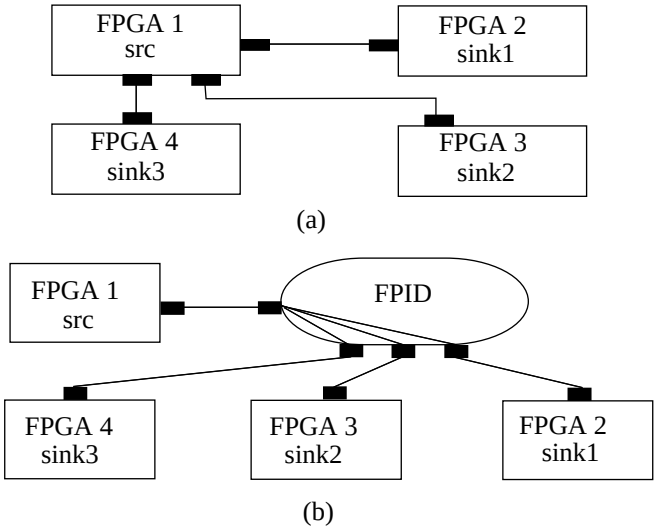

Figure 5 - Multi-terminal Net Routing (a) Without an FPID (b) With an FPID

in the sense that the same algorithm is used when routing nets through FPIDs. However, the difference here is that the router should also exploit the direct connections between FPGAs to minimize the number of FPGA and FPID pins used for routing and to minimize the net delay for critical inter-FPGA nets. A critical net is defined as an inter-FPGA net whose slack (when analyzed after partitioning, but before inter-FPGA routing) is less than the delay incurred for connecting two FPGAs via an FPID.

We have developed a timing-driven inter-chip routing tool, called HROUTE_TD, that understands the HCGP architecture and gives excellent routability and speed results for all the benchmark circuits.

The main objectives of HROUTE_TD are to try to route all critical nets using direct connections and to route all other (non-critical) nets using no more than two hops for each source-sink path. Our experience has shown that net ordering, based on slack first and then fanout, is crucial for obtaining good routability and speed. Wherever possible, HROUTE_TD uses direct connections to minimize source-sink net delay when routing critical nets. The HROUTE_TD algorithm works as follows: We first try to route all critical two-terminal nets using the direct connections between FPGAs to minimize usage of pins and net delay. Next, we try to route all multi-terminal nets through FPIDs using a routing algorithm similar to that used in PCROUTE, described above in Section 3.1.1. Finally, the remaining (non-critical) two terminal nets are routed using FPGAs or FPIDs. Any nets that remain unrouted are processed by a maze router. A detailed description of HROUTE_TD is given in [Khal99].

4 Experimental Results

In this Section we determine the effect of varying the value of P_p on the routability and speed of the HCGP architecture and compare the partial crossbar and HCGP architectures.

4.1 HCGP Architecture: Analysis of P_p

Recall the definition of P_p, given in Section 3.2, which is the percentage of pins per FPGA used for programmable connections. P_p is important because it affects the cost and routability of the HCGP architecture. Here we explore the effect of P_p on the routability and speed of the HCGP architecture. We mapped the fifteen benchmark circuits to the HCGP architecture using five dif-

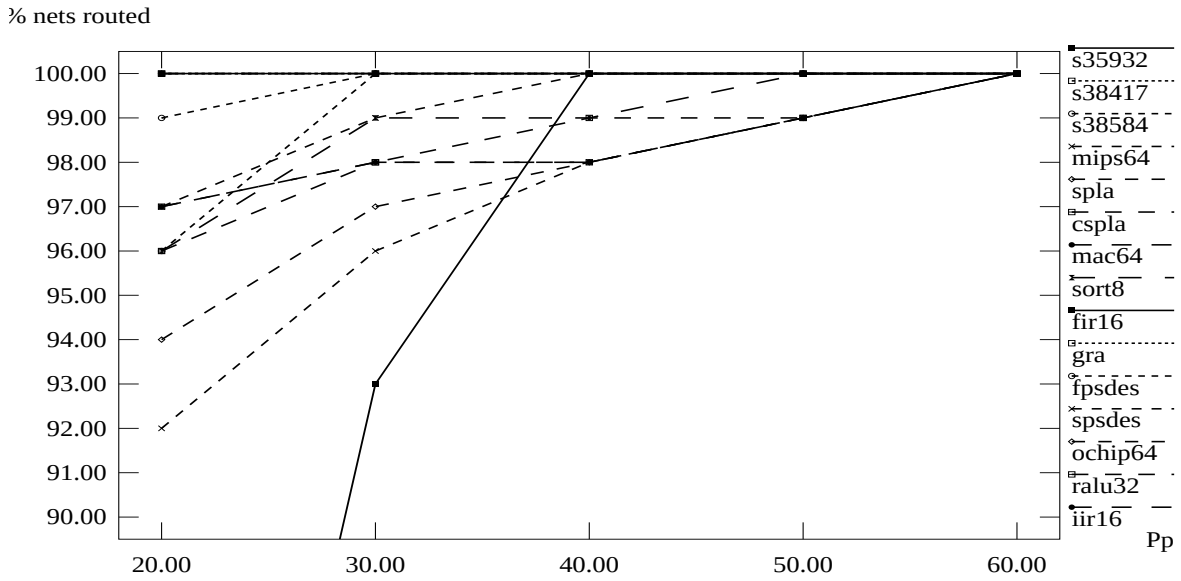

Figure 6 - The Effect of P_p on Routability of the HCGP Architecture

ferent values of P_p (20, 30, 40, 50, 60). The results are shown in Figure 6. The Y-axis represents the percentage of inter-FPGA nets routed and the X-axis represents the P_p values. The first clear conclusion is P_p = 60% gives 100% routability for all the benchmark circuits. Notice that about two thirds of the circuits routed at P_p <= 40%, and for the remaining one third, more than 90% of the nets routed. This implies that there is a potential for obtaining 100% routabilty for all circuits at P_p = 40% if we use a routability driven partitioner like the one used in [Kim96]. This will lead to further reduced pin cost for HCGP compared to the partial crossbar.

We conjecture that the P_p value required for routing completion of a given circuit on HCGP depends upon how well the circuit structure ‘matches’ the topology of the architecture.

We also investigated the effects of P_p on post-routing critical path delay. Table 3 shows the ten circuits that routed for P_p < 60%. The first column shows the circuit name. In subsequent columns, the critical path delay of each circuit for different values of P_p (20, 30, 40, 50, 60) is shown. A surprising conclusion is that (overall) the lower P_p values have no significant effect on the critical path delay. Compared to the delay value at P_p = 60%, for lower P_p values the delay remained the same or decreased slightly (only 4% less on average and 12% less in the best case). For circuits where delay was reduced, one or two programmable connections on the critical paths were replaced by faster hardwired connections. Note that as P_p is reduced, more hardwired connections are available. For circuits where delay remained the same, ‘segments’ on the critical path are part of very high fanout connections that have to be routed using FPIDs because of the lack of free pins (required for routing multi-terminal nets using hardwired connections). Even though more hardwired connections are available, they cannot be used for routing nets on the critical path.

Circuit	Post Routing Critical Path Delay (ns)				
	$P_p = 20$	$P_p = 30$	$P_p = 40$	$P_p = 50$	$P_p = 60$
s35932	unroutable	unroutable	53	53	53
s38417	87	94	94	94	94
s38584	unroutable	96	96	98	98
sort8	unroutable	unroutable	unroutable	460	499
fir16	147	160	163	167	167
gra	57	57	57	57	57
fpsdes	unroutable	173	176	176	176
spsdes	unroutable	unroutable	192	205	205
ochip64	50	50	50	50	50
iir16	143	143	143	152	152

Table 3: The Effect of P_p on Speed of the HCGP Architecture

4.2 Comparison of HCGP and Partial Crossbar

The 15 benchmark circuits described in Table 2 were mapped to the partial crossbar and HCGP architectures using the experimental procedure described in Section 2. The results obtained are shown in Table 4 and Table 5. In Table 4, the first column shows the circuit name. The second column shows the number of FPGAs needed for implementing the circuit on each architecture (recall that we increase the MFS size until routing is successful). The third column shows the pin cost normalized to the number of pins used by the HCGP architecture and the fourth column shows the normalized critical path delay obtained for each architecture. Table 5 is similar to Table 4 except that it shows actual (un-normalized) pin cost and delay values.

The number of FPIDs used is not shown because it is constant for each architecture. All the results for partial crossbar use $P_t = 17$. The parameter P_t determines the number of FPIDs required and the number of FPGAs in the architecture determine the pin count of each FPID. We have shown that the value of P_t used has no effect on the routability and speed of the partial crossbar [Khal97]. Therefore any arbitrary value of P_t can be used. However, for practical reasons, the value chosen should require FPIDs that have reasonable pin counts (about 400 pins or less, which are commercially available) for the largest partial crossbar required in our experiments. A reasonable choice in this respect is $P_t = 17$.

The value of P_p for the HCGP architecture was set to 60% to obtain good routability across all circuits, as discussed in Section 4.1. Notice that the parameter P_t also applies to the programmable connections in the HCGP. For the same reasons as in the partial crossbar (given in the previous paragraph), we chose $P_t = 14$ for the HCGP architecture. Also the number of global lines used in the HCGP architecture depends upon the MFS size (#FPGAs used) and the parameters P_p and P_t. In our experiments ($P_p = 60\%$, $P_t = 14$) the number of global lines used for the HCGP architecture varied from 5 to 15. Recall from Section 3.1 that the number of global lines for the partial crossbar is 5 corresponding to $P_t = 17$. The different values for number of global lines used in HCGP is due to the fact that the number depends upon both P_p and P_t instead of just P_t as in the partial crossbar architecture.

Circuit	Number of FPGAs		Normalized pin cost		Normalized post-routing critical path delay	
	Partial crossbar	HCGP	Partial crossbar	HCGP	Partial crossbar	HCGP
s35932	8	8	1.25	1.0	1.08	1.0
s38417	9	9	1.25	1.0	1.00	1.0
s38584	9	9	1.25	1.0	1.42	1.0
mips64	14	15	1.16	1.0	1.11	1.0
spla	18	18	1.25	1.0	1.16	1.0
cspla	18	18	1.25	1.0	1.18	1.0
mac64	6	6	1.25	1.0	1.34	1.0
sort8	12	14	1.07	1.0	1.07	1.0
fir16	10	10	1.25	1.0	1.43	1.0
gra	4	4	1.25	1.0	1.23	1.0
fpsdes	9	9	1.25	1.0	1.29	1.0
spsdes	8	8	1.25	1.0	1.21	1.0
ochip64	8	8	1.25	1.0	1.26	1.0
ralu32	9	14	0.80	1.0	1.21	1.0
iir16	6	6	1.25	1.0	1.05	1.0
Average	**10**	**10**	**1.20**	**1.0**	**1.20**	**1.0**

Table 4: Comparison of HCGP and Partial Crossbar Architectures

In reviewing Table 4, consider the circuit *mips64*. The first partitioning attempt resulted in 14 FPGAs required to implement the circuit on partial crossbar. However, the circuit was not routable on HCGP and the partitioning was repeated after reducing the number of pins per FPGA specified to the partitioner by 5%. This resulted in 15 FPGAs required to implement the circuit. The second partitioning attempt was routable on the HCGP architecture because more 'free pins' were available in each FPGA for routing purposes. The pin cost for the partial crossbar was still more than that for HCGP because it uses many more programmable connections, and hence more FPID pins. A partial crossbar always requires one FPID pin for every FPGA pin; the HCGP architecture requires a lower ratio, (0.6: 1) as shown in the previous section.

Inspecting Table 4, we can make several observations. First, the partial crossbar needs 20% more pins on average, and as much as 25% more pins compared to the HCGP architecture. Clearly, the HCGP architecture is superior to the partial crossbar architecture in terms of the pin cost metric. This is because the HCGP exploits direct connections between FPGAs to save FPID pins that would have been needed to route certain nets in partial crossbar. However, for routability purposes, the HCGP needs some free pins in each FPGA and may require repeated partitioning attempts for some circuits.

Table 4 also shows that the typical circuit delay is lower with the HCGP architecture: the HCGP gives significantly less delay for twelve circuits compared to the partial crossbar and about the same delay for the rest of the circuits. The reason is that the HCGP utilizes fast and direct con-

Circuit	Number of FPGAs		Pin cost		Post-routing critical path delay (in ns)	
	Partial crossbar	HCGP	Partial crossbar	HCGP	Partial crossbar	HCGP
s35932	8	8	3032	2432	57	53
s38417	9	9	3411	2736	94	94
s38584	9	9	3411	2736	139	98
mips64	14	15	5306	4560	462	418
spla	18	18	6822	5472	196	169
cspla	18	18	6822	5472	193	164
mac64	6	6	2274	1824	623	465
sort8	12	14	4548	4256	533	499
fir16	10	10	3790	3040	238	167
gra	4	4	1516	1216	70	57
fpsdes	9	9	3411	2736	227	176
spsdes	8	8	3032	2432	249	205
ochip64	8	8	3032	2432	63	50
ralu32	9	14	3411	4256	317	263
iir16	6	6	2274	1824	160	152
	Avg.: 10	**Avg.: 10**	**Total: 56092**	**Total: 47424**	**Avg.: 241**	**Avg.: 202**

Table 5: Actual Pin Cost and Delay Values for the Two Architectures

nections between FPGAs, whenever possible. From the delay values in Table 1, we can show that the interconnection delay is much smaller (12.6 ns) if we use direct connections between FPGAs compared to the delay value (25.6 ns) when connecting two FPGAs through an FPID. Another interesting observation is that even for the circuits where the HCGP needs more FPGAs compared to the partial crossbar, it still gives comparable or better delay value. This clearly demonstrates that the HCGP architecture is inherently faster due to the nature of its topology. It gives significant speed up, especially when we use timing driven inter-FPGA routing.

Table 5 shows the actual pin cost and delay values obtained for the partial crossbar and HCGP architectures. It is interesting that the estimated clock speeds for the partial crossbar architecture range from 20 MHz for the *ochip64* circuit to 1.6 MHz the *mac64* circuit. This range is representative of the clock rates expected in MFSs [Quic96].

5 Conclusions and Future Work

In this paper we have presented the Hybrid Complete-Graph and Partial-Crossbar (HCGP), a new routing architecture for multi-FPGA systems. Using an experimental approach, we evaluated and compared this architecture to the partial crossbar architecture and showed that it is superior in terms of pin cost and speed. To our knowledge, this is the first architectural study of board-level

MFSs that considers post-routing critical path delay when evaluating the speed performance of different architectures.

We explored a key parameter (P_p) associated with the HCGP architecture and experimentally determined its best value (60%) for obtaining good routability for a variety of circuits.

We believe that the HCGP architecture would give even better results if we use better mapping (CAD) tools for partitioning. A routabilty driven partitioner, similar to the one used in [Kim96], may result in further reduced pin cost by making circuits routable for even lower values of P_p (say 40%).

The HCGP architecture is suitable for single board MFSs using a maximum of about 25 FPGAs. As FPGA logic and pin capacities continue to rise, it makes sense to use single board systems using a few high capacity FPGAs to avoid the problems associated with using high pin count connectors for multi-board systems [Lewi97]. For applications where hundreds of FPGAs are needed, such as logic emulation, we could use 'clusters' of HCGPs interconnected using a hierarchical partial crossbar scheme [Butt92]. The hardwired connections, within each cluster and between different clusters, would still help in reducing the overall pin cost. Determining the P_p value suitable for such hierarchical architectures is an open research problem. We will need extremely large benchmark circuits and appropriate CAD tools to explore hierarchical architectures.

Acknowledgments

The authors would like to thank Dave Galloway for his help with the partitioning tool and Jason Anderson for his help in synthesizing the benchmark circuits. This research was supported by the Information Technology Research Center (ITRC) of Ontario and MICRONET.

References

[Alte94] Altera Corporation, *Reconfigurable Interconnect Peripheral Processor (RIPP10) Users Manual*, Version 1.0, 1994.

[Apti96] Aptix Corporation, *Product brief: The System Explorer MP4*, 1996. Available on Aptix Web site: http://www.aptix.com.

[Arno92] J. M. Arnold, D. A. Buell, and E. G. Davis, "Splash 2," *Proceedings of 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 316-322, 1992.

[Babb97] J. Babb et al, "Logic Emulation with Virtual Wires," *IEEE Trans. on CAD*, vol. 16, no. 6, pp. 609-626, June 1997.

[Brow92] S. Brown, R. Francis, J. Rose, and Z. Vranesic, *Field Programmable Gate Arrays*, Kluwer Academic Publishers, 1992.

[Butt91] M. Butts and J. Batcheller, "Method of Using Electronically Reconfigurable Logic Circuits," *U.S. Patent 5, 036, 473*, July 30, 1991.

[Butt92] M. Butts, J. Batcheller, and J. Varghese, "An Efficient Logic Emulation System," *Proceedings of IEEE International Conference on Computer Design*, pp. 138-141, 1992.

[Cass93] S. Casselman, "Virtual Computing and The Virtual Computer," *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, pp. 43-48, 1993.

[Chan93] P. K. Chan, M. D. F. Schlag, "Architectural Trade-offs in Field-Programmable-Device-Based Computing Systems," *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, pp. 152-161, 1993.

[Dray95] T. H. Drayer, W. E. King, J. G. Tront, and R. W. Conners, "MORRPH: A Modular and Reprogrammable Real-time Processing Hardware," *Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 11-19, 1995.

[Exem94] Exemplar Logic, *VHDL Synthesis Reference Manual*, 1994.

[Fidu82] C. M. Fiduccia, and R. M. Mattheyses, "A Linear-Time Heuristic for Improved Network Partitions", *Proc. of 19th ACM/IEEE Design Automation Conference*, pp. 241-247, 1982.

[FCCM] *Proceedings of IEEE Workshops/Symposia on FPGAs for Custom Computing Machines*, 1992 to 1998.

[Gall94] D. Galloway, D. Karchmer, P. Chow, D. Lewis, and J. Rose, "The Transmogrifier: The University of Toronto Field-Programmable System", *CSRI Technical Report (CSRI-306)*, CSRI, University of Toronto, 1994.

[Gall95] D. Galloway, "The Transmogrifier C Hardware Description Language and Compiler for FPGAs," *Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 136-144, 1995.

[Hauc94] S. Hauck, G. Boriello, C. Ebeling, "Mesh Routing Topologies for Multi-FPGA Systems", *Proceedings of International Conference on Computer Design (ICCD'94)*, pp. 170-177, 1994.

[Hutt96] M. Hutton, J.P. Grossman, J. Rose and D. Corneil, "Characterization and Parameterized Random Generation of Digital Circuits," *Proc.of the Design Automation Conference*, pp. 94-99, 1996.

[ICub97] I-Cube, Inc., *The IQX Family Data Sheet*, May 1997. Available at: www.icube.com.

[Joup87] Norman P. Jouppi, "Timing Analysis and Performance Improvement of MOS VLSI Designs," *IEEE Trans. on CAD*, vol. CAD-6, no. 4, pp. 650-665, July 1987.

[Khal95] M. A. S. Khalid and J. Rose, "The Effect of Fixed I/O Pin Positioning on The Routability and Speed of FPGAs," *Proceedings of The Third Canadian Workshop on Field-Programmable Devices (FPD'95)*, pp. 92-104, 1995.

[Khal97] M. A. S. Khalid and J. Rose, "Experimental Evaluation of Mesh and Partial Crossbar Routing Architectures for Multi-FPGA Systems," *Proceedings of the Sixth IFIP International Workshop on Logic and Architecture Synthesis (IWLAS'97)*, pp. 119-127, 1997.

[Khal99] M. A. S. Khalid, Routing Architecture and Layout Synthesis for Multi-FPGA Systems, *Ph.D. Thesis*, University of Toronto, 1999.

[Khal98] M. A. S. Khalid and J. Rose, "A Hybrid Complete-Graph Partial-Crossbar Routing Architecture for Multi-FPGA Systems," *Proc. of 1998 Sixth ACM International Symposium on Field-Programmable Gate Arrays (FPGA'98)*, pp. 45-54, February 1998.

[Kim96] C. Kim, H. Shin, "A Performance-Driven Logic Emulation System: FPGA Network Design and Performance-Driven Partitioning," *IEEE Trans. on CAD*, vol. 15, no. 5, pp. 560-568, May 1996.

[Knap96] D. W. Knapp, *Behavioral Synthesis: Digital System Design Using the Synopsys Behavioral Compiler*, Prentice Hall PTR, 1996.

[Lewi97] D. M. Lewis, D. R. Galloway, M. Van Ierssel, J. Rose, and P. Chow, "The Transmogrifier-2: A 1 Million Gate Rapid Prototyping System," *Proceedings of FPGA'97*, pp. 53-61, 1997.

[Lin97] S. Lin, Y. Lin, and T. Hwang, "Net Assignment for the FPGA-Based Logic Emulation System in the Folded-Clos Network Structure," *IEEE Trans. on CAD*, vol. 16, no. 3, pp. 316-320, March 1997.

[Mak97a] Wai-Kei Mak, D. F. Wong, "On Optimal Board-Level Routing for FPGA-based Logic Emulation," *IEEE Trans. on CAD*, vol. 16, no. 3, pp. 282-289, March 1997.

[Mak97b] Wai-Kei Mak, D. F. Wong, "Board-Level Multi-Terminal Net Routing for FPGA-based Logic Emulation," *ACM Trans. on Design Automation of Electronic Systems*, vol. 2, no. 2, pp. 151-167, April 1997.

[Prep96] *Programmable Electronics Performance Corporation*, HDL models for different circuits (synthesis benchmarks) are available on their Web site: http//www.prep.org.

[Quic96] Quickturn Design Systems, Inc., *System Realizer Data Sheet*, 1996. Available on Quickturn Web site:http://www.quickturn.com.

[Shih92] M. Shih, E. S. Kuh, "Performance-Driven System Partitioning on Multi-Chip Modules," *Proc. of the Design Automation Conference*, pp. 53-56, 1992.

[Syno97] Synopsys, Inc., *Design Compiler(Version 3.4a), Behavioral Compiler (Version 3.4a), and Library Com-*

piler (Version 3.4a), Reference Manuals. Documents available on-line.

[Van92] D. E. Van Den Bout, et al, "Anyboard: An FPGA-Based Reconfigurable System," *IEEE Design and Test of Computers*, pp. 21-30, June 1992.

[Varg93] J. Vargese, M. Butts, and Jon Batcheller, "An Efficient Logic Emulation System", *IEEE Trans. on VLSI Systems*, vol. 1, no. 2, pp. 171-174, June 1993.

[Vuil96] J. E. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, and P. Boucard, "Programmable Active Memories: Reconfigurable Systems Come of Age," *IEEE Transactions on VLSI*, Vol 4, No. 1, pp. 56-69, March 1996.

[Xili92] *The Programmable Gate Array Data Book*, Xilinx, Inc., San Jose, California, 1992.

[Xili94] Xilinx, Inc., *XACT Development System User Guide*, February 1994.

[Xili97] Xilinx, Inc., *Product Specification: XC4000E and XC4000X Series FPGAs*, Version 1.2, June 16, 1997. Available on Xilinx Web site: www.xilinx.com.

[Yang91] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide*, Version 3.0, Microelectronics Center of North Carolina, January 1991.