Multiple FPGA Partitioning with Performance Optimization

Kalapi Roy-Neogi and Carl Sechen

Dept. of Electrical Engineering, FT-10

University of Washington

Seattle, WA 98195

We address the problem of partitioning a technology mapped FPGA circuit onto multiple FPGAs of a specific target technology. The physical characteristics of the multiple FPGA system (MFS) pose additional constraints to the circuit partitioning algorithms: the capacity of each FPGA, the timing constraints, the number of I/Os per FPGA, and the pre-designed interconnection patterns of the MFS. Existing partitioning techniques which minimize just the cut sizes of partitions fail to satisfy the above challenges. We therefore present a rectilinear partitioning algorithm which efficiently and accurately handles timing specifications. The signal path delays are estimated during partitioning using a timing model specific to a multiple FPGA architecture. The model combines all possible delay factors in a system with multiple FPGA chips of a target technology. A new dynamic net-weighting scheme was incorporated to minimize the number of pin-outs for each chip. Finally, we have developed a graph-based global router for pin assignment which can handle the pre-routed connections of our MFS structure. We successfully partitioned the MCNC Xilinx FPGA benchmarks producing 100% routable designs with high utilization levels in all cases. Using the performance optimization capabilities in our approach we have successfully partitioned these benchmarks satisfying the critical path constraints and achieving a significant reduction in the longest path delay. An average reduction of 17% in the longest path delay was achieved at the cost of 5% in total wire length. We have proved the effectiveness of our performance optimization technique by verifying the timing predictions of our partitioner with the actual delays obtained after placement and routing of a partitioned MFS. Partitioning results obtained with the Xilinx mapped MCNC benchmarks are encouraging.

1 Introduction

Field Programmable Gate Arrays (FPGAs) are becoming a mainstream technology in board, system and application specific integrated circuit (ASIC) design processes. System-level ASIC designers are turning to FPGAs for design verification to take advantage of their low cost and fast prototyping. Current FPGA architectures can handle a maximum of only 6000 to 9000 gates compared to ASIC devices which offer hundreds of thousands. As a result, designers utilize multiple FPGAs when a single FPGA is not sufficient for a design implementation.

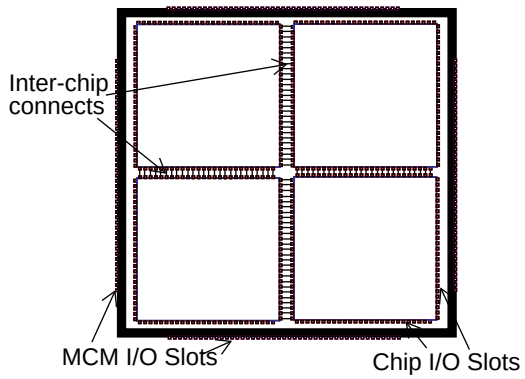

Fig.1 A Multiple FPGA System on an MCM.

Multiple re-programmable FPGAs have been configured on multichip modules (Figure 1) and on PCBs [9, 10]. A *multiple FPGA system* (MFS) can be modeled as a collection of FPGA chips configured on a single board or a package to realize a design. In order to effectively use MFSs and benefit from shorter time-to-market, users require an automatic method to partition a large design among multiple FPGAs. The quality of the partitioning results will influence several aspects of the design implementation:

1) *Capacity*: The partitioner must ensure that each chip contains a *feasibly* implementable amount of logic guided by the maximum gate capacity of the target FPGA architecture and utilization levels that can be handled by the placement and routing tools.

2) *Congestion in inter-chip communication*: The partitioner must be able to minimize the amount of inter-chip communications. The inter-chip connections of the packaging or the board design may or may not be fixed. The signals external to individual chips must be routed using the limited number of inter-chip connections to produce a feasible partitioning solution during *pin assignment* at the chip level. Any overflow generated during pin assignment will lead to a design which is not implementable

3) *Delay introduced for intra-chip and inter-chip communications*: High utilization of logic in individual chips causes congestion in intra-chip routing. This leads to longer paths and thus longer delays for signals internal to the chips. Also, the signals which cross one or more chip boundaries in the MFS will accumulate a substantial amount of delay associated with the I/O buffers and the inter-chip wire which can range from high, as in the case of PCBs, to moderate as in MCM based systems. The system cycle time will be determined by the length of the longest path from a primary input to the primary output of the entire MFS. The partitioner must satisfy the timing specifications for the MFS.

Thus the constraints of the MFS partitioning problem are: 1) Set of FPGA chips with their locations, dimensions and maximum capacity; 2) Configurations of the chip level I/O frames for inter-chip signals; 3) Configurations of the MFS package level I/O slots for the system I/O signals. In addition, the following design constraints need to be satisfied during MFS partitioning: 4) The timing constraints of the system being implemented; 5) Additional user constraints such as the utilization levels within the chips and the preplaced logic which must remain within a particular chip.

MFS partitioning over multiple chips can be performed before technology mapping onto the target FPGA or after technology mapping. If the partitioner manipulates the gate level netlist (as in [11]) before technology mapping, estimation of chip utilization and routability is difficult without the exact count of the target technology logic blocks. Also, there is no information at this level regarding delay of logic blocks and interconnects for an unmapped circuit. This is a major limitation because timing problems and achieving minimum system delay are very important for large complex designs which would typically be partitioned over multiple FPGAs. After technology mapping, the partitioner can take into account the target FPGA technology specific details such as total logic block count, the routing resources associated with each group of logic blocks assigned to each partition, and the actual timing information during the partitioning process. Hence, the partitioning process should follow the technology mapping stage.

Several approaches partition technology-mapped FPGA circuits onto multiple chips using multi-way netlist partitioning strategies [1,2,10]. These strategies only minimize the nets cut between partitions and do not understand the notion of distance between partitions. Thus, these methods cannot be deployed successfully to this problem since issues such as total wire length and the length of

critical signal paths cannot be controlled. Minimization of the number of pinouts on a partition is just one of the several important objectives of an MFS partitioning system. A cone partitioning/clustering approach [4] was used to partition Actel mapped FPGAs. Though this approach has the potential to minimize the critical paths while clustering, the path delay in this method can be calculated only based on intrinsic delay of the logic blocks. Without maintaining physical positions of the components during partitioning, it is difficult to estimate any routing delay which is an important component of the total delay of a path. Thus, timing cannot be ascertained accurately enough to be useful.

We therefore propose a rectilinear partitioning solution to the MFS partitioning problem which maintains the relative position of the partitions with respect to each other and thus maintains physical positions of the logic blocks during partitioning. This approach efficiently and accurately handles timing specifications. The signal path delays are estimated during partitioning using a timing model which combines all the possible delay factors involved in a system with multiple FPGA-based chips of a target technology. No previous work has offered a complete timing-driven model. We have incorporated a new dynamic net-weighting scheme to minimize the number of pin-outs for each chip. In order for each of the FPGAs to be placed and routed independently, pin assignment is performed on the I/O frames of each FPGA after partitioning. We have developed a graph-based global router for pin assignment which can handle the pre-routed connections of our MFS structure. Using the performance optimization capabilities in our approach, we have successfully partitioned these benchmarks satisfying the critical path constraints and achieving a significant reduction in the longest path delay. An average reduction of 17% in the longest path delay was achieved at the cost of only 5% in total wire length. Although our main interest was not minimizing the total dollar cost of FPGAs used, our results have outperformed the only results [1] which are available on the Xilinx mapped MCNC benchmarks by 7.5% in cost.

Our MFS partitioning system consists of two main phases: multi-FPGA partitioning and pin assignment, as shown in Figure 2. This paper is organized as follows. In section 2, we will discuss

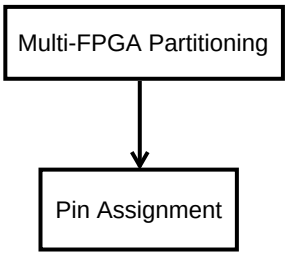

Fig.2 The MFS Partitioning System.

the previous work related to FPGA partitioning. In section 3, we describe the general features of our simulated-annealing based partitioning algorithm. Our new method of explicitly minimizing the number of chip-level I/Os is presented in section 4. Our new timing-driven capability specifically for multiple FPGA systems is described in section 5. The pin assignment stage is described in section 6. We present the results of the MCNC partitioning benchmarks in section 7. We conclude in section 8.

2 Previous Work

A review of general approaches to partitioning can be found in [16]. The earliest partitioning work specifically targeted for multiple FPGA partitioning approach was reported by Thomae *et al* [10]. They developed the Anyboard rapid prototyping system for digital hardware designs consisting of the Anyboard PC card made of multiple Xilinx FPGAs. A recursive mincut algorithm extended with additional FPGA constraints in which the total dollar cost of the FPGA devices was minimized for a set of Xilinx mapped ISCAS benchmarks was proposed in [1]. Another multiple FPGA partitioning approach based on group migration was proposed in [2]. This approach explicitly used the pin constraints during partitioning. Motivated by the new challenge of multiple FPGA partitioning, a gate replication technique was proposed to reduce the cut size of partitions [3]. A cone partitioning/clustering approach [4] was used to partition Actel mapped FPGAs and demonstrated better results than the classical mincut algorithm. This approach has the potential to minimize the critical paths while clustering. Several approaches applicable to technology mapping of logic circuits onto FPGA technologies have capability of performance optimization during partitioning of logic circuits [5, 6, 7, 8]. However, the path delay in the above methods can be calculated only based on intrinsic delay of the logic blocks.

Several commercial vendors have developed tools for partitioning FPGAs. The Prism software tool [12] from NeoCAD provides an environment to perform timing-driven partitioning over multiple FPGAs. InCA has an FPGA partitioner named Concept Silicon [13] which partitions an FPGA or PLD netlist onto multiple FPGAs. Quickturn's RPM emulation system [14] creates a hardware prototype from an ASIC or full-custom chip netlist. A hierarchical partitioner is used to partition the design over as many FPGAs as necessary.

3 Simulated-Annealing based Multiple FPGA Partitioning

The need to handle signal path timing constraints on very large designs forces us to base our MFS partitioning algorithm on a derivative of simulated annealing. Each chip in this *N*-(multiple) FPGA combination is considered a partition and each partition is subdivided into bins. Figure 3 shows an example of bin configurations for an MFS with six FPGAs. During the partitioning process, a component to be partitioned will move from bin to bin. Its location at any instant is taken to be the center of the bin to which it currently belongs. The finer the grid the cut lines produce, the higher the number of bins they generate. A large number of bins make the wire length calculations more precise, especially with respect to timing. However, with a large number of bins, the search space for component moves is large. This makes annealing more expensive in terms of CPU time. An effective trade-off between the accuracy of wire length and CPU time was obtained by using a number of bins on the order of four per partition on average. For

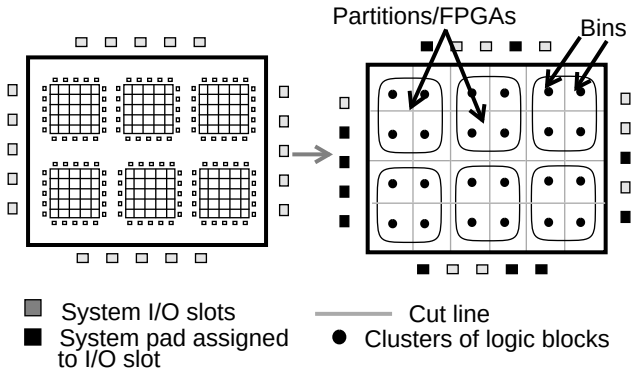

Fig.3 Physical definition of the MFS partitions.

new state generation, the algorithm only picks moves which are feasible in order to condense the search space of new states. The feasibility of a move is determined in terms of a pre-defined target utilization for each FPGA in the MFS structure. The system-level pin assignment or pad placement must be performed with respect to direct I/O connectivity of the logic blocks and thus is performed simultaneously during partitioning. The new state generation function picks moves which involve both logic blocks and system I/O pads in order to accomplish pad placement at the same time as the MFS partitioning. The annealing schedule used is a statistically derived schedule proposed by Lam [15]. We have formulated a new cost function which explicitly minimizes cutsize and thus number of I/Os on each FPGA and handles FPGA specific delay models to minimize the critical timing paths.

The cost function consists of two terms as shown in (1). The first term is the total weighted wire length, represented by *W.* The second term is the timing penalty function, represented by P_t.

$$C = W + P_t \quad (1)$$

4 Dynamic Net Weighting Scheme

At the end of partitioning, it is desirable to obtain the lowest number of *pin-outs* or I/Os possible for each chip, since there are a limited number of chip level I/O slots. If each net has the same

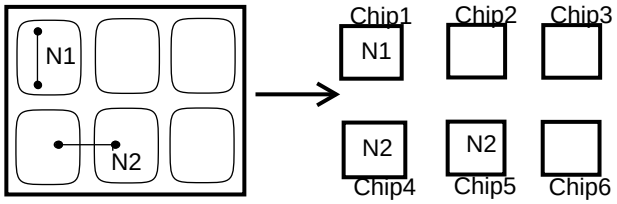

Fig.4 Comparing two nets of same length but different weights.

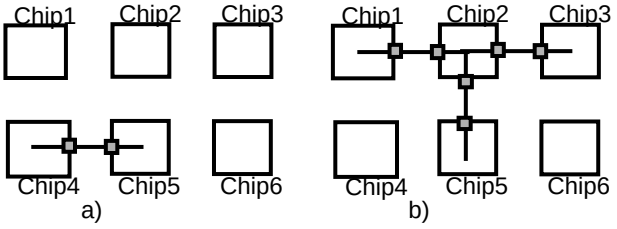

Fig.5 Net weighting scheme.

weight, minimizing the total wire length would not generally minimize the number of I/Os. We therefore introduce a new dynamic net-weighting scheme which minimizes the number of pin-outs. In our scheme, nets which traverse two adjacent bins but are in two different chips (*e.g.* N_2 in Figure 4) must be penalized more than nets which traverse two bins in a single chip *(e.g.* N_1*)*. Thus, our a net-weighting scheme is guided by the number of I/O's a signal needs if the net traverses more than one chip. The nets which are restricted to one chip, or the *single-chip* nets, do not need any I/Os and thus have a weight equal to 1.

Figure 5a) shows the situation when a net traverses two adjacent chips. This net needs at least two I/Os to make the connection between the two chips. Figure 5b) shows a net which traverses four chips and needs at least six I/Os. For each net topology encountered in a given MFS, integral weights are designed depending on the number of I/Os the net needs for inter-chip connections. The number of inter-chip connections, IO_n, is computed from the number of edges required to connect the net by the rectilinear minimum spanning tree. The weight of a net, *n*, is $w_n = 2IO_n + K$. A constant, K, increases the differences in weight from the *single-chip* nets. We use $K = 2$. We use an efficient bit manipulation technique to update the net weights dynamically during annealing. A look-up-table is maintained to store the weights for each net topology in an MFS. At any point during the annealing, the new topology of a moved net is determined and the corresponding weight is looked up and used to compute the weighted wire length. The summation of the half perimeters of the nets weighted by the dynamic net weight is the total weighted wire length, (as in (2)) for a particular configuration of cells. *W* is given by:

$$W = \sum_{n=1}^{N_n} (S_x(n) + S_y(n)) \cdot w_n, \quad (2)$$

where $S_x(n)$ and $S_y(n)$ are the width and height of the minimum bounding rectangle of the net, respectively, and w_n is the weight of net *n*.

5 Timing Penalty

The timing penalty in the cost function is calculated based on the slacks in the critical paths. A critical path may consist of several nets. The timing penalty is minimized dynamically during partitioning. We will first describe the propagation delay model for a timing path over multiple FPGAs. Based on this model, we will define the timing penalty.

The total delay on a path p over multiple FPGAs is the sum of the delay generated in the configurable logic blocks (CLBs) in each chip, $T_L(p)$ **,** and the total interconnect delay, $T_R(p)$.

$$T_{pd}(p) = T_L(p) + T_R(p) \quad (3)$$

$T_R(p)$ is the sum of the constituent net routing delays, $T_R(n)$, due to the *intra-chip* and *inter-chip* connections of the net.

$$T_R(p) = \sum_{n \in p} T_R(n) \quad (4)$$

Logic Delay: The total logic delay of a path p is:

$$T_L(p) = N_D \cdot T_{CLB}, \quad (5)$$

where N_D is the number of logic levels or depth of the particular critical path. T_{CLB} is the intrinsic delay of the CLB. For a given technology and CLB design of an FPGA, T_{CLB} is constant and independent of the configuration, number of inputs and outputs.

Routing Delay: The total routing delay of a net n, $T_R(n)$, is the sum of the delay due to the intra-chip, $T_S(n)$, and inter-chip connections, $T_M(n)$, of a net.

$$T_R(n) = T_S(n) + T_M(n) \quad (6)$$

Intra-chip routing delay: $T_S(n)$ is a function of: the routing architecture of the FPGAs used, fanout of a connection, length of a connection, the process technology, and the programming technology. The two main components of $T_S(n)$ are delay due to the switches in the interconnect path and the parasitics of the wire segments. The delay due to the switches can be modeled for a particular programming technology and the number of switching stages between CLBs in the routing architecture as shown in [20]. (For the anti-fuse technology and single segment routing, the number of switches between two logic blocks were taken to be 2 and the RC model was formed accordingly in [20]). The total switching delay including the parasitics seen by the wire segments (used by the net) can be modeled as a lumped RC:

$$T_S(n) = R_{SW}C_{SW} = R_{SW}(C_g + C_p) \quad (7)$$

R_{SW} is the equivalent drive resistance or the switching ON resistance and C_{SW} is the total load capacitance seen by the driver. C_{SW} consists of the gate input capacitance, C_g, and the parasitic capacitance, C_p, of the wire segments used to form the interconnection. C_p depends on the process technology used for the wiring segments and can be computed using the lumped capacitance model and is proportional to wire length. The wire length of a net can be estimated at the partitioning stage using the half-perimeter bounding box:

$$C_p = C_x S_x(n) + C_y S_y(n) \quad (8)$$

C_x and C_y are the capacitances (per unit length) of the vertical and horizontal tracks or busses in the routing architecture. Thus (7) can be expanded as:

$$T_S(n) = (R_{SW}C_g + R_{SW}[C_x S_x(n) + C_y S_y(n)]) \quad (9)$$

Inter-chip routing delay: In addition to the delay in the FPGA chips, a net acquires an additional delay when it crosses the chip boundaries. Depending on the type of MFS, MCM or PCB, the modeling of an interconnect between two chips will differ [18]. Interconnect wires on PCBs are usually wider (60-100 μ'm) and thicker (30-50 μ'm) than thin-film MCMs (where the wire width is in the range of 10-25 μ'm and thickness 5-8 μ'm). Figure 6 shows a generalized model for the interconnect of an inter-chip connection in an MFS following the macro-model described in [18]. The model consists of a transmitter capacitance, receiver capacitance and a transmission line modeling the wire segment in between them. The capacitor at the driving end, C_D, models the output capacitance of the driver and the pad capacitance of the chip on the MFS, while the capacitor on the receiving end, C_L , consists of the input capacitance of the receiver and the pad capacitance of the receiver. PCB interconnects usually have low resistance per unit length and thus behave like distributed LC transmission lines (lossless). These lines are generally terminated with a resistor that matches the characteristic impedance, Z_0, to avoid reflections. The total resistance of MCM interconnect lines is comparable to the characteristic impedance (which depends on the structural properties of the substrate) and are thus lossy. MCM interconnect lines are usually unterminated [19]. The inductance of the chip-to-MCM bond is assumed to be negligible which is typical for flip-chip-attached integrated circuits. The line parameters *R*, *L*, and *C* of the MFS interconnect will depend on the material properties such as the dielectric constant of the insulator (ε), the resistivity of the metal (ρ), the permittivity (μ) of free space and the line geometry of the wire.

Based on this model, the delay for a chip-to-chip interconnect, T_{CC} , appropriate to the particular MFS is pre-computed. We assume that a net which connects to more than one chip will be connected by the shortest path tree between the chips and we let

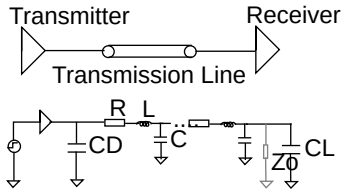

Fig.6 The inter-chip interconnect structure and the circuit used in delay modeling.

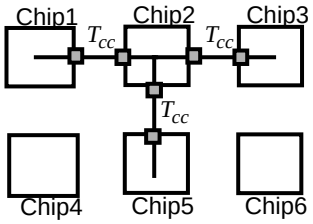

Fig.7 Inter-chip connection delay.

IO_n be the number of inter-chip connections a net requires under this assumption. In our MFS modelling, the spacing between the chips in Figure 7 is comprised of inter-chip connections which run perpendicularly to the cell edges. Hence, the total inter-chip connection delay for a net is:

$$T_M(n) = IO_n \cdot T_{CC} \tag{10}$$

The total path delay over all nets is:

$$T_{pd}(p) = T_L(p) + \sum_{n \in p} \left(\left(R_{SW} C_g + R_{SW} \left[C_{L_h} S_x(n) + C_{L_v} S_y(n) \right] \right) + IO_n \cdot T_{CC} \right)$$

The total timing penalty is computed as the sum of the penalties over all specified critical paths. For each critical timing path, the user supplies an upper bound $T_{ub}(p)$ and a lower bound $T_{lb}(p)$ on the required arrival times. The penalty assigned for a path p is the amount the delay deviates from satisfying the bounds.

$$P(p) = \begin{cases} T_{pd}(p) - T_{ub}(p) & \text{if} \quad T_{pd}(p) > T_{ub}(p) \\ T_{lb}(p) - T_{pd}(p) & \text{if} \quad T_{pd}(p) < T_{lb}(p) \\ 0 & \text{otherwise} \end{cases}$$

The total timing penalty is the sum of the penalties for all the critical paths specified.

$$P_t = \sum_{p=1}^{N_p} P(p)$$

6 Pin Assignment

In this section we will describe the second phase of the partitioning system. At the end of annealing, the system-level I/Os have been placed and each partition contains unplaced logic blocks. Following the rectilinear partitioning of the netlist, each of the *n* partitions (FPGAs) are converted into complete and independent layout problems in this phase. Pin assignment is performed on the chip-level I/O frames so that the chips in the MFS can be interconnected consistently using the pre-wired connections between the chips and those between the chips and system I/Os. This phase is mandatory for MFS partitioning in order to make the application complete. The signals which cross one or more chip boundaries are *external* signals. Given the total number of pin-outs for each partition, the objective is to assign the external signals to the chip level I/O's in a way such that there is no overflow.

We employ a graph-based global router in this phase. An example of the global routing graph for an MFS is shown in Fig.10a. A node is placed at the center of each bin. In order to route nets which connect pad pins, additional nodes are defined outside the MFS core as shown. All rectilinearly adjacent node pairs inside the core are connected by edges. However, to avoid route segments connecting adjacent pads, the edges connecting the

Set of external nets: N_{ex}; Set of edges with overflow E_o;
Set of routes for net n : R_n
Algorithm Global_Route_for_Pin_Assignment
  for all $n \in N_{ex}$
    r = generate_a_route(n, 0);
    $R_n = R_n \cup r$; /* add to the set of routes for net n */
  Calculate overflow on all edges and form E_o;
  Main_iteration = 0;
  While (total overflow > 0 OR Main_iteration ≤ MaxIteration)
    for all $e \in E_o$
      Update_edge_weight(e);
    for all $e \in E_o$
      for all $n \in e$
        r = generate_a_route(n, Max_improve);
        $R_n = R_n \cup r$;/* add to the set of routes for net n */
    Random_interchange(Main_iteration $\cdot |N_{ex}|$);
    Increment Main_iteration;
Subroutine Update_edge_weight(edge)
  if overflow(edge) > 0
    weight(edge) = ∞;
  else
    weight(edge) = length(edge);
Subroutine Random_Interchange(Rand_Iteration)
  iteration = 0;
  While (iteration < Rand_iteration)
    Randomly pick a net $n \in N_{ex}$;
    Randomly pick a route $r \in R_n$;
    Estimate total overflow with r in place of current route(n);
    if (total overflow decreases)
      current route(n) = r;
      update E_o;
  Increment iteration;

Fig.8 Global Routing Algorithm.

nodes which represent pads are excluded from the graph. A capacity is assigned to each edge. The edges which intersect any chip boundary are assigned a capacity equal to the number of preplaced interconnect wires or I/O slots available on that boundary within the range of that edge. All internal edges are assigned a large capacity to encourage the router to use these edges over the external edges if possible. Initially, the weight of an edge is equal to the length of the edge.

The global router seeks the shortest possible routes while minimizing the overflow over available routing resources. The global routing algorithm is shown in Figure 8. Initially, the shortest path routes are found for all external signals. Based on these routes, the total overflow is calculated. The function *Update_edge_weight* is used to update the weights of the edges with overflow. Routes which use edges with overflow are discarded and the corresponding signals are re-routed using an iterative rip-up and re-route scheme. The pseudo-code for generating a route for a net is shown in Figure 9.

Pin assignment is performed based on the final routes given by the global router for the external signals as shown in Fig.10b. For each route obtained for a net, we generate an I/O pin at each intersection of a chip boundary and a route segment. Let a net consisting of logic blocks in *Chip1*, *Chip2*, *Chip3* and *Chip4* be routed using the T-shaped route as shown in Fig.10a). I/O pins *a*, *b* and *c* are assigned at each of the intersection points of the route segments with the chip edges. Pin *a* is assigned to *Chip3* and *Chip2*, pin *b* is assigned to *Chip1* and *Chip2* and pin *c* is assigned to *Chip2* and *Chip4*. Since the routes follow the grid lines, a group of pins are likely to be produced at the same intersection point if several nets share that segment. In such cases, the pins are assigned in the same order on a shared chip boundary. *N* independent layout problems are created at the end of the global routing/pin assignment, such that they can be independently placed and routed.

Set of pins for net n : $P(n)$; Set of trees for net n : $T(n)$
Subroutine Generate_a_route(net *n*, Max_improve)
Make each node corresponding to $pin \in P(n)$ a tree and form $T(n)$;
while ($|T(n)| > 1$)
Find a shortest cost path p between two nodes, v_i and v_j, $v_i \in T_k$ and $v_j \in T_l$;/* $T_k \in T(n)$ and $T_l \in T(n)$ */
Merge the trees and path into one tree $T_k = T_k + T_l + p$; $T(n) = T(n) - T_l$;
if (Max_improve > 0) Improve_route_tree($T(n)$, Max_improve);
return ($T(n)$);

Subroutine Improve_route_tree(*T*, Max_improve)
iteration = 0;
while (iteration < Max_improve)
Select a random edge $e \in T$;
Create a path p_1 by tracing e to nodes with degree $d > 2$;
Create two trees T_1 and T_2 by removing p_1 from T ;
Find the shortest cost path p_2 between two nodes $v_i \in T_1$ and $v_j \in T_2$;
If cost(p_2) < cost(p_1) then $T = T_1 + T_2 + p_2$;
else $T = T_1 + T_2 + p_1$;
Increment iteration;

Fig.9 Subroutine Generate_a_route for a net.

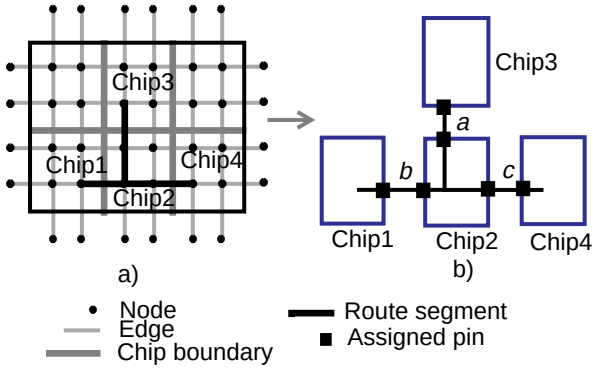

Fig.10 a) A global route on the graph. b) Pin assignment based on a route.

7 Results

Our MFS partitioning system, *MFSP*, has been developed in C with an X11 graphics interface. Although the implementation and formulation of MFSP was general, we used MFS structures consisting of Xilinx FPGAs to demonstrate its effectiveness. We used the MCNC *partitioning93* benchmarks which were the ISCAS benchmark circuits mapped onto the Xilinx 3000 series devices. The parameters for the five classes of the XC3000 device family we used in our experiments are shown in Table 1. CLB represents the number of configurable logic blocks and IOB represents the number of I/Os in each device. The cost in dollars is normalized to the smallest device and shown in the last column. The characteristics of the mapped ISCAS circuits are shown in Table 2.

Performance Optimization

We first tested the timing-driven capabilities of the partitioner on the MCNC benchmark circuits. Multiple FPGA configurations using Xilinx devices on a PCB were used to obtain these results. The critical path constraints for MFSP are normally user-specified. Due to lack of any standard timing constraints available for these benchmarks, we conducted the following experiment to verify the

Table 1 Devices used from Xilinx 3000 series.

Device	Type No.	CLB	IOB	Cost ($)
XC3020xx-xx	1	64	64	1.00
XC3030xx-xx	2	100	80	1.36
XC3042xx-xx	3	144	96	1.84
XC3064xx-xx	4	224	110	3.15
XC3090xx-xx	5	320	144	4.83

Table 2 XC3000 mapped ISCAS circuits.

Circuits	CLBs	#nets	IOBs	#pins
c1355xc3	70	115	73	399
c1908xc3	116	191	58	683
c2670xc3	150	361	221	1006
c3540xc3	283	489	72	1645
c5315xc3	377	699	301	2409
c6288xc3	833	1472	64	3438
c7552xc3	489	921	313	2924
s1196xc3	143	226	30	850
s1238xc3	158	251	30	934
s1423xc3	112	188	24	647
s5378xc3	381	628	86	2332
s9234xc3	454	716	43	2671
s15850xc3	915	1377	154	4977
s13207xc3	842	1265	102	5309
s38584xc3	2901	3884	292	17483

effectiveness of the timing penalty of the partitioner. For each circuit we first used MFSP to find a partitioning without imposing any delay bounds. Using the nominal net lengths obtained from this run, we extracted (in order) the *m* most delay critical primary input (PI) to primary output (PO) pin pairs. (The value of *m* was limited so as to not more than double the overall CPU time versus the case when no delay bounds are imposed. We verified that none of the non-included pin pairs gave rise to a critical delay at the conclusion of the partitioning).

We extract the current longest path between these pins. These constitute the set of critical paths used in our timing penalty function described in section 5, and we impose the delay bound on these paths. Because a particular critical path may not always be the critical path for a pair of pins, we update the set of critical paths once every iteration during the course of the annealing based

Table 3 Performance driven partitioning.

Circuits	#FPGAs used	# Critical PI/PO pairs	No constraints		With constraints	
			Within Spec.	Outside Spec.	Within Spec.	Outside Spec.
c1355	{2,0,0,0,0}	100	100	0	100	0
c1908	{2,0,0,0,0}	100	53	47	100	0
c2670	{0,0,4,0,0}	100	97	3	100	0
c3540	{0,0,3,0,0}	81	50	31	81	0
c5315	{0,2,2,0,0}	540	32	508	540	0
c7552	{0,0,4,0,0}	450	108	342	450	0
s1196	{0,2,0,0,0}	110	28	82	110	0
s1423	{2,0,0,0,0}	40	24	16	40	0
s1238	{0,2,0,0,0}	67	46	21	67	0
c6288	{0,0,0,6,0}	81	53	28	81	0

partitioning.

We compared the results with the timing penalty deactivated versus the results obtained with the timing penalty activated for each circuit. In Table 3 we have shown the number of paths which were within specifications and the number of paths which were outside the specifications in both cases. Column 1 shows the vector

Table 4 Performance driven partitioning.

Circuits	# Critical PI/PO pairs	Increase (nets cut) %	Increase (wire length) %	Reduction (longest path)%
c1355	100	5	3.4	30
c1908	100	22	4.5	21
c2670	100	3	3.7	7
c3540	81	16	1	10
c5315	540	8.6	1.6	30
c7552	450	26	12	32
s1196	110	17	3.2	19
s1423	40	23	8.6	12.5
s1238	67	24	5.6	3
c6288	81	23	2.6	4

which accounts for the number and type of devices used with the vector index corresponding to the type number in Table 1. Using our timing penalty function, MFSP successfully partitioned these circuits satisfying the timing constraints in all cases. As shown in Table 4 in all the circuits, MFSP achieved a significant reduction in the longest path delay by using the timing penalty function. The average reduction was 17%. These results were obtained at the cost of 17% in nets cut and 5% in wire length on average.

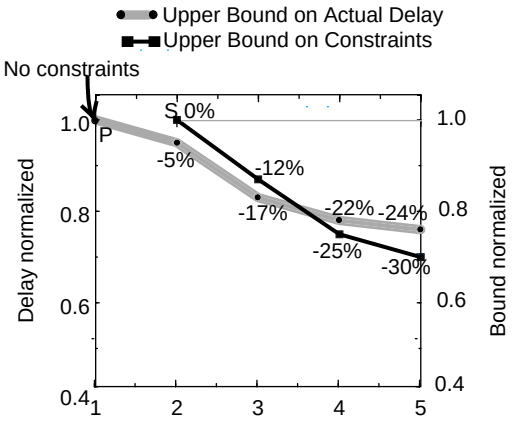

Fig.11 Delay reduction with constraint tightening on the circuit c3540xc3.

We also verified the effectiveness of the performance optimization capabilities in MFSP. Since we do not have available a tool which can determine the actual delays over a multiple-chip combination on a PCB, we determined the longest path delay for each chip using the Xilinx tools and summed these to obtain a worst-case upper bound on delay. When we ran MFSP on circuit c3540xc3 with no critical path constraints, the upper bound on the actual delay corresponds to point P in Figure 11. We then ran MFSP four additional times, the first time with the nominal delay bound (corresponding to point S) and then reducing (or tightening) the critical path delay bounds (by 12%, 25% and 30%). Notice from Figure 11 that the initial application of the delay bounds reduces the upper bound on the actual delay by 5% and then that each step of tightening the bounds monotonically reduces the upper bound yet further, ultimately by 24%. We have therefore shown that a monotonic tightening of the delay bounds in MFSP corresponds to a monotonic reduction in the upper bound on the actual delay of a placed and routed multiple FPGA system.

Cost Comparison

The only other partitioner for this problem whose results are available for this set of benchmarks is *NC* from North Carolina State University [1]. This approach minimized the total dollar cost of devices used to implement a circuit onto multiple FPGAs. Though we do not explicitly minimize the dollar cost of devices used, we used MFSP to partition the nine largest circuits from Table 2 to compare our costs with NC. Each circuit was partitioned into *n* FPGAs on an MFS structure. The output of MFSP is *n circuit*.map files corresponding to the *n* FPGAs. Each of the FPGAs was placed and routed using the automatic placement and routing tools from Xilinx, *apr*, which was executed in the default mode. We present the partitioning results in Table 5. Column 2 shows the distribution of the Xilinx devices used in each case. Note the distribution of the devices is determined by using the minimum size device which would fit each partition obtained from our program. We note the total dollar cost of devices used by each circuit in column 3. The average CLB and IOB utilizations are listed in columns 4 and 5 respectively. For all partitions for each circuit, we obtained 100% routable FPGA chips. The total CPU time required

Table 5 MFS partitioning results.

Circuits	Device distribution	Total cost	CLB utilization	IOB utilization
c3540	{0,2,1,0,0}	4.56	0.84	0.95
c5315	{0,2,2,0,0}	6.4	0.77	0.92
c6882	{0,0,4,2,0}	13.66	0.81	0.68
c7552	{0,0,4,0,0}	7.36	0.85	0.90
s5378	{0,3,1,0,0}	5.92	0.86	0.90
s9234	{0,1,3,0,0}	6.88	0.85	0.91
s13207	{0,0,0,2,2}	15.96	0.81	0.89
s15850	{0,0,5,0,1}	14.03	0.81	0.85
s38584	{2,10,0,0,7}	49.41	0.86	0.58

Table 6 Cost Comparison with [1].

Circuits	NC		MFSP		Cost Reduction (%)
	Device distribution	Total cost	Device distribution	Total cost	
c3540	{0,0,3,0,0}	5.52	{0,2,1,0,0}	4.56	17
c5315	{2,1,2,0,0}	7.03	{0,2,2,0,0}	6.4	9
c6882	{0,0,4,2,0}	13.66	{0,0,4,2,0}	13.66	0
c7552	{0,0,4,0,0}	7.36	{0,0,4,0,0}	7.36	0
s5378	{0,0,1,0,1}	6.67	{0,3,1,0,0}	5.92	11
s9234	{0,0,0,1,1}	7.98	{0,1,3,0,0}	6.88	14
s13207	{3,5,4,0,0}	17.16	{0,0,0,2,2}	15.96	7
s15850	{0,0,2,2,1}	14.80	{0,0,5,0,1}	14.03	5
s38584	{0,5,15,4,1}	51.83	{2,10,0,0,7}	49.41	5

the on the largest circuit s38584 is less than 45 minutes on a DEC 5000. Note this includes the CPU time required for automatic system pad placement and the pin assignment stage.

We show the results from [1] in Table 6. Although our approach does not explicitly minimize the total dollar cost of devices used, we improved the cost by 7.5% on average over NC. We also compared our cost with the theoretical lower bound of cost calculated in [1] by an integer programming technique, *lp_solve*. Although it may be unrealistic to expect that we can achieve the lower bounds of cost obtained from *lp_solve* which had no constraints on the terminals or pinouts on a partition nor on the number of nets cut between partitions, on average our approach achieved only 7% higher cost than the theoretical lower bound. Figure 12 compares the costs between the lp_solve, MFSP and NC programs.

However, in addition to the dollar cost, the average CLB utilization (total number of CLBs in a circuit divided by the total capacity of the FPGAs used) and average IOB utilization (total number of IOBs required by the partitions divided by the total number of IOBs on the FPGAs used) are two important factors which reflect on the effectiveness of a partitioner. In Figure 13 we compare our average CLB and IOB utilization with the results from NC. For all circuits except s15850, MFSP achieved higher utilization than NC. On average the improvement was 7% in CLB utilization. IOB utilization achieved was 7.5% higher than NC on average. But since all the partitions obtained from MFSP were hundred percent routable, the additional usage of IOBs is not a disadvantage.

8 Conclusions

We have presented a new multiple FPGA partitioning system. This approach efficiently and accurately handles timing specifications. The signal path delays are estimated during partitioning using a timing model which combines all possible delay factors in a sys-

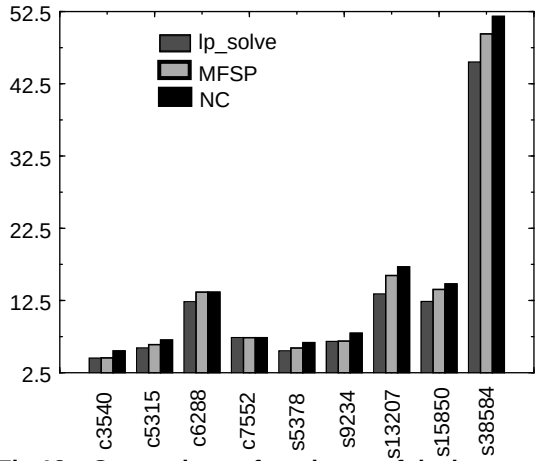

Fig.12 Comparison of total cost of devices used.

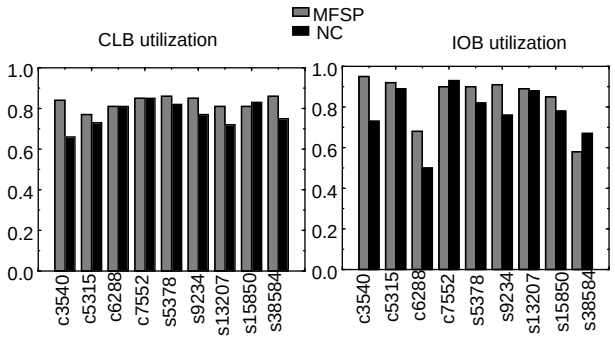

Fig.13 CLB Utilization and IOB utilization.

tem with multiple FPGA-based chips of a target technology. Furthermore, we have incorporated a new dynamic net-weighting scheme to explicitly minimize the number of chip-level I/Os. Finally, we have developed a graph-based global router for pin assignment which can handle the pre-routed connections of our MFS structure. Using the performance optimization capabilities in our approach we have successfully partitioned these benchmarks satisfying the critical path constraints and achieving a significant reduction in the longest path delay. An average reduction of 17% in the longest path delay was achieved at the cost of 5% in total wire length. We have proved the effectiveness of the performance optimization technique by verifying the timing predictions of our partitioner with the actual delays obtained after placement and routing of a partitioned MFS. In comparison to the only other partitioning system which was applied to the Xilinx mapped MCNC benchmarks [1], we produced partitioned results with 7.5% lower total dollar cost.

In order to overcome the pin limitations in a multiple FPGA structure, several new interconnect designs for inter-FPGA communications have been proposed [9,14,21,22,23]. The rectilinear nature of MFSP, makes it very appropriate for application in these high performance prototyping environments. The graph used by the global router in the pin assignment stage needs to be appropriately modified to accommodate new interconnect patterns. Hence future work will be directed toward accommodating new MFS designs with new interconnect wiring designs.

9 References

[1] R. Kuznar, F. Brglez, and K. Kozminski, "Partitioning Digital Circuits for Implementation in Multiple FPGA ICs," Technical Report TR93-03, MCNC, 1993.

[2] N. Woo and J. Kim, "An Efficient Method of Partitioning circuits for Multiple-FPGA Implementation", *Proceedings of Design Automation Conference*, pp. 202-207, 1993.

[3]J. Hwang and A. E. Gamal, "Optimal Cell Replication", *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 432-435, November 1992.

[4] G. Saucier, D. Brasen, and J. P. Hiol, "Partitioning with Cone Structures", *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 236-239, November 1993.

[5] R. Murgai, R. K. Brayton, and A. Sangiovanni-Vincentelli, "On Clustering for Minimum Delay/Area," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 6-9, November 1991.

[6] R. Rajaraman and D. F. Wong, "Optimal Clustering for Delay Minimization", in *Proceedings of Design Automation Conference*, 1993, pp. 309-314.

[7]P. Sawkar and D. Thomas, "Performance Directed Technology Mapping for Look-up table based FPGAs", in *Proceedings of Design Automation Conference*, 1993, pp. 208-212.

[8]J. Cong and Y. Ding, "On area/depth trade-off in LUT-based FPGA technology mapping", in *Proceedings of Design Automation Conference*, 1993, pp. 213-218.

[9] P. K. Chan, M. Schlag, and M. Martin, "BORG: A Reconfigurable Prototyping Board for FPGAs," *FPGA 92, First International ACM/SIGDA Workshop on Field Programmable Gate Arrays*, 1992, pp. 47-51.

[10] D. Thomae, T. Petersen, and D.E. Van den Bout, "The Anyboard Rapid Prototyping Environment," in Carlo H. Sequin, editor, *Advanced Research in VLSI*, pp. 356-370, MIT Press, 1991.

[11] W. O. Mcdermith, "A Bottom-Up Approach to FPGA Partitioning," in *Proc. IEEE Custom Integrated Circuit Conference*, 1992, pp. 5.4.1-5.4.4.

[12] K. Perry, "Eliminating Barriers to FPGA use by Timing Driven Partitioning," *Electronic Engineering*, Jan. 1993, pp. 41-44.

[13] "Concept Silicon Partitions your Design onto Multiple FPGAs," *Integrated Circuit Applications Pamphlet*, InCA Inc., Campbell CA 95008, 1992.

[14] S. Walters, "Computer-aided Prototyping for ASIC-based Systems," *IEEE Design and Test of Computers*, June 1991, pp. 4-10.

[15] J. Lam and J. M. Delosme, "Performance of a New Annealing Schedule," in *Proc. 25th Design Automation Conference*, 1988, pp. 306-311.

[16] K. Roy and C. Sechen, "A Timing Driven *N*-way Chip and Multi-Chip Partitioner", *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 240-247, November 1993.

[17] S. Singh *et al.*, "Optimization of Field Programmable Gate Array Logic Block Architecture for Speed," in *Proc. IEEE Custom Integrated Circuit Conference*, 1991, pp. 6.1.1-6.1.6.

[18] A. I. Kayssi and K. A. Sakallah, "Delay Macromodels for Point-to-Point MCM Interconnections," in *Proc. IEEE Multi-chip Module Conference*, 1992, pp. 79-82.

[19] C. W. Ho *et al.*, "The Thin-film Module as a High Performance Semiconductor Package," in *IBM J. Res. Develop*, **26**, 1987, pp. 286.

[20] J. L. Kouloheris and A. E. Gamal, "FPGA Performance versus Cell Granularity," in *Proc. IEEE Custom Integrated Circuit Conference*, 1991, pp. 6.2.1-6.2.4.

[21] I. Dobbelaere, "Peripheral Circuit Design for Field Programmable MCM Systems," in *Proc. IEEE Multi-chip Module Conference*, 1992, pp. 119-22.

[22] R. Guo *et al.*, "A 1024 Pin Universal Interconnect Array with Routing Architecture", *Proc. Custom integrated Circuits Conference*, 1992, pp.4.5.1-4.5.4.

[23] J. Babb *et al.*, "Virtual Wires: Overcoming Pin Limitations in FPGA-based Logic Emulators", *IEEE Workshop on FPGAs for Custom Computing machines*, Napa, CA, April 1993.