Robust On-Chip Bus Architecture Synthesis for MPSoCs Under Random Tasks Arrival

Sujan Pandey[†]
NXP Semiconductors Research
Eindhoven, The Netherlands
pandey@informatik.uni-bremen.de

Rolf Drechsler
Dept. of Computer Science
Univ. of Bremen, Germany
drechsle@informatik.uni-bremen.de

***Abstract*— A major trend in a modern system-on-chip design is a growing system complexity, which results in a sharp increase of communication traffic on the on-chip communication bus architectures. In a real-time embedded system, task arrival rate, inter-task arrival time, and data size to be transferred are not uniform over time. This is due to the partial re-configuration of an embedded system to cope with dynamic workload. In this context, the traditional application specific bus architectures may fail to meet the real-time constraints. Thus, to incorporate the random behavior of on-chip communication, this work proposes an approach to synthesize an on-chip bus architecture, which is robust for a given distributions of random tasks. The randomness of communication tasks is characterized by three main parameters which are the average task arrival rate, the average inter-task arrival time, and the data size. For synthesis, an on-chip bus requirement is guided by the worst-case performance need, while the dynamic voltage scaling technique is used to save energy when the workload is low or timing slack is high. This, in turn, results in an effective utilization of communication resources under variable workload.**

I. Introduction

After partitioning of a complex system into hardware and software, and mapping it onto the appropriate modules of an SoC, parts of a system's functions are implemented in software that runs on a standard processor while the rest of system functions are implemented in (synthesized) hardware. These hardware and software components communicate with each other by exchanging data through communication resources to accomplish a certain task. At this point of the design flow, a big challenge left to the designer is the synthesis of an efficient on-chip communication architecture.

The early works on communication synthesis mainly focused on minimizing the bus width for a single global bus [8]. In [18] an automatic bus generation for an MPSoC was proposed. The approach considers three different types of buses, which can be generalized to a shared bus, point-to-point, and FIFO based architecture. The bus architecture is generated for a given bus width considering real-time constraints. In [17] a method of communication synthesis based on the library elements and constraints graph was presented, where the library elements are a collection of communication links and communication nodes. The approach mainly focuses on synthesizing a communication bus topology for a point-to-point communication architecture. In [21] a bus model for communication in embedded systems with arbitrary topologies was proposed, where a point-to-point communication is a special case for the real-time application. The algorithm selects the number of buses, the type of each bus, the message transferred on each bus, and schedules the communication bus. In [9] a template based communication synthesis technique was presented that supports shared buses and point-to-point connection. In [16] a floorplan aware automated bus based communication synthesis algorithm was proposed, which is, however mainly based on the bus templates of AMBA [1] with standard bus widths.

All the above techniques synthesize an application specific bus architecture, which may fail to meet the real-time constraints, if the communication behavior of tasks is random. Recently, a technique for dynamic re-configuration of a synthesized bus topology was studied in [19] to cope with variable workload. The approach optimizes a post synthesis bus architecture for the re-configuration of bus protocols. A bus scheduling approach for aperiodic tasks was proposed in [6], which models random tasks and schedules them for a synthesized bus architecture. This approach also deals with the bus optimization technique rather than synthesizing a bus architecture. However, the random tasks modeling technique proposed in our work is similar to their approach.

The idea of scaling voltage of a task by exploiting its timing slack for energy reduction is not new. The technique presented in [10] proposes dynamic voltage scaling of a microprocessor under variable workloads, while the work of [7] used a voltage scaling technique for both processor and communication bus for energy reduction of IPs and bus architecture. However, the approach is only used for the power optimization of a post synthesis bus architecture. Recently, a simultaneous bus synthesis and voltage scaling technique was presented in [14], [15], which finds the optimal bus width and the number of buses. Furthermore, it explores a trade-off between communication resources and power consumption during bus synthesis. The bus synthesis technique in our work is similar to [14], [15], however, the proposed approach in [14], [15] was limited to a task with a deterministic arrival time. Thus, the synthesized bus architecture may fail to meet the real-time constraint if the arrival time and rate of tasks is random due to the partial re-configuration of a system.

The main contribution of this work is to synthesize a bus architecture in the presence of random tasks arrival. As a result of this, the synthesized bus architecture is robust for a given probability distribution of random tasks. The randomness of tasks is modeled with three parameters, i.e., task arrival rate, inter-task arrival time, and data size with their probability distribution functions. The bus synthesis is guided by the worst-case performance need, while the dynamic voltage scaling technique is used to reduce the energy when the timing slack is high or the workload is low. The dynamic voltage scaling technique presented in this work is similar to [7], [14], [15]. In this paper, the bus synthesis problem is formulated as an optimization problem and solved using a convex optimization tool. The experiments carried out on automatically generated tasks and real-life multimedia applications validate the proposed bus synthesis technique under random task arrival and show that the synthesized bus architecture is robust for a given distribution of random tasks.

The reminder of this paper is organized as follows. In Section II, we give preliminaries on the target architecture model and com-

[†] This work was done, while the first author was with the Computer Architecture Group at University of Bremen, Germany.

munication tasks. Section III introduces a motivational example for bus synthesis under random task arrival rate, inter-task arrival time, and random data size. Section IV derives a model for random task arrival rate, inter-task arrival time, and data size to synthesize a bus architecture. Section V gives a mathematical formulation and optimization techniques for bus synthesis and voltage scaling problems. The continuously scaled voltages of each task are transformed into discrete voltages in Section VI using an voltage selection algorithm. In Section VII, we present case studies and results to validate our bus architecture synthesis method under random task arrival and finally, in Section VIII, we give the conclusion of this work.

II. Preliminaries

We consider an embedded system which is realized as a multiprocessor system-on-a-chip (MPSoC). Such a system consists of several on-chip processing modules like general-purpose processors, application specific integrated processors (ASIPs), application specific integrated circuits (ASICs) or field-programmable gate arrays (FPGAs). These on-chip modules communicate with each other by transferring data through a shared bus. We assume that a system has been partitioned into HW/SW and mapped efficiently onto the appropriate modules of an SoC as shown in Fig. 1(a). In the figure, a set of tasks $\tau \in T$, which is mapped onto a module, is called *data processing tasks*. These tasks are for processing data such as fast Fourier transformation (FFT) or discrete cosine transformation (DCT) or any computation within a module. After processing data, a driver of a module, establishes communication between modules and transfers data for further processing. All communications $c \in C$ that take place among the on-chip modules using on-chip buses are captured by communication tasks c_i as indicated by black boxes as shown in Fig. 1(b). Since a complex system runs a diversity of applications on a single SoC, the workload offered to an embedded system is not uniform over time. This introduces randomness on size of data to be transferred, communication task arrival rate, and inter-task arrival time. These parameters are extracted by profiling a HW/SW system for different scenarios at system level using the following formulae

$$\sum c_i \quad : t_i \leq t \leq t_j, \tag{1}$$

$$\sum_{(i=1,j=i+1)}^{|C|} ASAP(c_j) - ASAP(c_i). \tag{2}$$

Eq. (1) and (2) give the task arrival rate and the inter-task arrival time for a given window t_i and t_j, respectively. Based on the profiling of a system, communication tasks and their dependencies are modeled as shown in Fig. 1(a), where communication tasks c_1, c_2, and c_3 with solid lines are the tasks for one scenario. While additional tasks c_4 and c_5 with dotted lines are for another scenario. Each communication task in the figure takes certain time duration to transfer data. This duration is called a *communication lifetime interval* (CLTI), which is a function of data size, bus width, and voltage. From the extended task graph $G_E(T, E)$ a directed acyclic communication task graph $G_C(C, \Pi)$ is obtained to schedule communication tasks for different bus widths and voltages. In Fig. 1(c), a node $c \in C$ is a communication task, while an edge $\pi \in \Pi$ gives the dependency between the communication tasks. Further, an edge between two nodes c_i and c_j weighted with w is the data processing time of a task τ_i, which gives an early start time constraint for a successor c_j to transfer data using a bus. The data processing time of each task τ can be evaluated as

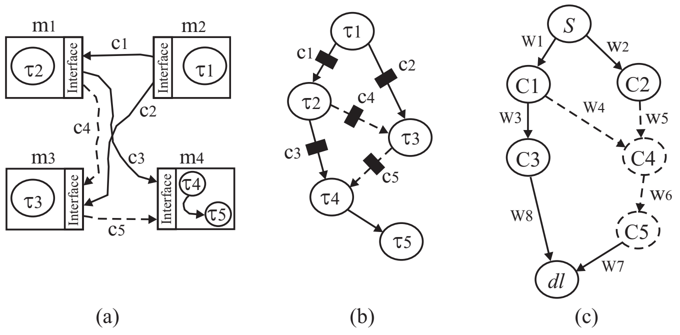

Fig. 1. Architecture model with random tasks. (a) Target architecture with mapped tasks. (b) Extended task graph $G_E(T, E)$. (c) Communication task graph $G_C(C, \Pi)$.

$$w = \sum_{\tau \in T} NC_\tau \cdot T_d, \tag{3}$$

where NC_τ is the number of cycles to execute a task τ and T_d is a gate delay, which is a function of voltage and technology dependent parameters [20] as shown in Eq. (4). Similarly, the CLTI of each communication task c is modeled as

$$T_d = \frac{\kappa_3 \cdot V_{dd}}{[\kappa_1 \cdot V_{dd} + \kappa_2 \cdot V_{bs} - V_{th}]^\alpha}, \tag{4}$$

$$CLTI_{c,r,V_{dd},V_{bs}} = \left\lceil \frac{NB_c(\zeta)}{b_r} \right\rceil \cdot T_d, \tag{5}$$

where $NB_c(\zeta)$ (number of bits) is a random size of data to be transferred by a task c with bus width b_r, and supply voltage V_{dd}. In Eq. (4) notations κ_1, κ_2, κ_3, and α are the technology dependent parameters. The dynamic energy consumption of communication task c is modeled as

$$E_c = C_{eff} \cdot V_i^2 \cdot T_d(\zeta), \tag{6}$$

where C_{eff} is the effective switched capacitance of the communication bus. The energy overhead for switching from voltage V_i to V_j is

$$\varepsilon_{i,j}^{\Delta V} = C_r(V_i - V_j)^2 \cdot \delta_{i,j}^{\Delta V}, \tag{7}$$

where C_r is the capacitance of the power rail. The time overhead for switching from V_i to V_j is given by

$$\delta_{i,j}^{\Delta V} = \rho|V_i - V_j|, \tag{8}$$

where ρ is a constant.

III. Motivational example

In this section we present a motivation for bus synthesis under a random task arrival rate with a random inter-task arrival time and illustrate that the synthesized bus is robust for a given probability distribution of the task arrival rate, the inter-task arrival time, and the data size. The voltage scaling technique is used to reduce the bus energy consumption when the data traffic is low or the timing slack of the communication tasks is high. This, in turn, results in an optimum utilization of the buses under random data traffic. We consider the partitioned and mapped system as shown in Fig. 1(a) and schedule the tasks to synthesize a bus architecture. Fig. 2 depicts task scheduling and bus synthesis for three different scenarios, where each scenario is characterized by average task arrival rate λ_n, average inter-task arrival time λ_τ, and random data size $NB(\zeta)$. In the figure a black rectangle denotes the data transfer delay at the nominal voltage and a white rectangle denotes a slack of a communication task. Fig. 2(a) depicts three communication tasks with inter-task arrival time 1/3·[{ASAP(c_2)-ASAP(c_1)}+{ASAP(c_3)-ASAP(c_2)}+{ASAP(c_3)-ASAP(c_1)}] = 2.66ms. After scheduling the tasks the synthesized bus architecture with interconnection of modules is shown in Fig. 2(d). As the tasks in Fig. 2(a) do not overlap with each other, a single shared bus meets the real-time constraints. Fig. 2(b) shows the schedule of five communication tasks with average inter-task arrival time 2.9ms. In this scenario the synthesized single shared bus shown in Fig. 2(d) does not meet the real-time constraints, as the tasks overlap with each other. Thus, two shared buses are needed to meet the time constraints as shown in Fig. 2(e). Similarly, in Fig. 2(c) five communication tasks with average inter-task arrival time 2.8ms are depicted. After scheduling of the tasks, two shared buses with interconnection of modules are shown in Fig. 2(e).

Among the three different scenarios shown in Fig. 2 the worst-case performance needed is λ_n = 5 and λ_τ = 2.8ms. Thus, the bus is synthesized considering the worst-case scenario and the voltage of communication tasks is scaled to reduce the energy consumption when the average number of task arrivals λ_n is low and the average inter-task arrival time λ_τ is high or the timing slack is high. This effectively utilizes the bus resources even when there is a variation in data traffic. In Fig. 2(c), the slack of the communication tasks c_1, c_2, and c_5 can be used to scale the voltage, while the voltage of other tasks should be kept to the nominal voltage. Similarly, for the scenario with λ_n = 3 and λ_τ = 2.66ms, the slack of communication tasks c_1, c_2, and c_3 can be exploited for energy reduction. Intuitively, the more the slack of communication task is, the less is the energy consumption due to voltage scaling.

IV. Modeling of random tasks

We assume that the communication tasks $c \in C$ and their arrival rate and inter-task arrival time have a Poisson distribution. Eqs. (9) and (10) give probability density functions of the task arrival rate and inter-task arrival time, respectively.

$$f(c, \lambda_n) = \frac{e^{-\lambda_n} \cdot \lambda_n^c}{c!}, \tag{9}$$

where
c = number of communication tasks
λ_n = average arrival rate of communication tasks in a given time interval $[t_i,\, t_j]$

$$f(t, \lambda_\tau) = \frac{e^{-\lambda_\tau} \cdot \lambda_\tau^t}{t!}, \tag{10}$$

where
t = arrival time of a communication task
λ_τ = average inter-task arrival time of communication tasks

The probabilistic constraint of random task arrival rate can be expressed as

$$P(C_n \leq c_l) \geq \beta_c, \tag{11}$$

where β_c is a confidence level such that in Eq. (11) the number of task arrivals in a time interval $[t_i,\, t_j]$ is less than or equal to c_l with a probability β_c. After an algebraic manipulation of Eqs. (11) and (9), Eq. (11) can be expressed as

$$\lambda_n \leq -ln(1 - K_c \cdot \beta_c). \tag{12}$$

Similarly, the probabilistic constraint of random inter-task arrival time can also be modeled as Eqs. (11) and (12) with confidence level β_τ.

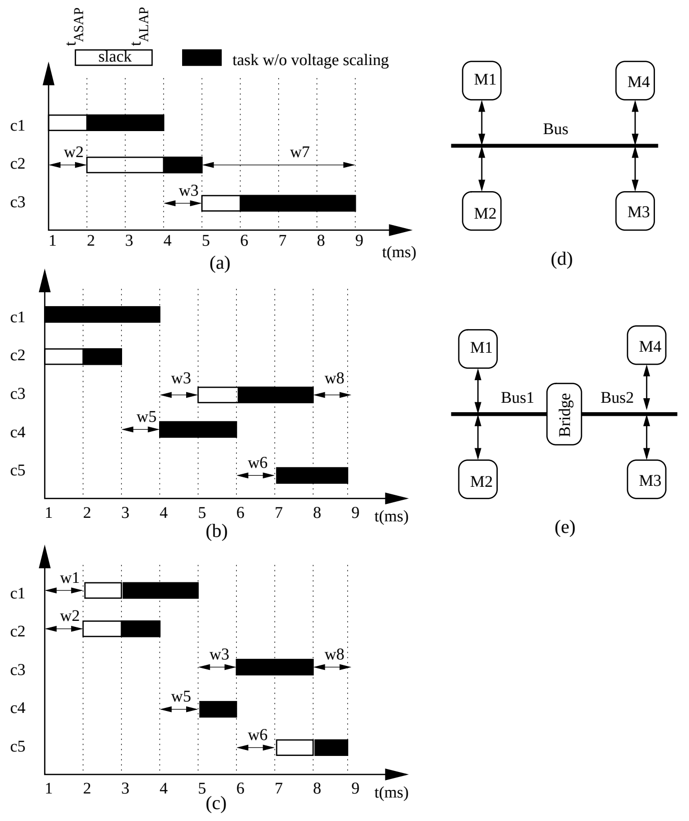

Fig. 2. Random number of tasks with their random arrival time. (a) Task scheduling with average tasks arrival rate (λ_n) = 3 and average inter-task arrival time (λ_τ) = 2.66ms. (b) Task scheduling with average task arrival rate (λ_n) = 5 and average inter-task arrival time (λ_τ) = 2.9ms. (c) Task scheduling with average task arrival rate (λ_n) = 5 and average inter-task arrival time (λ_τ) = 2.8ms. (d) and (e) Synthesized bus for different scenarios (a), (b), and (c) respectively.

In Eq. (12) the notation K_c is a constant term. Further, data size to be transferred by each communication task $c \in C$ is modeled as a random variable with a known probability distribution function. Let dl be the deadline of task $c \in C$ then the relation between the CLTI and deadline dl can be written as [14], [15]

$$\forall c \in C,\; P(dl_c - CLTI_{c,r,V_{dd},V_{bs}} - \delta_{i,j}^{\Delta V} \geq 0) \geq \eta. \tag{13}$$

Eq. (13) gives a probabilistic delay constraint for each task $c \in C$ such that the data transfer delay of each task should be less than or equal to the deadline. The notation η can be considered to be a confidence level. This constraint can be transferred into the deterministic constraint as follow [14], [15]

$$\begin{aligned} &dl_c - \mu_{CLTI}(NB_c, T_d) - \delta_{i,j}^{\Delta V} \\ &- \phi^{-1}(1-\eta) \cdot \sigma_{CLTI}(NB_c, T_d) \geq 0, \end{aligned} \tag{14}$$

where $\mu_{CLTI}(NB_c, T_d)$ and $\sigma_{CLTI}(NB_c, T_d)$ are mean and standard deviation of the CLTI, respectively. The term $\phi^{-1}(\cdot)$ is an inverse of the error function. Intuitively in Eq. (14) the notation η controls the scaling of voltage during bus synthesis. For different arrival rate, arrival time, and data size of tasks, the confidence level η sets the voltage to a certain level[1] so that the standard deviation of delay $\sigma_{CLTI}(NB_c, T_d)$ is changed in order to meet the real-time constraint and to utilize the bus resources effectively.

[1]The voltage should be in between V_{max} and V_{min}.

V. Simultaneous bus synthesis and voltage scaling

For the simultaneous bus synthesis and voltage scaling problem, the data processing tasks τ and communication tasks c are scheduled together, where voltage is scaled to reduce the energy consumption when the workload of a system is low or the timing slack is high. The formulation of an optimization problem is given as follows:
Minimize:

$$\sum_{r \in R} C_r \cdot r_i \tag{15}$$

subject to,

$$s_\tau + w_{\tau,V_{dd},V_{bs}} + \delta_{i,j}^{\Delta V} \leq dl_\tau \quad : \forall \tau \in T \tag{16}$$

$$s_\tau \geq s_{c'} + CLTI_{c',r,V_{dd},V_{bs}} \cdot X_{c,t,r} + \delta_{i,j}^{\Delta V} \quad : \forall (c, c') \in \Pi \tag{17}$$

$$\sum_{r \in R} \sum_{t=ASAP}^{ALAP} X_{c,t,r} = 1 \quad : \forall c \in C \tag{18}$$

$$t \cdot X_{c,t,r} \geq (s_{\tau'} + w_{\tau',V_{dd},V_{bs}} + \delta_{i,j}^{\Delta V}) \cdot X_{c,t,r} \quad : \forall\ (c', c) \in \Pi \tag{19}$$

$$\sum_{c \in C} \sum_{\substack{(t' \in \{t,\cdots,t+d_r-1\} \\ \cap \{ASAP_c,\cdots,\psi\})}} X_{c,t',r} \leq b_r \quad : \forall t \in \Omega, r \in R \tag{20}$$

$$P((dl_c - s_c - CLTI_{c,r,V_{dd},V_{bs}} - \delta_{i,j}^{\Delta V}) \cdot X_{c,t,r} \geq 0) \geq \eta \tag{21}$$

$$P(C_n \leq c_l) \cdot X_{c,t,r} \geq \beta_c \ \forall c \in C \tag{22}$$

$$P(C_\tau \leq c_t) \cdot X_{c,t,r} \geq \beta_\tau \ \forall c \in C \tag{23}$$

$$V_{dd_{min}} \leq V_{dd} \leq V_{dd_{max}} \wedge V_{bs_{min}} \leq V_{bs} \leq V_{bs_{max}} \tag{24}$$

The objective is to minimize the communication bus cost (bus width and number of buses) as shown in Eq. (15), where $r_i \in R$ is a library of on-chip buses with different bus widths. The C_r of each bus r_i is expressed in terms of the bus width, e.g., the cost of a 32-bit wide bus is twice the cost of a 16-bit wide bus and is stored in a lookup table. In Eq. (16), summation of start time s_τ, execution time $w_{\tau,V_{dd},V_{bs}}$ and switching overhead $\delta_{i,j}^{\Delta V}$ of each task τ should be less than or equal to its deadline dl_τ. Further, a task τ can start its execution only after its predecessor (communication task c) completes transferring data as shown in Eq. (17). A binary decision variable $X_{c,t,r} \in \{0, 1\}$, indicates scheduling of a communication task c at time $t \in \{0, \cdots, \lambda\}$, with a bus width r as shown in Eq. (18). The term λ is the maximum possible time to schedule a task c. Eq. (19) gives a dependency between successor (communication task c) and predecessor (data processing task τ') such that a task c is scheduled at time t to maximize sharing of buses and to scale voltage for energy reduction. Let $\Omega = \cup_{c \in C}\{ASAP_c', \cdots, ALAP_c\}$ be a time window such that the tasks that are scheduled within this interval could overlap. If the timing of a task overlaps with another task then the task is assigned to a separate bus with index b and width r as shown in Eq. (20) [14], [15]. Since, the delay interval $CLTI_{c,r,V_{dd},V_{bs}}$ of a task c is a function of the two random variables data size NB_c and gate delay T_d (see Eq. (5)), Eq. (21) gives a probabilistic constraint such that the overall delay of each task c must be less than or equal to the deadline dl_c with a confidence level η. Its equivalent deterministic constraint is given in Eq. (14). Similarly, the probabilistic constraints for random task arrival rate and inter-task arrival time are given in Eqs. (22) and (23) with their confidence level β_c and β_τ respectively. The supply voltage V_{dd} and body bias voltage V_{bs} of both tasks τ and c are scaled continuously and their constraints are given in Eq. (24). In the above formulation, the objective function is linear to the optimization variable r_i and the probabilistic constraints (Eq. (21), (12), (22), and (23)) are non-linear to voltage and optimization variable r_i. Thus, the above described simultaneous bus synthesis and voltage scaling problem belongs to the convex quadratic optimization problem [15], which finds a global optimal solution in a polynomial time complexity [12].

```
HEURISTIC-DBS-DVS($b_r^{opt}, V_{dd}^{opt}, V_{bs}^{opt}$)
1  $V_{dd_z} \leftarrow$ GETDISCRETEVDD();
2  $V_{bs_z} \leftarrow$ GETDISCRETEVBS();
3  /*Linear relaxation method*/
4  $V_{dd}' \leftarrow \lceil V_{dd}^{opt} \rceil$ if $V_{dd}^{opt} \in [V_{dd_z}^{LB}, V_{dd_z}^{UB}]$;
5  $V_{bs}' \leftarrow \lceil V_{bs}^{opt} \rceil$ if $V_{bs}^{opt} \in [V_{bs_z}^{LB}, V_{bs_z}^{UB}]$;
6  /*Check condition*/
7  while $Eq.\ (16) \leq dl_\tau$ and $Eq.\ (21) \leq dl_c$
8  do
9      for $\tau \in T$ and $c \in C$
10     do
11             $V_{dd}' \leftarrow$ GETNEXTGREATERVDD() $\in V_{dd_z}$;
12         }
13     }
14     return $(V_{dd}', V_{bs}')$;
```

Algorithm 1: Discrete supply and body bias voltages selection.

VI. Discrete Voltage Selection

As the continuous voltage scaling technique gives an ideal energy reduction characteristics, it cannot be applied for digital design due to the limitations of a voltage regulator. Thus, a heuristic is proposed to transform continuously selected optimal supply voltage V_{dd}^{opt} and body bias voltage V_{bs}^{opt} from Section V into discrete voltages V_{dd}' and V_{bs}' as shown in Algorithm 1. At line 1-2, it reads a discrete set of supply V_{dd_z} and body bias V_{bs_z} voltages. At line 4-5, the algorithm quantises continuous voltages V_{dd}^{opt} and V_{bs}^{opt} to their upper bounds of V_{dd}' and V_{bs}', respectively. We could choose the lower bounds of supply and body bias voltages to get the minimum energy consumption, however, this may violate the given real-time constraints. At line 7 of the algorithm, delay constraints of each task τ and c are checked with their deadlines dl_τ and dl_c, respectively for near-optimal V_{dd}' and V_{bs}'. If the condition is met then the heuristic returns those near-optimal voltages at line 14. Otherwise, at each time next supply voltage, which is greater than the V_{dd}' is selected from the set of discrete supply voltages V_{dd_z} at line 11.

VII. Case Studies

We validate the effectiveness of the proposed technique using a generated benchmark as well as a real-life multimedia applications, i.e. an audio decoder [2] and a speech recognition system [3]. The automatically generated benchmark consists of 119 [15] communication tasks and the data size to be transferred by each task is a normally distributed random variable with mean data size (μ_{NB}) 16, 32, 64, 128, 256, and 512-bit and standard deviation $3\sigma_{NB}$ = 40% of μ_{NB}. Each data processing task τ and communication task c can scale their supply voltage from 1.4V to 0.8V and the body bias voltage from 0V to -0.8V. The on-chip communication buses are given as a library of buses with different bus widths, which range from 16 to 128-bit wide. For the experimental purpose, we consider a bus with

λ_τ	Average task arrival rate								Run time
	λ_n = 19				λ_n = 23				
	Synthesized Bus	Bus Cost	$\mu_{V_{dd}}$ (V)	$\mu_{V_{bs}}$ (V)	Synthesized Bus	Bus Cost	$\mu_{V_{dd}}$ (V)	$\mu_{V_{bs}}$ (V)	(sec)
0.3	1 $*$ 48-bit 1 $*$ 32-bit 2 $*$ 16-bit	7	1.10	-0.39	1 $*$ 64-bit 1 $*$ 32-bit 2 $*$ 16-bit	8	0.97	-0.41	$\sim$ 41
0.5	1 $*$ 48-bit 1 $*$ 32-bit 1 $*$ 16-bit	6	1.03	-0.38	1 $*$ 48-bit 1 $*$ 32-bit 1 $*$ 16-bit	6	1.09	-0.37	$\sim$ 39
0.7	1 $*$ 48-bit 2 $*$ 16-bit	5	1.05	-0.33	1 $*$ 48-bit 1 $*$ 32-bit	5	1.07	-0.39	$\sim$ 38
0.9	1 $*$ 48-bit 1 $*$ 16-bit	4	1.02	-0.32	2 $*$ 32-bit 1 $*$ 16-bit	5	1.05	-0.33	$\sim$ 35
1.0 [15]	1 $*$ 48-bit	3	0.99	-0.29	-	-	-	-	$\sim$ 32

TABLE I
THE SYNTHESIZED OPTIMAL BUS WIDTH AND NUMBER OF BUSES FOR DIFFERENT INTER-TASK ARRIVAL TIMES AND TASK ARRIVAL RATES

$4mm$ in length and its corresponding single line capacitance for 70nm technology is $609fF$ [10]. Other technology dependent parameters for 70nm node were adopted from [4]. The bus synthesis algorithm was implemented in C as a pre-processing model to interface with a convex solver MOSEK [5].

The first set of experiment was carried out on the automatically generated tasks with an aim to synthesize a robust bus architecture in the presence of random communication tasks with a random arrival time. The average number of task arrivals (λ_n) is 4 for a time interval of 6 sec. The confidence levels for the task arrival rate β_c and task arrival time β_τ are set to 99$\%$. While the confidence level η of tasks are set to 81$\%$. We performed simultaneous voltage scaling, scheduling, allocation, and binding of communication tasks using the proposed optimization technique presented in Section V. Table I presents the synthesized bus widths and the number of buses for different inter-task arrival times λ_τ and task arrival rates λ_n. The results are compared to the results of [15] with the deterministic task arrival (i.e., λ_τ = 1.0, λ_n = 19, 3σ_{NB} = 15$\%$, and η = 89$\%$ in the last row). The results show that the synthesized bus architecture considering a deterministic task arrival does not meet the real-time constraints for tasks with random task arrival. (Note, in [15] only the supply voltage of tasks is scaled, thus the mean supply voltage in the table for λ_τ = 1.0 and λ_n = 19 is high, it is because of V_{dd} and V_{bs} scaling.) In the column entitled λ_τ, the average inter-task arrival time of tasks is normalized to the maximum inter-task arrival time $\lambda_\tau(max)$. In the columns entitled *Synthesized Bus* (2 and 6), the synthesized bus widths and the number of buses are presented for different λ_n and λ_τ. The results show that the bus widths and number of buses increase with decreasing inter-task arrival time as shown in columns *Synthesized Bus* (2 and 6). Intuitively, the smaller the inter-task arrival time is, the larger is the number of overlaps among the tasks. This in turn results in an increment of communication bus cost. In the columns entitled *Bus Cost*, the cost was evaluated in terms of bus area so that the cost of 16, 32, 48, and 64-bit wide buses are 1, 2, 3, and 4, respectively. Similarly, column *Synthesized Bus* (6) shows the synthesized bus widths and the number of buses for average task arrival rate 23. In the columns entitled $\mu_{V_{dd}}$ and $\mu_{V_{bs}}$ (4, 5, 8, and 9) mean supply and body bias voltages were evaluated for different task arrival rates and inter-task arrival times.

The second experiment was conducted on real-life multimedia applications, which include an Ogg Vorbis decoder [2] and a speech recognition system [3]. The audio decoder includes four main decoding steps, which are inverse quantization, channel decoupling, reconstruct curve, and IMDCT. After manually partitioning and mapping of the decoder, the IMDCT was mapped to a single hardware and the rest of the functionality was mapped to a processor. Furthermore, raw audio data was mapped to a compact flash (CF) memory with an CF-interface. The extracted audio data was mapped to an audio buffer for streaming. Similarly, the second speech recognition system consists of three main components: front end, decoder, and linguist. The front end includes series of data processing tasks such as pre-emphasis, hamming window, FFT (fast Fourier transformation), mel frequency filter, IFFT, cepstral mean normalization, and feature extraction to generate the features from the speech. The speech system takes as input a large number of speech along with their transcriptions into phonemes to provide the speech models for the phonemes. The recognition is based on the HMM (hidden Markov model) to decode the speech. The American English lexicon consisting of 32 phonemes and a database of 17 different words has been used (spelling out the names of the months, numbers and digits) [11]. After partitioning of the speech system, the front end was mapped to a dedicated hardware including FFT and filters. The task training and recognition were mapped to a PowerPC processor. Based on the partitioned and mapped system communication tasks graph, their arrival rate and time were extracted by profiling the HW/SW system. Fig. 3 and 4 show the synthesized bus widths and number of buses for a multimedia application. For an average task arrival rate 13 and inter-task arrival time 0.73, three buses with bus widths 24, 32, and 48 are required to meet the real-time constraints. The mean supply voltage $\mu_{V_{dd}}$ and body bias voltage $\mu_{V_{bs}}$ are 1.31V and -0.33V respectively. However, for another scenario with task arrival rate 17 and inter-task arrival time 0.59 the synthesized buses of Fig. 3 is not robust due to the overlaps among the communication tasks. Thus, an additional bus of 32-bit wide is required as shown Fig. 4, which is robust for a given distribution of task arrival rates, inter-task arrival times, and the variation in data size. For the synthesized bus architecture of Fig. 4, the mean supply voltage $\mu_{V_{dd}}$ and body bias voltage $\mu_{V_{bs}}$ are 1.28V and -0.38V, respectively. In order to utilize the bus architecture effectively over time, a dynamic voltage scaling technique is used when the workload is low or the timing slack is high. Further, the memory architecture is synthesized based on the algorithm presented in [13]. The memory synthesis algorithm is based on clique partitioning of a data dependency graph.

Summarizing the experiments, we synthesized bus architectures for automatically generated tasks and real-life multimedia applications incorporating both random tasks arrival time and task arrival rate. The

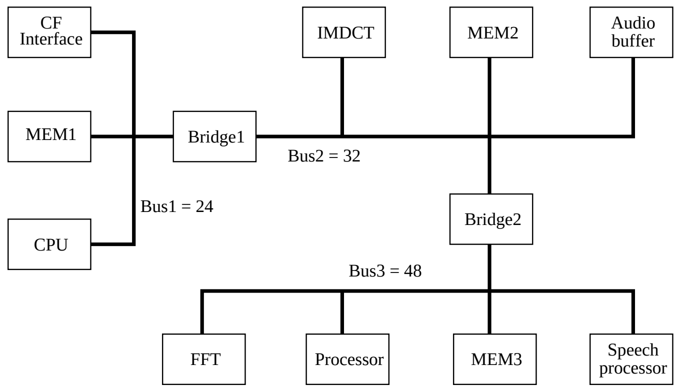

Fig. 3. Bus widths and number of buses for the real-life multimedia application with an average task arrival rate λ_n = 13 and inter-task arrival time λ_τ = 0.73

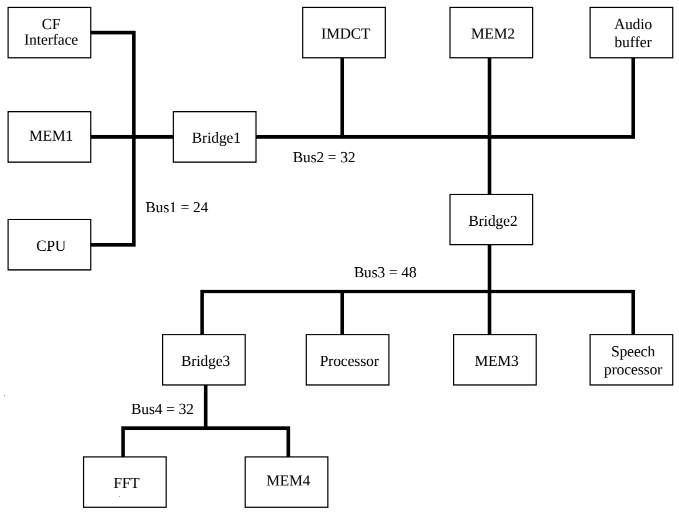

Fig. 4. Bus widths and number of buses for the real-life multimedia application with an average task arrival rate λ_n = 16 and inter-task arrival time λ_τ = 0.59

synthesis results in compare to the tasks with a deterministic arrival rate and arrival time (i.e., λ_n = 19 and λ_τ = 1.0 in the last row) show that the bus width and the number of buses change for different arrival rates and arrival times. Thus, the bus synthesis technique without considering those parameters fails to meet the real-time constraints.

VIII. Conclusion

In this paper, we proposed a robust on-chip bus architecture synthesis technique in presence of random on-chip tasks. The term robust means that the synthesized bus architecture meets the real-time constraints for different scenarios. The task arrival rate, the inter-task arrival time, and the data size are modeled as random variables with known probability distribution function. The bus architecture synthesis technique is formulated as scheduling, allocation, and binding problems. Once correctly formulated these problems are solved with the help of an optimization tool, which finds the optimal bus widths and the number of buses for a robust on-chip communication. The dynamic voltage scaling technique is used to reduce the energy consumption and to utilize the bus resources effectively. The experiments conducted on the automatic generated tasks and the real-life multimedia applications validate the effectiveness of the proposed technique under random on-chip tasks.

As part of future work, we intend to apply dynamic re-configuration of the communication bus topology so that the bus resources utilization factor can be improved effectively.

Acknowledgement

The authors would like to acknowledge and thank Tudor Murgan for the fruitful discussions.

References

[1] http://www.arm.com/products/solutions/AMBA_Spec.html.
[2] http://www.xiph.org.
[3] http://www.speech.cs.cmu.edu/sphinx/.
[4] http://www-device.eecs.berkeley.edu.
[5] http://www.mosek.com/documentation.html#manuals.
[6] T. F. Abdelzaher, V. Sharma, and C. Lu. A utilization bound for aperiodic tasks and priority driven scheduling. *In IEEE Tran. on Computers*, Vol. 53(No. 3):334–350, 2004.
[7] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. M. A. Hashimi. Simultaneous communication and processor voltage scaling for dynamic and leakage energy reduction in time constrained systems. In *proc. of ICCAD*, pages 362–369, 2004.
[8] M. Gasteier and M. Glesner. Bus-based communication synthesis on system level. *In ACM Tran. of design automation electronic systems*, pages 1–11, 1999.
[9] D. Lyonnard, S. Yoo, A. Baghdadi, and A. A. Jerraya. Automatic generation of application specific architectures for heterogeneous mpsoc. In *proc. of DAC*, pages 518–523, 2001.
[10] S. Martin, K. Flautner, T. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for low power microprocessor under dynamic workloads. In *proc. of ICCAD*, pages 721–725, 2002.
[11] Z. Ming. Architecture exploration for speech-feature-extraction acceleration. *Bachelor thesis, Institute of Microelectronics Systems, Darmstadt University of Technology, Darmstadt, Germany*, September 2005.
[12] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. Studies in Applied Mathematics, 1994.
[13] S. Pandey, C. Genz, and R. Drechsler. Co-synthesis of custom on-chip bus and memory for mpsoc architectures. In *proc. of IFIP Int. Conf. on very large scale integration (VLSISoC)*, pages 304–307, 2007.
[14] S. Pandey and M. Glesner. Statistical on-chip communication bus synthesis and voltage scaling under timing yield constraint. In *proc. of DAC*, pages 663–668, 2006.
[15] S. Pandey and M. Glesner. Simultaneous on-chip bus synthesis and voltage scaling under random on-chip data traffic. *In IEEE Tran. on VLSI systems*, Vol. 15(No. 10):1111–1124, Oct. 2007.
[16] S. Pasricha, N. Dutt, E. Bozorgzadeh, and M. Ben-Romdhane. Fabsyn:floorplan-aware bus architecture synthesis. *In IEEE Tran. on VLSI systems*, Vol. 14(No. 3):241–253, 2006.
[17] A. Pinto, L. P. Carloni, and A. V. Singiovanni. Constraint driven communication synthesis. In *proc. of DAC*, pages 783–788, June 2002.
[18] K. K. Rye and V. MooneyIII. Automated bus generation for multiprocessor soc design. *In IEEE Tran. on CAD*, Vol. 23(No. 11):1531–1549, Nov. 2004.
[19] K. Sekar, K. Lahiri, A. Ragunathan, and S. Dey. Flexbus:a high performance system-on-chip communication architecture with a dynamically configurable topology. In *proc. of DAC*, pages 571–574, 2005.
[20] N. H. E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design*. Addison wesley, 1994.
[21] T. Y. Yen and W. Wolf. Communication synthesis for distributed embedded systems. In *proc. of ICCAD*, pages 288–294, 1995.