Integrated System Architecture Synthesis of Distributed Embedded Systems for Multimedia Applications

Jin-Lin Liu, Shiann-Rong Kuang, and Shen-Fu Hsiao
Department of Computer Science Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan
jinlin.liu@m2k.com.tw

ABSTRACT

An integer linear programming (ILP) based approach incorporating with dynamic voltage scaling technique for integrated system architecture synthesis of embedded multimedia systems is presented in this paper. The approach maps each computation task of a particular application onto a system component with multiple supply voltages, schedules the execution of these computation tasks and communication time into pipeline stages, and generates communication topology simultaneously. The objective is to minimize the total hardware area and/or total power consumption subject to the throughput constraint on the pipelined system architecture.

1: INTRODUCTIONS

To achieve flexible implementations and satisfy different design requirements such as area, performance, and power consumption, most of complex embedded multimedia systems are implemented by heterogeneous architectures that consist of different types of processing elements (PEs) and communication links [1, 2]. Architecture synthesis of such systems involves various steps such as mapping, scheduling, communication synthesis, and dynamic voltage scaling. All these steps are highly inter-dependent [3]. As a result, developing an integrated approach to cope with these design steps simultaneously is essential and crucial. Besides, multimedia applications typically have intensive computations and an endless stream of input data with throughput constraints. To meet the throughput constraints with a minimum cost, it is necessary to pipeline the system to construct more efficient architectures.

The work in [4] presents an integer programming (IP) models to solve the problems of mapping and scheduling. It can obtain optimal solution but has high complexity. Except for the optimal approaches [4, 5], various heuristic approaches including iterative improvement, constructive method, and genetic algorithm etc. have been used to synthesize system architectures [6-8]. However, all the approaches mentioned so far assume that the computation tasks in the system execute sequentially so that they cannot construct efficient pipelined architecture for multimedia applications. In contrast, [9, 10] have proposed approaches to build the pipelined system architecture. COHRA [11] and COSYN [12] are another two co-synthesis systems that support pipelined scheduling.

Most of the above-mentioned approaches have not focused on trading system's power consumption and performance by dynamic voltage scaling (DVS) technique. Several approaches [13, 14] have demonstrated the efficiency of system architecture synthesis with DVS technique in reducing the power consumption. Our approach is based on a set of ILP formulation that minimizes the total area and/or total power consumption of pipelined system architecture under the throughput constraints. Its ability to consider mapping, pipelined scheduling, communication synthesis, and DVS simultaneously to trade area with power consumption under throughput constraint makes it substantially different from other optimal and heuristic approaches.

The remainder of this paper is organized as follows. Section 2 describes the integrated system architecture synthesis problem of distributed embedded systems for multimedia applications. Section 3 explains our ILP model to solve the integrated problem. Section 4 applies the proposed ILP approach to some examples and discusses the experimental results. Finally we give a conclusion.

2: PROBLEM DESCRIPTION

2.1: TARGET ARCHITECTURE

A distributed embedded system today usually consists of multiple heterogeneous PEs (uPs, DSPs, ASICs, etc.), memory components, communication networks and communication interfaces. For the communication network, on-chip shared bus is adopted in our target architecture. Moreover, as data communication between PEs in the pipelined architecture is usually very frequent, single system bus very likely becomes performance bottleneck. Therefore, we assume that local buses are possibly used to connect PEs directly to avoid communication congestion and higher power consumption in the system bus. In addition, each PE contains a buffer in order to temporarily store the data from/to communication link. For the buffering model, the out-going data can be transferred in the later time after finishing computation and the incoming data can be held before computation. Therefore, the scheduling will be more flexible and the communication will not affect the computation of PEs. An example of target architecture, which consists of four PEs, one system bus, and one local bus between SW1 and HW3, is shown in Fig. 1.

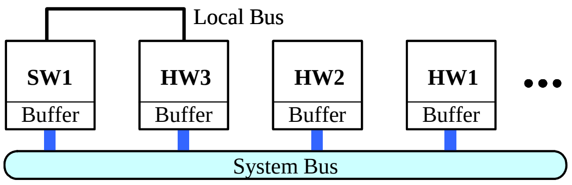

Figure 1: An example of target architecture

2.2: TASK GRAPH

Typically multimedia applications are dominated by dataflow constructs and can be described as a task graph at a coarse level of granularity. A task graph $G(V, E, I, O)$ is a directed acyclic graph, where V denotes a set of functional nodes and E denotes a set of communication edges. Moreover, I and O are dummy nodes called the input and output nodes which are used to model the I/O environment and specify the throughput constraint. $F_i \in V$ is a functional node and $C_n(F_{i1}, F_{i2}) \in E$ is a communication edge from source functional node F_{i1} to destination functional node F_{i2}. For the dependency relation in G, the functional node should be activated after all its input edges are finished. And the output edges cannot be activated until the source functional node is finished. The loop-carried dependency is denoted by a dotted edge with dependence distance. The dotted edge $C_n(F_{i1}, F_{i2})$ with dependence distance D_n implies that the data produced by F_{i1} in the m^{th} iteration is consumed by F_{i2} in the $m+D_n$ iteration. Considering the task graph depicted in Fig. 2, C_2 and C_3 can be activated after F_1 is finished, and F_4 cannot be activated until both C_4 and C_5 are finished. Moreover, $D_7 = 1$ indicates that the data produced by F_3 in the m^{th} iteration is consumed by F_1 in the $m+1$ iteration.

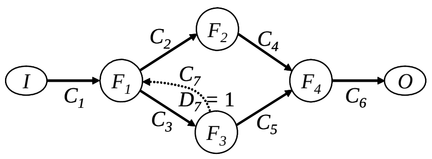

Figure 2: An example of task graph

2.3: PIPELINED SCHEDULING

In the iterative nature of multimedia applications, pipelined scheduling benefits the performance if there are enough resources. Pipelining divides the system into concurrently executing stages, thus increasing its data rate. The throughput of a multimedia system is the rate at which it processes input samples, and this is usually the prime constraint on most multimedia applications. In the proposed approach, pipelined scheduling is useful to design the system architecture with less area and/or power consumption under throughput constraint. The throughput constraint $\mathfrak{J}$ specifies the difference in the arrival time of two consecutive input samples. We also refer to this time as the pipeline stage delay, since this would be the required delay of a pipeline stage in the design. Fig. 3 depicts a two-stage pipelined schedule of the task graph shown in Fig. 2 under throughput constraint $\mathfrak{J} = 120$.

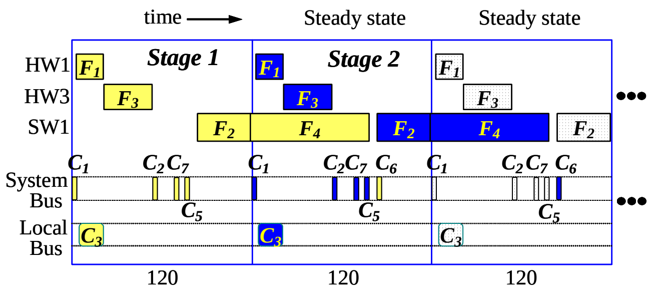

Figure 3: An example of pipelined scheduling

2.4: COMMUNICATION SYNTHESIS

The communication topology has significantly impact on the performance and power consumption of system architecture. There are three communication types for each $C_n(F_{i1}, F_{i2}) \in E$ in our target architecture. First, if F_{i1} and F_{i2} are executed on the same PE, the communication data don't be transferred through bus leading to no extra communication time and power. For example, $C_4(F_2, F_4)$ don't be shown in Fig. 2 because F_2 and F_4 are mapped onto the same PE (SW1). Second, communication data is transferred through a private local bus. The local bus will increase the area of target architecture, but better performance or lower power consumption can be achieved. For example, $C_3(F_1, F_3)$ in Fig. 3 is performed on a private local bus. Third, communication data is transferred through the shared system bus. The pipeline scheduler must avoid the communication collision on the shared system bus.

2.5: DYNAMIC VOLTAGE SCALING

DVS dynamically scales the supply voltage and operational frequency of system's PEs in accordance with the performance requirements of the applications to reduce power consumption. DVS are highly inter-dependent with mapping, pipelined scheduling, and communication synthesis. For example, pipelined scheduling cannot be performed without the execution time of computation tasks, which are available only after PEs and supply voltages are selected. In the target architecture, we assume that each PE can operate at multiple supply voltages. The voltage selection problem must be considered simultaneously to tradeoff area, performance, and power consumption.

Based on the above explanation, the problem of integrated system architecture synthesis for embedded multimedia applications can be described as follows. Given a task graph $G(V, E, I, O)$, a PE library *MLib* with the corresponding implementation information including PE's area, execution time, power consumption etc., and a throughput constraint $\mathfrak{I}$, the problem is to

- map each $F_i \in V$ to a PE instance run at a feasible voltage to execute its function;
- schedule function computation and data communication in pipeline;
- assign each $C_n(F_{i1}, F_{i2}) \in E$ to an internal communication, a local bus, or the shared system bus;

such that

- the throughput constraint $\mathfrak{I}$ is satisfied;
- the total area and/or total power consumption are minimized.

3: ILP FORMULATION

In this section, we develop an ILP formulation to solve the problem of integrated system architecture synthesis. Assume that the types of PEs in *MLib* which can perform the function of F_i are collected in library ML_i. The following notations and base (independent) variables are necessary to describe the ILP formulation.

- MN_j: the number of available PE of type $M_j \in MLib$
- VN_j: the number of available supply voltage for PE of type $M_j \in MLib$
- BN: the number of available local bus
- S_{MAX}: the number of available pipeline stage in the design
- $x_{i,j,k}$: a binary variable associated with functional node $F_i \in V$. $x_{i,j,k} = 1$ if F_i is performed by the kth instance ($1 \le k \le MN_j$) of PE of type $M_j \in ML_i$; otherwise, $x_{i,j,k} = 0$.
- $v_{i,j,k,h}$: a binary variable associated with functional node $F_i \in V$. $v_{i,j,k,h} = 1$ if F_i is performed by the kth instance ($1 \le k \le MN_j$) of PE of type $M_j \in ML_i$ run at voltage $V_{j,h}$; otherwise, $v_{i,j,k,h} = 0$.
- l_n: a binary variable associated with communication edge $C_n(F_{i1}, F_{i2}) \in E$. $l_n = 1$ if the communication data of C_n is transferred through a local bus; otherwise, $l_n = 0$.
- F_i^{ST} and C_n^{ST}: the start time of functional node $F_i \in V$ and communication edge $C_n \in E$

Based on the above notations and independent variables, we define the following dependent variables and then build the constraints of our ILP model.

3.1: PE USAGE VARIABLES

A binary variable $a_{j,k}$ is used to represent whether or not the kth instance of PE of type M_j is selected to execute any functional node at least once. Variable $a_{j,k}$ can be determined as

$$\sum_{F_i \in FM_j} x_{i,j,k} \le A \cdot a_{j,k}, \ \forall\ M_j \in MLib \text{ and } 1 \le k \le MN_j \quad (1)$$

where FM_j is the set of functional nodes that can be implemented by PE of type M_j and A is a constant which is as large as the number of functional nodes. Then, the instance number of PE of type M_j used (denoted as MD_j) can be expressed as

$$MD_j = \sum_{1 \le k \le MN_j} a_{j,k}, \quad \forall\ M_j \in MLib \quad (2)$$

3.2: BUS USAGE VARIABLES

Except for l_n, we introduce two extra variables s_n and y_n associated with communication edge $C_n(F_{i1}, F_{i2})$ to represent the communication type of C_n, where only one type is true. Variable s_n is a 0–1 integer that is 1 if the communication data of C_n is transferred through the system bus, else 0. Another variable y_n is also a 0–1 integer that is 1 if the source and destination functional nodes F_{i1} and F_{i2} of C_n executed on the same PE instance, else 0. $y_n = 1$ represents that the communication data don't be transferred through bus. Let $ML_{i12} = ML_{i1} \cap ML_{i2}$, we have

$$y_n = \sum_{M_j \in ML_{i12}} \sum_{1 \le k \le MN_j} x_{i1,j,k} \wedge x_{i2,j,k}, \quad (3)$$

The AND operation $\wedge$ in (3) can be easily converted to linear form by introducing addition variables and constraints.

In addition, we define another binary variable $r_{j1,k1,j2,k2}$. Variable $r_{j1,k1,j2,k2}$ is 1 if there is a local bus between k1th instance of PE of type M_{j1} and k2th instance of PE of type M_{j2}, else 0. The variable represents a physical local bus in the target architecture, not a functional local link defined as l_n. Different functional local links may be mapped onto the same physical local bus if their source and destination PE instances are identical. Variable $r_{j1,k1,j2,k2}$ is calculated as

$$r_{j1,k1,j2,k2} = \bigvee_{C_n \in E} \left(x_{i1,j1,k1} \wedge x_{i2,j2,k2} \wedge l_n \right), \ \forall\ M_{j1} \text{ and } M_{j2} \in MLib,\ 1 \le k1 \le MN_{j1} \text{ and } 1 \le k2 \le MN_{j2} \quad (4)$$

where $\vee$ denotes the OR operation. Then, the number of used local bus, denoted as BD, can be evaluated as follows.

$$BD = \sum_{M_{j1} \in MLib} \sum_{1 \le k1 \le MN_{j1}} \sum_{M_{j2} \in MLib} \sum_{1 \le k2 \le MN_{j2}} r_{j1,k1,j2,k2} \quad (5)$$

Note that BD must be smaller than or equal to BN.

3.3: SCHEDULE TIME VARIABLES

Assume that $FT_{i,j,h}$ denotes the computation time for functional node F_i performed by PE of type M_j run at voltage $V_{j,h}$ and CT_n denotes the communication time for transferring the data of $C_n(F_{i1}, F_{i2})$ through system bus. Both of them are provided from implementation information. Then, the end time F_i^{ET} of $F_i \in V$ and the end time C_n^{ET} of communication edge $C_n(F_{i1}, F_{i2}) \in E$ can be calculated as

$$F_i^{ET} = F_i^{ST} + \sum_{M_j \in ML_i} \sum_{1 \le h \le VN_j} \left(FT_{i,j,h} \cdot \sum_{1 \le k \le MN_j} x_{i,j,k} \right) \quad (6)$$

$$C_n^{ET} \;=\; C_n^{ST} \;+\; CT_n \cdot s_n \quad (7)$$

3.4: PIPELINE VARIABLES

Each functional node $F_i \in V$ must be scheduled into a pipeline stage F_i^{Stage} which can be determined in (8). The pipeline start time F_i^{PST} and pipeline end time F_i^{PST} of F_i in a steady state of the pipeline are calculated in (9) and (10). respectively. The pipeline stage C_n^{Stage}, pipeline start time C_n^{PST}, and pipeline end time C_n^{PET} of communication edge C_n can be determined and calculated by the similar method.

$$\mathfrak{I} \cdot (F_i^{Stage} - 1) \le F_i^{ST} < \mathfrak{I} \cdot F_i^{Stage} \quad (8)$$

$$F_i^{PST} = F_i^{ST} - \mathfrak{I} \cdot (F_i^{Stage} - 1) \quad (9)$$

$$F_i^{PET} = F_i^{ET} - \mathfrak{I} \cdot (F_i^{Stage} - 1) \quad (10)$$

3.5: OBJECTIVE FUNCTION

The objective of our ILP model is to minimize the total area and/or total power consumption of the system while satisfying the throughput constraints. In addition, the stage number of the system (denote by O_S) is also minimized simultaneously. Let MA_j, LBA, and SBA denote the area of M_j, local bus, and system bus, respectively. $FP_{i,j,h}$, CSP_n, and CLP_n are power consumption of F_i performed by PE of type M_j run at voltage $V_{j,h}$, C_n transferred on the system bus , and C_n transferred on a local bus, respectively. The total area and total power consumption of the system, denoted as T_Area and T_Power, can be expressed as (11) and (12), respectively. The objective function in (13) uses α and β to scale the weight of area and power. Note that α and β have to be large enough to ensure that O_S is the second object after minimizing power and area.

$$T\_Area = \sum_{M_j \in MLib} MA_j \cdot MD_j + LBA \cdot BD + SBA \quad (11)$$

$$T\_Power = \sum_{F_i \in V} \sum_{M_j \in ML_i} \sum_{1 \le h \le VN_j} \left(FP_{i,j,h} \cdot v_{i,j,k,h} \right) + \sum_{C_n \in E} \left(CSP_n \cdot s_n + CLP_n \cdot l_n \right) \quad (12)$$

Minimize $\alpha \cdot T\_Area + (\beta - \alpha) \cdot T\_Power + O\_S$ (13)

3.6: PE SELECTION CONSTRAINT

(14) and (15) show that each functional node $F_i \in V$ is executed exactly on one PE instance run at a specific voltage. Therefore,

$$\sum_{M_j \in ML_i} \sum_{1 \le k \le MN_j} x_{i,j,k} = 1, \qquad \forall\, F_i \in V \quad (14)$$

$$x_{i,j,k} = \sum_{1 \le h \le VN_j} v_{i,j,k,h} \quad (15)$$

3.7: DEPENDENCY CONSTRAINTS

The communication edge $C_n(F_{i1}, F_{i2}) \in E$ must be scheduled between F_{i1} and F_{i2}.The edge with $D_n > 0$ must delay the communication for D_n stage time. The dependency constraint is shown in (16). (17) restricts that the last node must be scheduled at or before $\mathfrak{I} \cdot S_{MAX}$, where CO is the set of incoming edges of the output node O.

$$C_n^{ET} - \mathfrak{I} \cdot D_n \le F_{i1}^{ET} \quad \text{and} \quad F_{i2}^{ST} \le C_n^{ST} \quad (16)$$

$$C_n^{ET} \le \mathfrak{I} \cdot S_{MAX}, \quad \forall\, C_n \in CO \quad (17)$$

3.8: PE SHARING CONSTRAINT

If multiple functional nodes, e.g. F_{i1} and F_{i2}, are mapped onto the same PE instance, their computation time cannot be overlapped in the steady state. Therefore, the execution interval of F_{i1} (i.e., $[F_{i1}^{PST}, F_{i1}^{PET}]$) in the steady state must not be overlapped with that of F_{i2} (i.e., $[F_{i1}^{PST}, F_{i2}^{PET}]$), where F_i^{PST} and F_i^{PET} denote the start time and end time of F_i in a steady state of the pipeline. (18) and (19) represent the constraint, where $w_{i1,i2}$ is a binary variable that is 1 if $F_{i1}^{PST} < F_{i2}^{PST}$, else 0. When $x_{i1,j,k}$ or $x_{i2,j,k}$ is 0, F_{i1} and F_{i2} are not mapped onto the same PE instance of type M_j and both (18) and (19) are always true. When both $x_{i1,j,k}$ and $x_{i2,j,k}$ are 1, either $F_{i2}^{PST} \ge F_{i1}^{PST}$ and $\mathfrak{I} + F_{i1}^{PST} \ge F_{i2}^{PST}$ ($w_{i1,i2} = 1$) or $\mathfrak{I} + F_{i2}^{PST} \ge F_{i1}^{PET}$ and $F_{i1}^{PST} \ge F_{i2}^{PET}$ ($w_{i1,i2} = 0$) must be satisfied such that the intervals cannot overlap each other.

$$2\mathfrak{I} \cdot (1 - x_{i1,j,k}) + 2\mathfrak{I} \cdot (1 - x_{i2,j,k}) + \mathfrak{I} \cdot (1 - w_{i1,i2}) + F_{i2}^{PST} \ge F_{i1}^{PET} \quad (18)$$

$$2\mathfrak{I} \cdot (1 - x_{i1,j,k}) + 2\mathfrak{I} \cdot (1 - x_{i2,j,k}) + \mathfrak{I} \cdot w_{i1,i2} + F_{i1}^{PST} \ge F_{i2}^{PET} \quad (19)$$

3.9: BUS USAGE CONSTRAINT

For each C_n, the data should be transferred through the system bus, through a local bus, or not through bus. Only one type is true, therefore

$$y_n + s_n + l_n = 1\,, \quad \forall\, C_n \in E \quad (20)$$

In addition, if multiple communication edges, e.g., C_{n1} and C_{n2}, are mapped onto the system bus, their data cannot be transmitted concurrently. This constraint can be defined to be analogous to (18) and (19) as follows.

$$2\mathfrak{I} \cdot (1 - s_{n1}) + 2\mathfrak{I} \cdot (1 - s_{n2}) + \mathfrak{I} \cdot (1 - u_{n1,n2}) + C_{n2}^{PST} \ge C_{n1}^{PET} \quad (21)$$

$$2\mathfrak{I} \cdot (1 - s_{n1}) + 2\mathfrak{I} \cdot (1 - s_{n2}) + \mathfrak{I} \cdot u_{n1,n2} + C_{n1}^{PST} \ge C_{n2}^{PET} \quad (22)$$

where $u_{n1,n2}$ is an auxiliary binary variable, C_n^{PST} and C_n^{PET} denote the start time and end time of C_n in a steady state of the pipeline.

4: EXPERIMENTAL RESULTS

In this section, two experiments are made to demonstrate the advantages of proposed ILP approach. The first experiment explains how pipelined scheduling, communication synthesis and DVS impact the system's area and power, and shows the benefit of solving them simultaneously. The second experiment tests the ability of proposed approach to trade system's throughput with area and power consumption. In these experiments, all the input data including task graph, PE library and implementation information are described in a text file. We wrote some PERL scripts to generate the corresponding ILP formulation from the file and the ILP solver LINGO was used to solve the ILP formulation with complex constraints on a PC with the 2.0 GHz processor and 2G main memory. The CPU run time for solving the ILP formulation in our experimental suite varied between 759 seconds and 2714.75 minutes.

In the first experiment, a task graph with eight functional nodes and twelve communication edges is adopted. The available number of pipeline stage S_{MAX}, available number of local bus *BN*, available number of each type of processors, and available number of each type of ASIC are 3, 3, 1, and 1, respectively. In addition, there are three discrete voltage levels for each type of processor, but only one voltage level for each ASIC. To compare with other optimal approaches, we simplify our ILP formulation to approximate the conventional optimal approaches and compare their design quality in terms of total area and total power consumption. Three approximate approaches ('-dvs', '-pipe', and '-local') are adopted in this experiment. The minus symbol means that the corresponding design step is removed from our ILP model. That is, '-dvs' represents that DVS doesn't be performed by removing the two lower voltage levels for all processors. The '-pipe' is the approach without pipelined scheduling by setting the available pipeline stage number S_{MAX} to 1. The '-local' approach gives the limitation that all data are only transferred through the system bus and the number of available local bus *BN* is set to 0.

Table 1 shows the results of experiment 1 under a fixed throughput constraint $\mathfrak{T}$ = 500. Different combinations of α and β are used to scale the importance of area and power as described in (13). We assume that $0 \le \alpha \le 100$ and $\beta = 100$ in the experiment. Choice of $0 < \alpha < \beta$ implies that both area and power are considered simultaneously. The importance of both factors depends on the value of α and the resultant total area and power values are usually between the corresponding values obtained by the $\alpha = 0$ and $\alpha = 100$ extremes. The larger α means that area is more important in the design. In addition, *OBJ* used to evaluate the synthesis results of these approaches is obtained by followed formulation:

$$OBJ = [\alpha \cdot T\_Area + (\beta - \alpha) \cdot T\_Power] / 100 \quad (23)$$

T_A and *T_P* in Table 1 denote the *T_Area* and *T_Power* of synthesized system architecture. The results show that our proposed approach can actually explore larger design space to obtain better solution than other approaches. The proposed approach is especially useful when reducing power consumption is the most important design goal (i.e., α is very small). By the design results with the combination of $\alpha = 5$ and $\beta = 100$, DVS can achieve more significant power saving than pipelined scheduling and local bus communication. However, DVS seems unable to work very well when reducing system's area is the main design goal. On the other hand, pipelined scheduling plays an important role in reducing both of area and power consumption. For example, the '-pipe' approach in the combination of α = 90 and β = 100 gives a non-pipelined system architecture which consists of PEs with higher voltage level and additional local buses to satisfy throughput constraint. Therefore, larger area and more power consumption are required. Finally, transferring data through a local bus not only saves power than the system bus, but also increase the slack time (transferring data through local bus spends zero communication time). Therefore, local bus communication can further enhance the efficiency of DVS to reduce the power consumption. Similar to DVS, however, local bus communication seems to be unhelpful to reduce the system's area.

Table 1: Design results of experiment 1

option	$\alpha = 5, \beta = 100$		$\alpha = 20, \beta = 100$		$\alpha = 90, \beta = 100$	
	(*T_A*, *T_P*)	*OBJ*	(*T_A*, *T_P*)	*OBJ*	(*T_A*, *T_P*)	*OBJ*
-dvs	(1370, 377)	426.7	(1020, 410)	532.0	(600, 583)	598.3
-pipe	(1220, 232)	281.4	(1070, 250)	414.0	(650, 575)	642.5
-local	(1200, 204)	253.8	(1200, 204)	403.2	(600, 565)	596.5
our	(1350, 188)	246.1	(1200, 204)	403.2	(600, 565)	596.5

The second experiment adopts another task graph with six nodes and seven edges. The available numbers of pipeline stage, local bus, microprocessor, DSP processor, and each type of ASIC are set to 3, 2, 1, 1, and 1, respectively. We varied the value of throughput constraint $\mathfrak{T}$ to generate the pipelined architectures with different system costs (area and power consumption). Moreover, various choices of α have been considered under the same throughput constraint $\mathfrak{T}$ to trade system area with system power consumption.

Table 2 lists the design results including the total area (*T_A*), the total power consumption (*T_P*), the number of microprocessor used (*#uP*), the number of DSP processor used (*#DSP*), the number of ASIC used (*#ASIC*), the number of local bus used (*BD*), and the pipeline stage used (*#S*) for different throughput constraints. When throughput constraint $\mathfrak{T}$ is loose (1200), the resulted architecture may be non-pipelined (i.e. *#S* = 1) and has the least total area and total power consumption. When throughput constraint is tight, the resulted architecture is pipelined to satisfy the constraint and has lager total area and total power consumption. Moreover, various α values under the same throughput constraint $\mathfrak{T}$ lead to different system architectures. The results in Table 2 indicate that the proposed ILP formulation is able to make the tradeoffs between system throughput and system cost as well as the total area and total power consumption.

Table 2: Design results of experiment 2

$\Im$	α	*T_A*	*T_P*	*#uP*	*#DSP*	*#ASIC*	*BD*	*#S*
200	90	1570	531	1	1	1	1	3
	50	1570	531	1	1	1	1	3
	20	1670	463	1	1	1	3	3
	10	1970	426	1	1	2	3	3
	5	1970	426	1	1	2	3	3
300	90	1200	592	1	1	0	0	3
	50	1220	485	1	0	2	0	3
	20	1570	330	1	1	1	1	3
	10	1670	316	1	1	1	3	3
	5	1970	285	1	1	2	3	3
500	90	920	430	1	0	1	0	3
	50	920	430	1	0	1	0	3
	20	1200	322	1	1	0	0	3
	10	1570	238	1	1	1	1	2
	5	1670	230	1	1	1	3	2
800	90	600	478	1	0	0	0	3
	50	600	478	1	0	0	0	2
	20	920	251	1	0	1	0	2
	10	1200	215	1	1	0	0	2
	5	1350	205	1	1	0	3	2
1200	90	600	258	1	0	0	0	2
	50	600	258	1	0	0	0	2
	20	750	218	0	1	0	0	2
	10	750	218	0	1	0	0	2
	5	850	211	0	1	0	2	1

5: CONCCLUSION

Mapping, pipelined scheduling, communication synthesis, and DVS are important and strong inter-dependent design steps in embedded system design. This paper has proposed an ILP based approach to solve them simultaneously. The proposed approach took voltage selection into account while performing mapping, pipelined scheduling and communication synthesis to ensure that the design results can conform to the real situation. Experimental results have shown that the proposed ILP approach can generate the optimal pipelined architecture with the least total area and/or total power consumption for multimedia applications.

6: ACKNOWLEDGMENT

This work was supported in part by the National Science Council, R.O.C., under Grant NSC 94-2213-E-110-050.

REFERENCES

[1] T. Yen and W. Wolf, *Hardware-Software co-synthesis of distributed embedded systems.* Kluwer academic publishers, 1996.

[2] O. Ogawa, S. Bayon de Noyer, P. Chauvet, K. Shinohara, Y. Watanabe, H. Niizuma, T. Sasaki, and Y. Takai, "A practical approach for bus architecture optimization at transaction level," in *Proc. Design, Automation and Test in Europe Conference and Exhibition*, pp. 176–181, 2003.

[3] H. Liu and D. F. Wong, "Integrated partitioning and scheduling for hardware/software co-design," in *Proc. International Conference on Computer Design: VLSI in Computers and Processors*, pp. 609–614, Oct. 1998.

[4] R. Niemann and P. Marwedel, "An algorithm for hardware/software partitioning using mixed integer linear programming," in *Proc. Design Automation for Embedded Systems*, vol. 2, pp. 165–193, March 1997.

[5] S. A. Khayam, S. A. Khan, and S. Sadiq, "A generic integer programming approach to hardware/software codesign," in *Proc. IEEE Int. Multi Topic Conf.*, pp. 6–9, Dec. 2001.

[6] R. P. Dick and N. K. Jha, "MOGAC: a multiobjective genetic algorithm for hardware-software cosynthesis of distributed embedded systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, No. 10, pp. 920–935, 1998.

[7] L. Pontani and D. Dupont, "Scheduling and assignment for real-time embedded systems with resource contention," in *Proc. Euromicro Symp. on Digital System Design*, pp. 55–61, Sept. 2003.

[8] S. Banerjee and N. Dutt, "Efficient search space exploration for HW-SW partitioning," in *Proc. Int. Conf. on Hardware/Software Codesign and System Synthesis*, pp. 122–127, 2004.

[9] S. Bakshi and D. D. Gajski, "Partitioning and pipelining for performance-constrained hardware/software systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, Vol. 7, No. 4, pp. 419–432, 1999.

[10] K. S. Chatha and R. Vemuri, "Hardware-software partitioning and pipelined scheduling of transformative applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, Vol. 10, No. 3, pp. 193–208, 2002.

[11] B. P. Dave and N. K. Jha, "COHRA: hardware-software cosynthesis of hierarchical heterogeneous distributed embedded systems," *IEEE Trans. Compu.-Aided Des. Integr. Circuits Syst.*, Vol. 17, No. 10, pp. 900–919, 1998.

[12] B. P. Dave, G. Lakshminarayana, and N. K. Jha, "COSYN: Hardware-software co-synthesis of heterogeneous distributed embedded systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, Vol. 7, No. 1, pp. 92–104, 1999.

[13] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," in *Proc. Int. Symp. on Low Power Electronics and Design*, pp. 197–202, 1998.

[14] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, "Energy-efficient mapping and scheduling for DVS enabled distributed embedded systems," in *Proc. Design, automation and test in Europe Conf.*, pp. 514–521, March 2002.