Synthesis of Interconnection Structures for Multi-Chip Designs

Yung-Hua Hung and Alice Parker

CENG Technical Report 92-02

Department of Electrical Engineering – Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-4476

Synthesis of Interconnection Structures for Multi-Chip Designs

Abstract

Many digital systems are implemented with multiple chips. However, little work has been reported on the synthesis of multi-chip designs. Due to the complexity, the problem of synthesizing multi-chip designs is divided into two subproblems: (1) interchip connection construction, and (2) scheduling. The problem can be approached in either order: interchip connection determined before or after scheduling. The former approach has already been reported by us. In this paper, we describe a technique to determine interchip connection after scheduling. The problem of determining the interchip connection can be modeled as a maximum-gain clique partitioning problem. A heuristic procedure, which is based on the special properties of the compatibility graph, has been developed. It consists of repeatedly applying weighted matching on bipartite subgraphs. The experimental results are reported and compared to our previous approach.

1 Introduction

Modern digital systems usually contain multiple chips. In designs composed of multiple chips, interchip communication often affects performance and should be considered at an early stage of the design process. However, most current high-level synthesis techniques (e.g. [PP88, PK89, PK91, HHL91]) assume the design is to be implemented with a single chip, or assume unlimited interconnection resources.

Due to the complexity, the problem of synthesizing multi-chip designs is divided into two subproblems: (1) interchip connection construction, and (2) scheduling. The problem can be approached in either order: interchip connection determined before or after scheduling. Previously [HP92], we approached multiple-chip pipeline synthesis by constructing interchip connection before scheduling. In our earlier approach, the interchip connection is constructed, given the I/O pin constraints, with the goal of maximizing communication bandwidth among chips. Then, the design is scheduled, based on the interchip connection.

In this paper, we describe a technique to synthesizing the interchip connection after scheduling of a pipelined design has been done. Also, the values to be transferred are assigned to the communication buses in the process of constructing the interchip connection. We then compare experimental results to the previous approach.

2 Related Research

DeMicheli et al. [KDM90, DeM90] applied an iterative approach consisting of three stages, namely resource binding, scheduling, and partitioning. In their approach, resource binding is performed before scheduling. In the binding stage, operations are bound to specific resources. There will be a large number of binding configurations. In each iteration, one of the binding configurations is selected, either by an exhaustive search or a heuristic search with some cost criteria. Once a resource binding is selected, the operations bound to the same resource are serialized to resolve resource conflicts. A branch-and-bound approach is used to explore the alternatives. Then, a SIF model, a variation of a control data flow graph (CDFG) containing binding and serialization information, is partitioned into multiple chips. Their system can handle only non-pipelined designs. Pin sharing among I/O operations was not reported, as the pin cost, which is used as one of the constraints in the process of partitioning, is computed by simply adding the costs of all I/O operations in the partition.

3 Problem Statement

We assume each design has been already been partitioned onto multiple chips. A design is first scheduled, using pipelining where possible to meet constraints. After the pipelined design has been scheduled, the interchip connections are determined with the objective of minimizing the number of I/O pins required. The interchip connections can be used for data transfers between chips for the given schedule without communication conflicts.

The bus structure shown in Figure 1 will be used as our interchip connection model. In the figure, $r_{i,h}$ is the number of pins of an I/O port of chip P_i connected to bus C_h. In our model, each chip can be connected to a number of communication buses through bidirectional I/O ports. Also, each bus can be connected to more than one chip. The width of an I/O port of a chip connected to a bus is not necessarily the same as the widths of I/O ports of other chips connected to the bus.

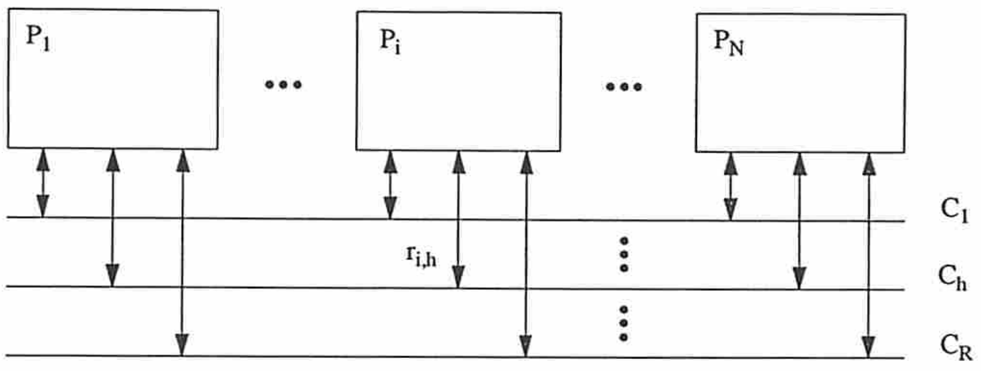

Figure 1: Interchip connection model

4 Problem Assumptions

It is assumed that I/O transfers are activated at the beginning of a clock cycle, and must be completed in a single cycle. When two minor clocks are used, I/O transfers can be activated at the beginning of a minor clock cycle and must be completed in the same minor cycle. It is also assumed that every communication bus can only be used to transfer at most one value in one control step. That is, values with smaller bit width are not grouped and transferred on a communication bus with a larger bit width. Also, values have to be transferred as a whole[1].

[1]This assumption is being relaxed in ongoing research.

5 Synthesis Approach

5.1 Scheduling

Force-Directed Scheduling [PK89] is used to schedule a CDFG that has been partitioned into a number of partitions, each of which will be implemented in a chip, given an initiation rate and pipe length. All partitions are scheduled at the same time because of the interdependency among partitions imposed by the constraints that output and inputs of each value between partitions must take place in the same cycle. FDS tries to minimize the number of hardware resources required by balancing the utilization of the resources. However, the force of assigning an I/O operation in a control step is very difficult to compute accurately because the interchip connection is not known *a priori*. Here, the combination of the forces of the corresponding input operation and output operation is used as the force of assigning an I/O operation in a control step.

5.2 Determining Interchip Connections

In pipelined designs with an initiation rate of L, an execution instance will be initiated before the previous execution instance is completed. So, the operations scheduled in control steps $k, k+L, k+2L\dots$ are executed overlapped, and cannot share resources. The set of control steps $k, k+L, k+2L\dots$ will be referred to as *control step group* k. The operations scheduled in different control step groups will never be executed concurrently, and can share the same hardware resources. Two I/O operations $w_1 = (P_{i_1}, P_{j_1})$, and $w_2 = (P_{i_2}, P_{j_2})$ are said to be *compatible* if and only if they can share a communication bus. The I/O operations are said to be in *conflict* if they are not compatible. The I/O operations which are scheduled in different control step groups can share a communication bus, and so are compatible. I/O operations which are scheduled in the same control step group can share a communication bus only if they transfer the same value in the same control step.

After scheduling, each I/O operation has been assigned to one of the control step groups. Then, the interchip connection can be determined with the goal of minimizing the number of I/O pins required. The problem can be modeled as a compatibility graph, in which each node represents an I/O operation. There is an edge between two nodes if and only if their corresponding I/O operations are compatible. On each edge is an associated weight, which represents the number of I/O pins which can be shared, and so can be saved if the I/O operations corresponding to these two end nodes are assigned to a shared communication bus. The weight, $weight(w_1, w_2)$, on the edge between w_1 and w_2 corresponding to two compatible I/O operations is given by

$$weight(w_1, w_2) = \begin{cases} (wf_i + wf_j)\min(B_{w1}, B_{w2}) & \text{if } i_1 = i_2 \wedge j_1 = j_2 \\ wf_i \min(B_{w1}, B_{w2}) & \text{if } i_1 = i_2 \wedge j_1 \neq j_2 \\ wf_j \min(B_{w1}, B_{w2}) & \text{if } i_1 \neq i_2 \wedge j_1 = j_2 \\ 0 & \text{otherwise} \end{cases}$$

where wf_i is a weighting factor used to specify the importance to share I/O pins of partition P_i, and B_{w1}, and B_{w2} are the bit widths of I/O operations $w1$, and $w2$, respectively. The minimum of two bit widths is taken because it indicates the number of pins which can be saved if the two I/O operations share a communication bus. The minimization of I/O pins for different partitions can compete with each other. The partition with a higher weighting factor will be given priority over the one with a lower value. Note that an edge can have a zero weight, and it is quite different from no edge at all since the I/O operations corresponding to the two nodes with an edge with a zero weight connected to them can share a communication bus even if they do not share I/O pins.

For the case of bidirectional I/O ports, in which an I/O port can be used as either a source or destination of an I/O transfer, both I/O operations $w = (P_i, P_j)$ and $w' = (P_j, P_i)$ require the same communication path on the communication bus if they are assigned to the same communication bus.

The problem of minimizing the number of I/O pins can be reduced to the problem of partitioning the compatibility graph into a number of disjoint cliques having the largest gain, where the gain is the summation of the weights on all edges that are contained in the cliques. The nodes in a clique represent the fact that the corresponding I/O operations are assigned to the same communication bus. The number of communication buses is equal to the number of cliques partitioned. The structure of a communication bus (the I/O ports of which chips are connected to the bus) can be easily determined from the assignment of I/O operations to the bus since there must be a communication path with a sufficient bit width for each I/O operation assigned to the bus.

The problem is similar to the clique partitioning problem, which had initially been used by Tseng and Siewiorek [TS83] to model the problem of operator, register, and in-chip connection binding, and which is widely used for synthesis. The general clique partitioning problem has been known to be NP-hard. For *interval graphs*, which represent the intersection between a set of intervals along the real line, the left edge algorithm, which is a polynomial time algorithm, can be used to partition the graph into a minimum numbers of cliques. Springer [Spr91] has shown that the least-cost clique partitioning problem is NP-hard even for the interval graph. We believe that the problem of determining the interchip connection with the goal of minimizing the number of I/O pins required is also an NP-hard problem.

The compatibility graph of our problem has some special properties. The graph can be divided into L groups $G_0, G_1, \ldots, G_{L-1}$. Group G_k contains the nodes which correspond to the I/O operations scheduled in control step group k. So, there is an edge between any node in G_i and any node in G_j (for $i \neq j$). Group G_k can be further divided into a number of subgroups $SG_{k,1}, SG_{k,2}, \ldots, SG_{k,n_k}$. The nodes in subgroup $SG_{k,j}$ correspond to the I/O operations which transfer the same value at the same control step. Each subgroup usually contains only one node. There is an edge between any two nodes in a subgroup, while there is no edge between any two nodes in two different subgroups in the same group. Finding a set of cliques is equivalent to finding a set of disjoint subsets, S_k, in which there can be more than one node from the same group only if all of the nodes from the same group are in the same subgroup. By taking the advantage of the above properties, the heuristic procedure shown in Figure 2 has been developed. At first, the nodes corresponding to the I/O operations assigned in the same control step group are grouped. The nodes in each group are divided into subgroups depending on whether the corresponding I/O operations transfer the same value in the same control step. The nodes in a subgroup are combined to form a supernode. After these steps, the compatibility graph becomes an interval graph, and there are only edges between nodes in different groups. The graph induced from any two groups is bipartite. The cliques can be constructed by applying a series of bipartite weighted matchings to two groups giving priority to larger groups. After the matching between the two groups is found, A new group will be created by combining these two groups according the matching, and replaces them. The Hungarian algorithm, which has a complexity of $O(n^3)$, has been used for the problem of bipartite weighted matching. The worse case complexity is $O(Ln^3)$, where n is the total number of I/O operations. For the cases in which the

```
**Determine Interchip Connection**
For each group $G_k$ do
    Divide $G_k$ into $SG_{k,1}, \ldots, SG_{k,n_k}$
Order $G_i$ such that $n_i \geq n_j$ for $i < j$
For $i = 1$ to $L - 1$ do
    Apply Hungarian algorithm to $G_0$ and $G_i$
    For $j = 1$ to $n_0$ do
        $SG_{0,j} = SG_{0,j} \cup SG_{i,match(j)}$
The nodes in $SG_{0,j}$ denotes that the corresponding I/O operations are assigned to bus $j$
```

Figure 2: A heuristic procedure for constructing interchip connection

I/O operations are evenly distributed among control step groups, the complexity will be $O(n^3/L^2)$.

6 Experimental Results and Summaries

6.1 AR filter

The AR lattice filter element [Kun84] is partitioned as shown in Figure 3, in the same way as in [HP92]. The values transferred across chips have a variety of bit widths. The number next to the I/O operation is the bit width. The I/O operations without numbers next to them have 8 bits. The stage time is 250 ns. Inputs arrive every 3 cycles. I/O operations take 10 ns. Additions are performed by adders with 30 ns. delay. Multiplications are performed by multipliers with 210 ns. delay. Chained operations are allowed.

Table 1 shows the numbers of I/O pins and operators required with variations of initiation rates and pipe lengths. For a given initiation rate, increasing the pipe length does not necessarily lead to a lower hardware requirement. For a given initiation rate, a design with least resource requirements was chosen from the designs with different pipe lengths. The results produced by our earlier approach are shown in Table 2 using the above least resource requirements as resource constraints. Earlier results for the designs with lower resource constraints are also shown in the table.

6.2 Elliptic filter

The fifth-order wave digital elliptic filter [KWK85] is partitioned as shown in Figure 4, and all values are 16 bits wide. I/O operations, addition, and multiplication take 1, 1, and 2 cycles, respectively. There are a number of data feedback edges with *degree* 1. A degree d on an edge denotes that the value produced by the source operation will be consumed by the destination operation d iterations (execution instances) later. The initiation rate for the filter must be at least 19 cycles, because the length of the critical loop (X33, +2, Xf, +5, Xe, ..., Xh, ..., Xj, ..., +28) is 19 cycles. In order to show the sharing of multiple buses and let more I/O operations execute concurrently, the degree on each data feedback edge has been modified to 4. Thus, the design can operate on four independent multiplexed streams of data. So, the minimum initiation rate becomes 5 cycles.

Table 3 shows the numbers of I/O pins and operators required with variations of initiation rates and pipe lengths. The results produced using our earlier approach are shown in Table 4. Our earlier approach could not produce any schedule for

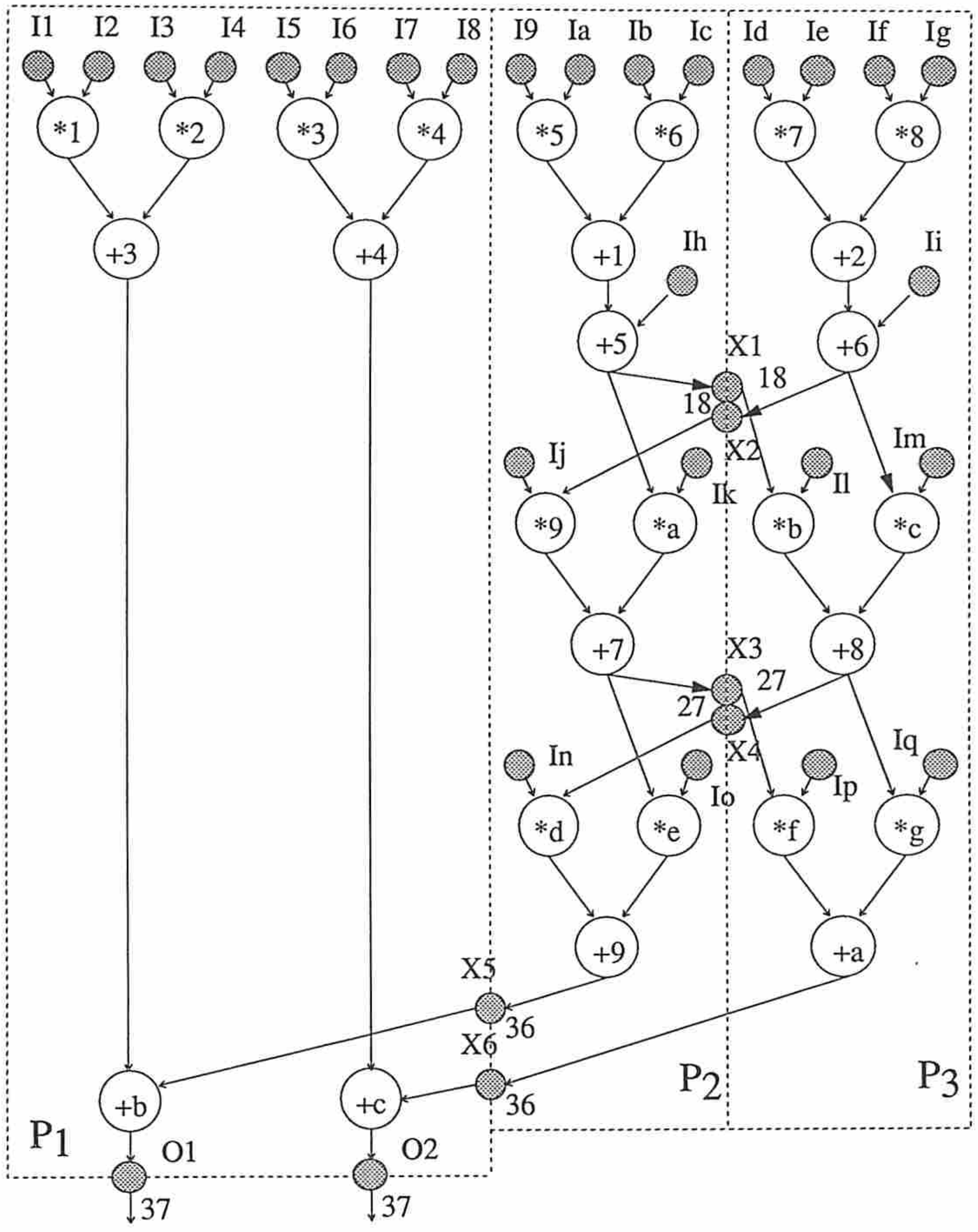

Figure 3: A partitioned AR filter

Inputs		Outputs								
Initiation	Pipe	#I/O pins			#Adders			#Multipliers		
Rate	Length	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3
3	6	162	95	95	2	2	2	2	4	4
	7	90	130	114	2	2	2	2	2	2
	8	90	114	114	2	2	2	2	3	2
	9	**91**	**105**	**113**	**2**	**2**	**2**	**2**	**2**	**3**
	10	126	105	105	2	2	2	2	3	2
4	**6**	**98**	**95**	**95**	**2**	**1**	**1**	**2**	**2**	**2**
	7	106	123	107	2	2	2	2	2	2
	8	82	105	105	2	2	2	1	2	2
	9	82	114	114	2	1	2	1	2	2
	10	98	95	95	2	2	1	1	2	2
5	6	82	106	122	2	1	1	2	2	2
	7	82	114	98	2	1	1	2	2	2
	8	**90**	**79**	**97**	**2**	**1**	**1**	**2**	**2**	**2**
	9	90	97	113	2	2	1	2	2	2
	10	90	97	113	2	2	1	2	2	2

Table 1: Resources required for the AR filter with variations of initiation rates and pipe lengths using the technique described in this paper

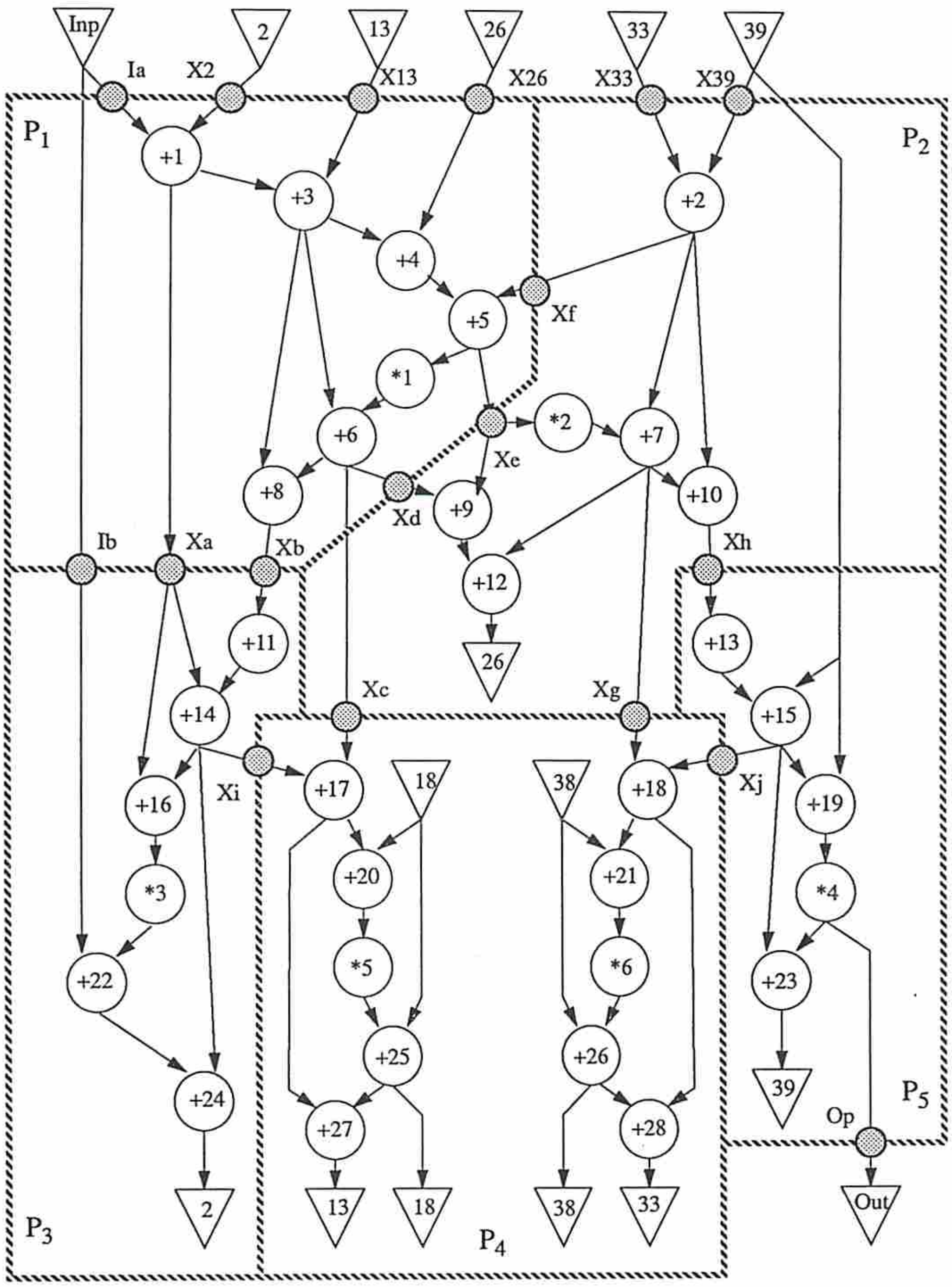

Figure 4: A partitioned elliptic filter

Inputs										Outputs
Initiation Rate	#I/O pins			#Adders			#Multipliers			Pipe Length
	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3	
3	**91**	**105**	**113**	**2**	**2**	**2**	**2**	**2**	**3**	**9**
	89	113	113	2	2	2	2	3	2	11
4	**98**	**95**	**95**	**2**	**1**	**1**	**2**	**2**	**2**	**11**
	53	87	87	1	1	1	1	2	2	12
5	**90**	**79**	**97**	**2**	**1**	**1**	**2**	**2**	**2**	**10**
	81	70	70	1	1	1	1	2	2	10

Table 2: Pipe length for the AR filter with variations of initiation rates and resource constraints using the technique described in [HP92]

several designs with an initiation rate of 5 and tight resource constraints. The reason is because of a slack time of only 1 control step in the critical loop and the greedy nature of the list scheduling technique implemented in our earlier approach.

From the above results, our earlier approach of determining interchip connection before scheduling can produce a schedule for a design having tighter I/O pin constraints. For most of the cases with looser resource constraints, the previous approach will produce a schedule with a longer pipe length. Our earlier approach can produce a schedule for a design with tighter pin constraints because the optimization of I/O pins is done first while in the current approach scheduling puts more constraints on the I/O pin optimization. The force of assigning an I/O operation in a control step needs further study since the combination of the forces of assigning the corresponding input and output operations does not reflect the usage of the communication bus accurately.

Inputs		Outputs														
Init.	Pipe	#I/O pins					#Adders					#Multipliers				
Rate	Length	P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5
5	21	64	48	32	32	48	2	2	2	3	1	1	1	1	2	1
	22	**48**	**32**	**32**	**32**	**32**	**2**	**1**	**2**	**2**	**1**	**1**	**1**	**1**	**1**	**1**
	23	48	32	16	48	32	2	1	2	2	1	1	1	1	1	1
	24	48	48	32	48	16	2	1	2	2	1	1	1	1	1	1
	25	48	32	32	48	32	2	1	2	3	1	1	1	1	1	1
6	21	48	32	16	64	16	2	2	1	2	1	1	1	1	2	1
	22	**32**	**32**	**32**	**32**	**32**	**2**	**1**	**1**	**2**	**1**	**1**	**1**	**1**	**1**	**1**
	23	32	32	32	32	32	2	1	1	2	1	1	1	1	1	1
	24	48	48	32	32	16	2	2	1	2	1	1	1	1	1	1
	25	48	48	32	32	32	1	1	2	2	1	1	1	1	1	1
7	21	48	32	48	32	32	2	1	1	2	1	1	1	1	2	1
	22	**48**	**48**	**32**	**32**	**16**	**2**	**1**	**1**	**2**	**1**	**1**	**1**	**1**	**1**	**1**
	23	48	32	32	32	32	2	1	1	2	1	1	1	1	1	1
	24	32	32	32	48	32	2	1	1	2	1	1	1	1	1	1
	25	32	32	32	48	32	2	1	1	2	1	1	1	1	1	1

Table 3: Resources required for the elliptic filter with variations of initiation rates and pipe lengths using the technique described in this paper

Inputs																Outputs
Init.	#I/O pins					#Adders					#Multipliers					Pipe
Rate	P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5	Leng.
5	**48**	**32**	**32**	**32**	**32**	**2**	**1**	**2**	**2**	**1**	**1**	**1**	**1**	**1**	**1**	-
	48	48	48	48	48	2	2	2	3	2	1	1	1	2	1	-
6	**32**	**32**	**32**	**32**	**32**	**2**	**1**	**1**	**2**	**1**	**1**	**1**	**1**	**1**	**1**	**23**
	32	32	32	32	16	2	1	1	2	1	1	1	1	1	1	23
7	**48**	**48**	**32**	**32**	**16**	**2**	**1**	**1**	**2**	**1**	**1**	**1**	**1**	**1**	**1**	**25**
	32	32	16	32	16	2	1	1	2	1	1	1	1	1	1	27

Table 4: Pipe length for the elliptic filter with variations of initiation rates and resource constraints using the technique described in [HP92]

7 Conclusions

The synthesis of multi-chip designs has been approached in two ways: interchip connection determined both before and after scheduling. From the experimental results, it seems that the most important aspect to optimize (performance or resources) should be addressed first. A better scheduling result might have been obtained if a more advanced scheduling technique rather than list scheduling was used in the earlier technique.

References

[DeM90] G. De Micheli and *et al*, "Partitioning of Functional Models of Synchronous

Digital Systems," *Proceedings of ICCAD*, Nov. 1990.

[HHL91] C. Hwang, Y. Hsu, and Y. Lin, "Scheduling for Functional Pipelining and Loop Winding," *Proc. of the 28th Design Automation Conference*, pp. 764-769, June 1991.

[HP92] Y. Hung and A. C. Parker, "High-Level Synthesis with Pin Constraints for Multiple-Chip Designs," *Proc. of the 29th Design Automation Conference*, June 1992.

[KDM90] D. Ku and G. De Micheli, "High-level Synthesis and Optimization Strategies in Hercules and Hebe," *EURASIC, Proceedings of the European Conference on ASIC design*, May 1990.

[Kun84] S. Y. Kung, "On Supercomputing with Systolic/Wavefront Array Processor," *Proceedings of the IEEE*, pp. 867-884, July 1984.

[KWK85] S. Y. Kung, H. J. Whitehouse, and T. Kaliath, "VLSI and Modern Signal Processing," Prentice-Hall, pp. 257-276, 1985.

[PK89] P. G. Paulin and J. P. Knight, "Force-Directed Scheduling for the Behavioral Synthesis of ASIC's," *IEEE Transactions on Computer-Aided Design*, vol. 8, pp. 661-679, June 1989.

[PK91] I. Park and C. Kyung, "Fast and Near Optimal Scheduling in Automatic Data Path Synthesis," *Proc. of the 28th Design Automation Conference*, pp. 680-685, June 1991.

[PP88] N. Park and A. C. Parker, "Sehwa: A Software Package for Synthesis of Pipelines from Behavioral Specifications," *IEEE Transactions on Computer Aided Design*, vol. 7, pp. 356-370, March 1988.

[Spr91] D. L. Springer, "Coloring and Clique Partitioning for Data Path Allocation," *PhD thesis, Carnegie Mellon University*, May 1991.

[TS83] C. Tseng and D. P. Siewiorek, "Facet: A Procedure for the Automated Synthesis of Digital Systems," *Proc. of the 20th Design Automation Conference*, pp. 490-496, June 1983.