K-Way Partitioning Under Timing, Pin, and Area Constraints

Gregory Tumbush and Dinesh Bhatia
Design Automation Laboratory
Department of Electrical and Computer Engineering and Computer Science
University of Cincinnati
Cincinnati, OH 45221–0030
Voice: (513)-556-2570
FAX: (513)-556-7326
email: dinesh.bhatia@UC.EDU

Abstract

Circuit partitioning is a very extensively studied problem. Our proposed methodology easily extends to multiple constraints that are very dominant in the design of large scale VLSI Systems. In this paper we formulate the problem as a nonlinear program (NLP). The NLP is solved for the objective of minimum cutset size under the constraints of pins, area, and timing. We have tested the unified framework for area, timing, and pin constraints. The NLP is solved using the commercial LP/NLP solver MINOS. We have done extensive testing using large scale RT level benchmarks and have shown that our methods can be used for exploring the design space for obtaining constraint satisfying system designs. We also provide extensions for solving system design problems where a choice between multiple technologies, packaging components, performance, cost, yield, and more can be the constraints for design related decisions.

1 Introduction

Ever changing complexity of VLSI systems requires support from CAE tools for automated decision making capability. Also, important design related decisions should be made early in the design process. This requires tools that have the capability to explore design choices, make tradeoffs between various constraints, and select/reject design options so as to obtain a very high quality constraint satisfying solution. Motivated with this task of automating the system design process we have conducted this research for *system level partitioning* problems.

In system level partitioning, a designer is presented with an application (design), a set of requirements, a set of options for realizing the design, and a set of constraints for implementing or physically realizing the overall design. In a typical design such parameters would include choice of packaging options, i.e., ICs from various technologies, their area and pin constraints, their costs, timing requirements on the overall design, yield, testability, and more. In the presence of such choices the designer must try to optimize the resources such that the final design implementation satisfies as many constraints as possible.

In this paper, we have modeled the problem of partitioning in the presence of multiple constraints as a non-linear programming problem (NLP) and have presented effective solutions for partitioning designs in the presence of *area*, *timing*, and *pin* constraints.

In[5] we perform bipartitioning under timing and area constraints only. We extend this work to k-way partitioning and now include pin constraints. We generate k vs P relationships for each

0-7803-4275-5/97 $8.00

benchmark under pin and area constraints only, where k is the number of partitions and P is the pins per package. Then at each k vs P data point we introduce timing constraints to find the minimum critical path delay under the same pin and area constraints.

2 k-way Partitioning Under Timing, Pin, and Area Constraints

Given a set of n netlist modules $V = \{v_0, v_2, \ldots, v_{n-1}\}$ and m distinct pairs of vertices called *edges* $E = \{e_1, e_2, \ldots, e_m\}$, we may represent the circuit as a graph $G = (V, E)$. The cardinality of every edge is 2, i.e. $|e| = 2$. A vertex $v_x, x = 0, \ldots, n-1$ is adjacent to a vertex $v_y, y = 0, \ldots, n-1$ if (v_x, v_y) is an edge, i.e., $(v_x, v_y) \in E$. A graph $G' = (V', E')$ is a *subgraph* of graph G if and only if $V' \subseteq V$ and $E' \subseteq E$.

We may also represent the circuit as a hypergraph $G = (V, E'')$ with n vertices and a set $E'' \subset 2^V$ of *hyperedges* or *nets*. Unlike the graph representation, the cardinality of every hyperedge is greater than or equal to 2. The vertices in a *hyperedge* are called the *terminals* of the hyperedge. The hypergraph will more directly model an actual circuit. Figure 1 shows a hypergraph and it's representation as a graph[3].

Given a set of n netlist modules the goal of k-way partitioning is to assign each $v_i, i = 0, \ldots, n-1$, to a specified number k of segments. If $k = 2$, the problem becomes that of graph bipartitioning. An edge e is *cut* if both terminals of e are not within a single segment. The total number of cut edges under a graph model is called the size of *cutset*. Typically one chooses to minimize the size of cutset according to some pre-defined criteria. In this paper we represent a circuit as a graph. We then perform k-way graph partitioning under timing, pin, and area constraints. The size of cutset is evaluated for the equivalent hypergraph representation in the circuit. It is known that minimizing the graph cutset of a circuit will also minimize the hypergraph cutset. For Figure 1, if terminals 1 and 2 are in one segment and terminals 3 and 4 are in another the cutset is two for the hypergraph and three for the graph.

The input to the partitioner is a netlist and the area of each netlist component. We represent each net as a clique of size equal to the number of terminals in the net and then optimize the graph cutset size over all nets according to capacity, i.e., area constraints while varying the pin constraints. For a specific pin constraint we attempt to obtain a satisfying partition with minimal k, that is $k = \lceil \frac{\sum_{j=0}^{n-1} a_j}{A} \rceil$, where a_j is the area of cell j and A is the area constraint. If a satisfying partition is not produced we increment k until a feasible partition is found. In this way we find the minimum k required for a wide range of P, the pins per package. This will produce a k vs P relation for each benchmark. We then incorporate $k \cdot T$ timing constraints which are derived from the T critical timing paths. The minimum critical path delay is found at each k vs P data point for each benchmark by attempting to cut none of the critical paths. The allowed cuts per critical path increases by one until a feasible partition is found. In this way we

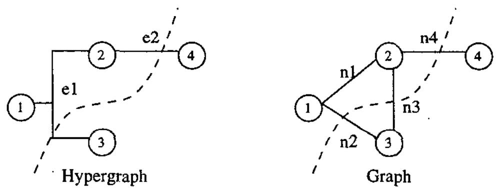

Figure 1: *Hypergraphs and Graphs*

find the minimum delay for each k vs P data point.

The CO problem is solved as an assignment problem. We associate a variable $x_i, 0 \leq i \leq n \cdot k - 1$ for n components and k segments. For example, if $n = 2$ and $k = 3$ then $x_0 = 1$ if component 0 is on segment 1, 0 otherwise, $x_1 = 1$ if component 1 is on segment 1, 0 otherwise, $x_2 = 1$ if component 1 is assigned to segment 2, 0 otherwise,...,$x_5 = 1$ if component 1 is on segment 3, 0 otherwise. In general, for k segments, if $x_i = 1$ then cell $i \cdot mod\ n$ is assigned to segment $\lfloor \frac{i}{n} + 1 \rfloor$. Each cell has k assignment variables. A solution to the NLP problem can result in non-integer assignment to x_i which will not form a feasible partitioning solution. Thus, fractional assignment variables have to be rounded for generating a feasible partitioning solution. We employ 0-1 rounding for changing the fractional assignments to an integer form. This can be done simply by choosing a value, *median*, and if $x_i \geq median$ set x_i to 1, 0 otherwise. However, it is possible that in some cases all k of cell j's assignment variables are less than *median*, i.e. $x_{j+n\cdot(c-1)} < median$, $c = 1, 2, \ldots, k$. If this is the case randomized rounding is employed. Given a fractional assignment variable, $x_i = p$, randomized rounding will round this variable to 1 with a probability p. Only one of cell j's k assignment variables can be assigned a value of 1 to obtain a feasible partition.

2.1 Partitioning under Pin and Area Constraints

Consider a three cell net, 0, 1, and 2, to be partitioned into three segments, i.e. $k = 3$. Let $x_0 = 1$ if component 0 is on segment 1, 0 otherwise, $x_1 = 1$ if component 1 is on segment 1, 0 otherwise, $x_2 = 1$ if component 2 is on segment 1, 0 otherwise, $x_3 = 1$ if component 0 is on segment 2, 0 otherwise,..., $x_8 = 1$ if component 2 is on segment 3, 0 otherwise. The cutset of this net will be:

$$\begin{aligned} cutset = \quad & x_0x_4 + x_0x_7 + x_0x_5 + x_0x_8 + x_1x_3 + x_1x_5 + x_1x_6 + x_1x_8 + x_2x_3 + \\ & x_2x_6 + x_2x_4 + x_2x_7 + x_3x_7 + x_3x_8 + x_4x_6 + x_4x_8 + x_5x_6 + x_5x_7 \end{aligned} \tag{1}$$

In general the graph cutset for a net is

$$\sum_{\forall r \in M} \sum_{c=1}^{k-1} \sum_{d=c+1}^{k} \sum_{\forall i \in Q_r} \sum_{\forall j \in Q_r \neq i} x_{i+n\cdot(c-1)} x_{j+n\cdot(d-1)} \tag{2}$$

where Q_r is the set of all non I/O elements on net r and M is the set of all nets.

As with any partitioning problem formulation, minimizing the *cutset* size is the most important objective for our formulation. A typical VLSI circuit contains majority of nets that are small, i.e., two to four terminals. Hence, in our implementation, for very large nets, we drop out the terms in the above mentioned expression. Large nets require many terms to model correctly, wasting time and memory. However, we always account for an extra (possible) cut in our cutset size evaluation process.

2.1.1 Pin Constraints

In addition to minimizing the cutset, we also consider the pin constraints. Any net must contain all I/O elements, a mixture of I/O elements and cells, or contain all cells. Obviously, a net containing all I/O elements imposes no additional pins while if it contains a mixture of I/O elements and cells will require a pin on whatever segment a cell is assigned to. A net containing all cells requires a pin on whatever segment a cell on the net is assigned to only if the net is cut. A net can require zero or one pin on a chip. A net as related to pins can only be cut once and must be modeled exactly in this case.

First we will derive the pin constraint for a net with all cells. Given a net with two cells, 1 and 2, let $x_1 = 1$ if cell 1 is on chip 1, 0 otherwise, and $x_2 = 1$ if cell 2 is on chip 1, 0 otherwise. The exact *logical* expression for the number of pins required on chip 1 for this net is $\overline{x_1}x_2 + \overline{x_2}x_1$. Using DeMorgans theorem this expression becomes $\overline{\overline{\overline{x_1}x_2} \cdot \overline{x_1\overline{x_2}}}$. Since $x_1 \in \{0,1\}$ and $x_2 \in \{0,1\}$ this equation is *numerically* equal to $x_1 + x_2 - 2x_1x_2$.

If we add a third cell, 3, with assignment variable $x_3 = 1$ if cell 3 is assigned to cell 1, 0 otherwise, the *logical* expression for the pins required on chip 1 is $\overline{\overline{\overline{x_1}x_2} \cdot \overline{x_1\overline{x_3}} \cdot \overline{\overline{x_2}x_3}}$ and is *numerically* equal to $x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3$. In general the number of pins required on a chip for nets consisting of cells only is

$$\sum_{\forall r \in S} \Big(\sum_{i=1}^{|M_r|-1} (-1)^{i+1} C_i^{M_{rc}} - 2F \prod_{j=1}^{|M_r|} x_j \Big) \quad c = 1, \ldots, k \tag{3}$$

where S is the set of all nets without any I/O cells, M_r is the set of cells on net r, M_{rc} is the set of assignment variables of cells on net r such that $x_{j+c(n-1)} \in M_{rc}$ and $j \in M_r$, $C_i^{M_{rc}}$ is the combinations of the set M_{rc} taken i at a time, and F equals 1 if $|M_r|$ is even, 0 otherwise.

Next, we will derive an expression for a net with a mixture of I/O elements and cells. Consider the 2 cell net described above with the addition of an I/O element. The exact *logical* expression for the number of pins required on chip 1 is $x_1 + x_2$. Using DeMorgans theorem this expression becomes $\overline{\overline{x_1}\,\overline{x_2}}$. *Numerically*, this expression is equal to $x_1 + x_2 - x_1x_2$.

If we add a third cell onto this net, as described above, the exact logical expression for the pins required on this net is $\overline{\overline{x_1}\,\overline{x_2}\,\overline{x_3}}$. *Numerically*, this expression is equal to $x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3 + x_1x_2x_3$. In general, the number of pins required on a chip for nets containing a mixture of I/O elements and cells is

$$\sum_{\forall r \in D} \sum_{i=1}^{|Q_r|} (-1)^{i+1} C_i^{Q_{rc}} \quad c = 1, \ldots, k \tag{4}$$

where D is the set of all nets containing a mixture of I/O elements and cells, Q_r is the set of non I/O elements on net r, Q_{rc} is the set of assignment variables of cells on net r such that $x_{j+c(n-1)} \in Q_{rc}$ and $j \in Q_r$, and $C_i^{Q_{rc}}$ is the combinations of the set Q_{rc} taken i at a time. The pin constraints are expressed for all k chips from equations 3 and 4.

2.1.2 Area Constraints

Let a_j, $j = 0, \ldots, n-1$ be the area of cell j. For k-way partitioning the k area constraints will be

$$\sum_{j=0}^{n-1} a_j x_{j+n(c-1)} \leq A_c \quad c = 1, \ldots, k. \tag{5}$$

where A_c is the area of package c.

2.1.3 Assignment Constraints

We ensure that each component is only assigned to one partition thru n assignment constraints. The general form of these constraints are

$$\sum_{c=1}^{k} x_{j+n(c-1)} = 1 \quad j = 0, \ldots, n-1 \tag{6}$$

2.2 Incorporating Timing Constraints

In addition to minimizing the cutset subject to pin and area constraints as described above, we now consider the timing constraints. In order to formulate timing constraints, we consider a set of *critical* paths. In practice such a constraint can be user defined. However, for our solution, we evaluate the first T longest paths in the given circuit. The T longest paths are found using Kundu's longest path algorithm [2]. This algorithm performs a levelized forward traversal of nodes with a merge sort of delay values, followed by a backward trace to identify T longest paths. All output cells are connected to a *pseudonode* for this purpose. The delay values on each edge is dependent on three factors: fanout from the source cell, delay of the source cell, and type of the source and destination cell. The source and destination cell can be an input, output, or internal cell.

Table 1 illustrates the determination of delay values where $delay_j$ is the delay of internal or I/O cell j, o_j is the fanout to output cells, i_j is the fanout to internal cells, β is the delay due to driving an output cell, μ is the delay due to driving an internal cell, and C is the timing penalty for an edge leaving the chip.

Table 1: Delay Values

Source	Sink	Edge Delay
Input	Output	$delay_j + o_j\beta + i_j\mu$
Input	Internal	$delay_j + o_j\beta + i_j\mu + C$
Internal	Output	$delay_j + o_i\beta + i_j\mu + C$
Internal	Internal	$delay_j + o_i\beta + i_j\mu$+(2C if edge cut, 0 otherwise)

As can be seen from this table, the only variable in the critical path delay is the cutset of internal edges on the T critical paths. Let x_{source} and x_{sink} be the assignment of internal source and sink cells. The timing penalty for an edge between the *source* and *sink* being cut is $2C(x_{source} + x_{sink} - 2x_{source}x_{sink})$. In general, the T'th timing constraint for all k partitions is

$$D_T + \sum_{\forall(i,j)\in E_T} 2C(x_{i+n(c-1)} + x_{(i+1)+n(c-1)} - 2x_{i+n(c-1)}x_{(i+1)+n(c-1)}) \leq Time_T \quad c = 1, \ldots, k \tag{7}$$

where D_T is the delay on critical path T if no edges are cut, E_T is the ordered set of edges traversed containing non I/O cells on critical path T and $Time_T$ is the maximum delay allowed on critical path T. Obviously, if $D_T + 2C > Time_T$ no edge on critical path T can be cut. We constrain T critical paths on each of the k chips for $k \cdot T$ timing constraints.

3 Experimental Results

All code is written in C++ and fortran and compiled using g++ and f77, respectively. MINOS is written in fortran. All benchmarks were tested on an UltraSparc with 512 MB of RAM.

We partition six RT level benchmarks generated from behavioral VHDL descriptions using the high level synthesis system DSS[4]. These benchmarks represent the structure of six large circuits whose characteristics are detailed in[1]. We consider the ten most critical paths for all benchmarks. The characteristics of these benchmarks are in Table 2. The last column of Table 2 shows the area constraint used for each benchmark which is $\frac{\sum_{i=0}^{n-1} a_i}{2} \cdot 1.05$ which results in a minimum k of 2.

Tables 3 - 8 show the results of partitioning under pin and area constraints only. Tables 9 - 14 show the results of considering timing constraints at each k vs P data point in Tables 3 - 8. In Tables 9 and 14 there are three runs that did not complete and contain dashes in the results columns.

The columns headings in Table 2 - 14 are:

- Benchmark - The name of the benchmark circuit
- Total Area - Combined area of cells in the benchmark in square microns
- Number Cells - Total number of cells in the benchmark
- Number Nets - Total number of nets in the benchmark
- Area Const. - Area constraint considered by MINOS
- Pin Const. - Pin constraint considered by MINOS
- k Run - Number of partitions considered by MINOS
- k Actual - Number of partitions actually required (May be different than k Run)
- Run Time - User + system CPU time in seconds required by MINOS to solve the problem.
- Cutsize - Cutset size of the hypergraph representation after rounding
- Cut Run - Number of cuts allowed on each critical path
- Cuts on Critical Path . . . - Cuts on each of 10 critical paths

Table 2: Benchmark Characteristics

Bench Mark	Total Area	Number Cells	Number Nets	Area Const
TLC	2206942	33	93	1158645
decompress	2972054	35	164	1560328
compress	3267322	37	186	1715344
find	7858374	60	285	4125646
fifo	20628509	51	584	10829967
viper	25471959	81	792	13372778

3.1 Analysis

Table 15 illustrates the average impact of incorporating timing constraints on the cutset size and runtime of each benchmark over all runs. Column 2 is the average percent increase in the run time from the tests only considering area and pins. Column 3 is the average percent increase in the cutset size(cutsize) from the tests only considering area and pins. Columns four thru thirteen

Table 3: k vs P Results for TLC

Pin Const.	k Run	k Actual	Run Time	Cut Size	Cuts on Critical Path ... 1	2	3	4	5	6	7	8	9	10
40	2	2	1.3	10	2	2	2	2	2	2	3	3	3	3
35	3	3	4.8	19	1	1	1	1	1	1	1	1	1	1
30	3	3	1.9	18	2	2	2	2	2	2	2	2	2	2
25	5	5	29.2	26	2	2	2	2	2	2	3	3	3	3
20	17	8	98	41	2	2	2	2	2	2	3	3	3	3

Table 4: k vs P Results for decompress

Pin Const.	k Run	k Actual	Run Time	Cut Size	Cuts on Critical Path ... 1	2	3	4	5	6	7	8	9	10
45	2	2	1.0	10	0	0	1	1	1	1	1	1	1	1
40	3	3	2.3	13	0	0	1	1	1	1	1	1	1	1
35	3	3	3.4	16	0	0	1	1	1	1	1	1	1	1
30	4	4	4.5	19	0	0	1	1	1	1	1	1	1	1
25	4	4	3.7	17	0	0	1	1	1	1	1	1	1	1

give the average percent decrease in the cuts on each critical path from the tests not considering timing.

The average run time penalty ranges from a high of 234% to a low of 16% and an average of 75% while the average cutsize penalty ranged from a high of 35% to a *decrease* in the cutsize of 13% and an average of 13%. Intuitively, one would not expect that the consideration of $k \cdot T$ extra constraints would cause the cutsize to decrease. When dealing with non-linear optimization functions and constraints it is quite possible for the NLP tool to stop in a local minimum. Different constraints and optimization functions produce different search directions and therefore different local minima.

The average decrease in the cuts on the ten critical paths ranged from a low of 0 to a high of 75% and on average the number of cuts on a critical path was reduced by 38%. Recall that the number of cuts on each critical path was constrained by a minimum pin constraint previously

Table 5: k vs P Results for compress

Pin Const.	k Run	k Actual	Run Time	Cut Size	Cuts on Critical Path ... 1	2	3	4	5	6	7	8	9	10
40	2	2	2.5	11	1	1	1	1	1	1	1	1	1	1
35	3	3	2.4	16	1	1	1	1	1	1	1	1	1	1
30	3	3	2.8	17	1	1	1	1	1	1	1	1	1	1
25	5	4	9.2	18	2	2	2	2	2	2	2	2	2	2

Table 6: k vs P Results for find

Pin Const.	k Run	k Actual	Run Time	Cut Size	Cuts on Critical Path ... 1	2	3	4	5	6	7	8	9	10
90	2	2	9.9	38	4	4	4	4	4	4	4	4	4	4
85	4	4	26.9	96	5	5	5	5	5	5	5	5	5	5
80	5	5	63.0	125	7	7	7	7	7	7	7	7	7	7
75	4	4	45.6	58	4	4	4	4	4	4	4	4	4	4
70	5	5	59.8	114	8	8	8	8	8	8	8	8	8	8
65	5	5	95.4	117	9	9	9	9	9	9	9	9	9	9
60	6	6	129.7	120	9	9	9	9	9	9	9	9	9	9
55	8	7	213.1	146	8	8	8	8	8	8	8	8	8	8
50	9	8	191	150	8	8	8	8	8	8	8	8	8	8

Table 7: k vs P Results for fifo

Pin Const.	k Run	k Actual	Run Time	Cut Size	Cuts on Critical Path ... 1	2	3	4	5	6	7	8	9	10
150	2	2	4.4	70	3	3	3	3	3	3	3	3	3	3
145	3	3	8.5	123	3	3	3	3	3	3	3	3	3	3
140	3	3	11.5	79	3	3	3	3	3	3	3	3	3	3
135	9	4	49.4	163	7	7	7	7	7	7	7	7	7	7
130	3	3	6.1	128	4	4	4	4	4	4	4	4	4	4
125	6	5	46	118	5	5	5	5	5	5	5	5	5	5
120	6	5	35.9	215	10	10	10	10	10	10	10	10	10	10
115	3	3	9.2	90	4	4	4	4	4	4	4	4	4	4
110	8	6	121.1	250	11	11	11	11	11	11	11	11	11	11
105	8	7	91.5	173	5	5	5	5	5	5	5	5	5	5

found. It is expected that for a given k, and a relaxed pin constraint a partition solution with lower delay would be found.

4 Concluding Remarks

In this paper we have presented a methodology that can be used for effective partitioning of circuits by taking multiple constraints into account. In general, partitioning with multiple constraints is solved by lumping cost parameters such as area, timing, power, and more into one multi-variable function. This has a tendency of not producing designs that can meet the required constraints. We have presented test results for a variety of large real circuits when taking area, pin, and timing costs into consideration. In general we have observed that our methods are fairly compute intensive and partitioning at gate level networks is not a preferred recommendation. However, partitioning using our techniques at RT level of design may be very effective as the size of a circuit's netlist is fairly small. Also, early design decisions in the higher levels of design abstraction are always preferred. Another effective method would be to form clusters on

Table 8: k vs P Results for viper

Pin Const.	k Run	k Actual	Run Time	Cut Size	Cuts on Critical Path . . .									
					1	2	3	4	5	6	7	8	9	10
230	2	2	27.9	139	5	5	5	5	5	5	5	5	5	5
225	4	2	48.3	142	6	6	6	6	6	6	6	6	6	6
220	2	2	18.4	114	4	4	4	4	4	4	4	4	4	4
215	4	4	225	231	4	4	4	4	4	4	4	4	4	4
210	2	2	38.2	123	4	4	4	4	4	4	4	4	4	4
205	4	3	68.6	194	8	8	8	8	8	8	8	8	8	8
200	3	3	23.9	230	4	4	4	4	4	4	4	4	4	4
195	4	4	106.7	254	5	5	5	5	5	5	5	5	5	5
190	3	3	25.4	193	2	2	2	2	2	2	2	2	2	2
185	4	4	43.6	261	10	10	10	10	10	10	10	10	10	10
180	3	3	17.0	200	6	6	6	6	6	6	6	6	6	6
175	5	5	75	361	12	12	12	12	12	12	12	12	12	12
170	4	4	129.7	229	4	4	4	4	4	4	4	4	4	4
165	7	7	630	508	12	12	12	12	12	12	12	12	12	12
160	13	5	665	307	11	11	11	11	11	11	11	11	11	11
155	3	3	7.2	178	7	7	7	7	7	7	7	7	7	7
150	8	6	348	391	10	10	10	10	10	10	10	10	10	10
145	12	8	1441	440	9	9	9	9	9	9	9	9	9	9
140	13	8	1817	416	9	9	9	9	9	9	9	9	9	9

Table 9: Results for TLC with Timing

Pin Const.	k Run	k Actual	Run Time	Cut Size	Cut Run	Cuts on Critical Path . . .									
						1	2	3	4	5	6	7	8	9	10
40	2	2	2.9	8	0	0	0	0	0	0	0	0	0	0	0
35	3	3	16.3	16	2	0	0	0	0	0	0	0	0	0	0
30	3	3	9.6	18	3	1	1	1	1	1	1	1	1	1	1
25	5	-	-	-	20	-	-	-	-	-	-	-	-	-	-
20	17	7	262	34	7	2	2	2	2	2	2	3	3	3	3

Table 10: Results for Decompress with Timing

Pin	k	k	Run	Cut	Cut	Cuts on Critical Path ...									
Const.	Run	Actual	Time	Size	Run	1	2	3	4	5	6	7	8	9	10
45	2	2	1.6	6	0	0	0	0	0	0	0	0	0	0	0
40	3	3	3.6	30	0	0	0	0	0	0	0	0	0	0	0
35	3	3	1.4	21	1	1	1	1	1	1	1	1	1	1	1
30	4	4	4.7	20	1	1	1	1	1	1	1	1	1	1	1
25	4	4	4.3	25	1	1	1	1	1	1	1	1	1	1	1

Table 11: Results for Compress with Timing

Pin	k	k	Run	Cut	Cut	Cuts on Critical Path ...									
Const.	Run	Actual	Time	Size	Run	1	2	3	4	5	6	7	8	9	10
40	2	2	2.8	11	1	1	1	1	1	1	1	1	1	1	1
35	3	3	1.8	17	0	0	0	0	0	0	0	0	0	0	0
30	3	3	4.9	17	3	1	1	1	1	1	1	1	1	1	1
25	5	4	15.2	20	2	1	1	1	1	1	1	1	1	1	1

Table 12: Results for Find with Timing

Pin	k	k	Run	Cut	Cut	Cuts on Critical Path ...									
Const.	Run	Actual	Time	Size	Run	1	2	3	4	5	6	7	8	9	10
90	2	2	17.5	139	5	2	2	2	2	2	2	2	2	2	2
85	4	4	56.1	77	4	4	4	4	4	4	4	4	4	4	4
80	5	4	87.8	78	4	4	4	4	4	4	4	4	4	4	4
75	4	4	52.4	108	4	4	4	4	4	4	4	4	4	4	4
70	5	4	71.7	99	3	3	3	3	3	3	3	3	3	3	3
65	5	5	79.4	115	4	4	4	4	4	4	4	4	4	4	4
60	6	6	180.1	126	4	4	4	4	4	4	4	4	4	4	4
55	8	7	407.8	122	4	4	4	4	4	4	4	4	4	4	4
50	9	9	238	156	5	5	5	5	5	5	5	5	5	5	5

Table 13: Results for Fifo with Timing

Pin Const.	k Run	k Actual	Run Time	Cut Size	Cut Run	Cuts on Critical Path ...									
						1	2	3	4	5	6	7	8	9	10
150	2	2	2.0	75	4	3	3	3	3	3	3	3	3	3	3
145	3	3	11.4	115	6	3	3	3	3	3	3	3	3	3	3
140	3	3	7.0	129	3	3	3	3	3	3	3	3	3	3	3
135	9	5	76	167	6	6	6	6	6	6	6	6	6	6	6
130	3	3	5.3	126	2	2	2	2	2	2	2	2	2	2	2
125	6	4	31.3	159	2	2	2	2	2	2	2	2	2	2	2
120	6	5	80.9	162	5	5	5	5	5	5	5	5	5	5	5
115	3	3	9.8	94	5	4	4	4	4	4	4	4	4	4	4
110	8	5	97.1	130	8	6	6	6	6	6	6	6	6	6	6
105	8	4	205	112	6	4	4	4	4	4	4	4	4	4	4

Table 14: Results for Viper with Timing

Pin Const.	k Run	k Actual	Run Time	Cut Size	Cut Run	Cuts on Critical Path ...									
						1	2	3	4	5	6	7	8	9	10
230	2	2	29.1	125	5	3	3	3	3	3	3	3	3	3	3
225	4	3	118.4	235	6	5	5	5	5	5	5	5	5	5	5
220	2	2	19.1	123	5	4	4	4	4	4	4	4	4	4	4
215	4	4	102.5	254	6	6	6	6	6	6	6	6	6	6	6
210	2	2	15.4	124	10	4	4	4	4	4	4	4	4	4	4
205	4	4	227.9	323	5	5	5	5	5	5	5	5	5	5	5
200	3	3	48	148	6	2	2	2	2	2	2	2	2	2	2
195	4	4	175.8	304	7	5	5	5	5	5	5	5	5	5	5
190	3	3	86.7	159	4	3	3	3	3	3	3	3	3	3	3
185	4	4	87.1	284	8	7	7	7	7	7	7	7	7	7	7
180	3	3	60.5	143	5	3	3	3	3	3	3	3	3	3	3
175	5	5	268.1	320	6	6	6	6	6	6	6	6	6	6	6
170	4	4	648.9	279	10	8	8	8	8	8	8	8	8	8	8
165	7	7	57.3	493	6	6	6	6	6	6	6	6	6	6	6
160	13	7	1947.6	425	7	7	7	7	7	7	7	7	7	7	7
155	3	-	-	-	20	-	-	-	-	-	-	-	-	-	-
150	8	7	348	391	1	10	10	10	10	10	10	10	10	10	10
145	12	12	1441	440	1	9	9	9	9	9	9	9	9	9	9
140	13	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 15: Impact on Run Time and Cutsize when Considering Timing

Bench Mark	% Increase Run Time	% Increase Cutsize	% Decrease Cuts on Critical Path ...									
			1	2	3	4	5	6	7	8	9	10
TLC	234	(13)	63	63	63	63	63	63	75	75	75	75
decompress	16	35	0	0	40	40	40	40	40	40	40	40
compress	32	14	38	38	38	38	38	38	38	38	38	38
find	42	30	42	42	42	42	42	42	42	42	42	42
fifo	19	(1)	24	24	24	24	24	24	24	24	24	24
viper	108	13	16	16	16	16	16	16	16	16	16	16

a gate level design. The clusters can then be considered as *supernodes* and an NLP formulation would solve fairly quickly.

Our on going work includes addressing the problem of hierarchical partitioning (when multiple constraints like area, pin, cost, timing are very important for designing VLSI Systems), and exploring methods for guiding NLP solver to obtain better constraint satisfying local minimas, perhaps close to global minimas.

References

[1] N. Kumar. *High Level VLSI Synthesis for Multichip Design*. PhD thesis, University of Cincinnati, 1994.

[2] S. Kundu. An incremental algorithm for identification of longest(shortest) paths. *INTEGRATION, the VLSI Journal*, 17:25–31, 1994.

[3] B. Preas and M. Lorenzetti. *Physical Design Automation of VLSI Systems*. The Benjamin/Cummings Publishing Company, Inc., 1988.

[4] J. Roy, N. Kumar, R. Dutta, and R. Vemuri. DSS: a distributed high-level synthesis system. *IEEE Design and Test of Computers*, pages 18–32, June 1992.

[5] G. Tumbush and D. Bhatia. Partitioning under timing and area constraints. In *To appear in Proceedings of the IEEE 1997 International Conference on Computer Design*, October 1997. to appear.