# Statistical On-Chip Communication Bus Synthesis and Voltage Scaling Under Timing Yield Constraint

Sujan Pandey, Manfred Glesner
Institute of Microelectronics Systems
Darmstadt University of Technology, Darmstadt, Germany
{pandey,glesner}@mes.tu-darmstadt.de

## ABSTRACT

We propose a statistical approach for minimizing on-chip communication bus width and number of buses with reduced communication energy under timing yield constraint. The *slack* is exploited to maximize sharing of buses and to reduce energy by simultaneously scaling the voltage during the communication synthesis. Because of the diversity of applications to be run on a single SoC, there exists variability of data size to be transferred among the on-chip communicating modules. This variability of data size is modeled as a normally distributed random variable. The resulting synthesis problem is relaxed to the convex quadratic optimization problem and is solved efficiently using a convex optimization tool. The effectiveness of our approach is demonstrated by applying optimization to an automatically generated benchmark and a real-life application. By varying the value of timing yield constraint, a trade-off between minimization of buses and energy reduction is explored. The experimental results show the significant reduction of communication energy with the increasing timing yield. However, the timing yield offers a limitation to minimize the size of bus width and number of buses, if the yield is increased beyond a certain limit.

**Categories & Subject Descriptors:** C.3 [Special-Purpose & Application-Based Systems]: Real-Time and Embedded.
**General Terms:** Algorithms, Design Aid.
**Keywords:** Communication Bus Synthesis, Voltage Scaling.

## 1. INTRODUCTION

Due to the increasing trend on a system complexity, there is a huge demand of communication placed by the on-chip communication traffic on the communication architecture. On the other hand, the technology scaling trend shows that the interconnect wires account for a significant fraction (up to 50% [14]) of the total energy consumption in an integrated circuit, and this fraction is only expected to grow in the future. Considering these trends, synthesizing an energy efficient on-chip communication bus is a challenging task to the system designers. The early work about communication bus synthesis focused mainly on minimizing bus width [17],[19] protocol selection [6],[13], interface synthesis [18], and synthesis of single global bus topology [9]. In [23] an automatic bus generation for a MPSoC was proposed. They generate bus for a given size of bus width considering real time constraint. In [22] a method of communication synthesis based on the library elements and constraints graph was presented. Their approach focus mainly to synthesize the communication topology for the point-to-point communication architecture. In [26] a bus model for communication in embedded systems with arbitrary topologies was proposed, where point-to-point communication is a special case for the real time application. In [15] a synthesis flow which supports shared bus and point to point connection templates was presented. In [21] a floorplan aware automated bus based communication synthesis algorithm was presented. All above approaches deal only with the real time constraint and the size, however, they do not consider the energy consumption of communication buses.

There has been already a significant amount work done in the area of system level approaches to reduce the energy of embedded systems. Dynamic voltage scaling (DVS) and adaptive body biasing (ABB) can be an option to reduce the energy consumption [10],[16],[25]. In [4], DVS and ABB techniques are used to reduce the energy for fat wires and repeaters based communication bus. However, they assume that bus width and topology of communication bus have been already identified and given to them. Another possibility to reduce communication energy is the usage of bus encoding techniques [5]. In [11] proposed shared bus splitting, which dynamically breaks down long, global buses into smaller, local segments to improve the energy consumption of bus and to meet dynamic data traffic demand. In [8] proposed a model to estimate the switching activity of communication buses and to compare the effectiveness of power optimization techniques at the system level.

All the above techniques do not take into account the voltage scaling for communication energy reduction during the communication bus synthesis process. In this paper, we propose a method to synthesize the minimum size of bus width and minimum number of buses with reduced energy under the variability of data size to be transferred by each on-chip communicating module. This variability of data size is due to the diversity of applications to be run in a single embedded system and is modeled as a normally distributed random variable. The voltage is scaled to reduce the communication energy and to meet the desired timing yield during the communication synthesis. The results show that voltage scaling can be used to reduce the communication energy, however, if the voltage is scaled to the minimum level (by increasing the timing yield constraint) the synthesized size of bus width and number of buses will not be the minimum.

To our best knowledge, this is the first attempt to solve the statistical on-chip communication bus synthesis and voltage scaling problem under timing yield constraint.

## 2. PRELIMINARIES

In this paper, we consider embedded systems which are realized as multiprocessor systems-on-a-chip (SoC). Such a system consists of several on-chip processing modules like general-purpose processor, an application specific integrated circuit (ASIC) or a field-programmable gate array (FPGA). These on-chip modules communicate with each other by transferring data through communication buses like shared buses or point-to-point connection. We assume that Hw/Sw parti-


*DAC 2006,* July 24–28, 2006, San Francisco, California, USA.

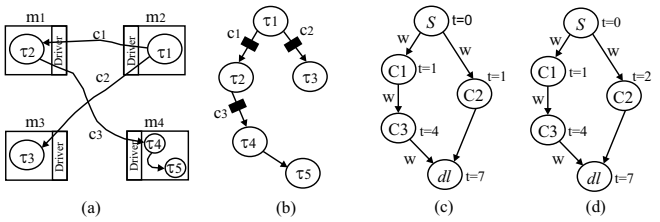


Fig. 1: Architecture model. (a) Target architecture with mapped tasks. (b) Extended tasks graph. (c) Communication task graph with ASAP scheduling of CLTIs for 16-bit wide bus. (d) Communication task graph with ALAP scheduling of CLTIs for 16-bit wide bus.

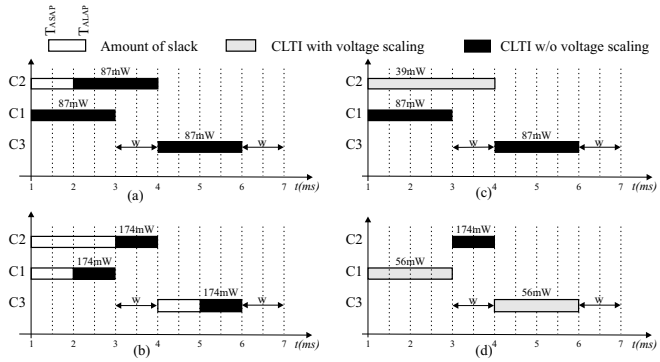


Fig. 2: Scheduling of CLTIs and voltage scaling of on-chip communication bus. (a) Scheduling of CLTIs for 16-bit wide bus. (b) Scheduling of CLTIs for 32-bit wide bus. (c) Scheduling and voltage scaling of CLTIs for 16-bit wide bus. (d) Scheduling and voltage scaling of CLTIs for 32-bit wide.

tioning and mapping of tasks onto the appropriate modules of SoCs have been done efficiently as shown in Fig. 1(a). Based on these mapped tasks, a directed acyclic extended graph $G_E(T, E)$ is obtained to extract the data processing tasks $\tau$ and the data communication tasks $c$ of a given application. In the extended graph, a node $\tau \in T$ represents the data processing task, which is mapped onto the on-chip module, while edge $e \in E$ indicates data dependency between the tasks (i.e. communication). All the communications that take place over the on-chip communication buses are captured by communication task $c_i$, as indicated by square in Fig. 1(b). If the tasks $\tau_i$ and $\tau_j$ are mapped to the same module then there exist an edge between them without the square. This indicates that the tasks $\tau_i$ and $\tau_j$ do not communicate using an on-chip communication bus. The notation $c_i$ is a communication task, which takes a certain time duration to transfer data from one module to another module by using an on-chip communication bus. This time duration is called a communication lifetime interval (CLTI), which shows how long time a task $c_i$ uses a communication bus. Furthermore, each communication task has its start time and deadline to finish the task. From the extended graph $G_E(T, E)$, a directed acyclic communication task graph $G_C(C, \Pi)$ is obtained with the start node $S$ and deadline node $dl$ to schedule the CLTIs of the communication tasks. In the communication task graph, a node $c \in C$ is a communication task, while an edge $\pi \in \Pi$ gives the dependency between the communication tasks. Fig. 1(c) depicts the communication task graph with ASAP (as soon as possible) scheduling of CLTIs for 16-bit wide bus with deadline 7ms. An edge between two nodes $c_i$ and $c_j$ weighted with $w$ is the data processing time of a task $\tau_i$, which gives an early start time constraint for a successor $c_j$ to transfer data using a communication bus. Fig. 1(d) depicts the ALAP (as late as possible) scheduling of the CLTIs for 16-bit wide bus with deadline 7ms. In Fig. 1(c) and (d), there is a difference in ASAP and ALAP time for the node $c_2$. This difference between the ALAP and ASAP time of a communication task is called *slack*. It measures how free we are to schedule the communication task $c_i$ into different time slots so as to maximize the sharing of communication buses and to minimize bus energy consumption. This *slack* is a function of three variables, which are size of data to be transferred, size of bus width and voltages. Their relation can be written as,

$$slack_{c,r,V} = t_{ALAP_{c,r,V}} - t_{ASAP_{c,r,V}} \quad (1)$$

where $t_{ALAP_{c,r,V}}$ can be expressed as,

$$t_{ALAP_{c,r,V}} = dl_c - CLTI_{c,r,V} \quad (2)$$

$$CLTI_{c,r,V} = \frac{NB_c(\zeta)}{b_r} \cdot T_c \quad (3)$$

In above equations, $dl_c$ is a deadline to finish a task, $CLTI_{c,r,V}$ is the communication lifetime interval for a task $c$ with a bus of type $r$ and supply voltage $V$, $NB_c(\zeta)$ is a random number of data bit to be transferred by a task, $b_r$ is the size of bus width of bus type $r$ and $T_c$ is the time period of one clock cycle. For the sake of clarity, we consider only the supply voltage scaling for the dynamic energy consumption. Nonetheless, the leakage energy as well as Adaptive Body Biasing (ABB) techniques [10],[16],[25],[4] can easily be incorporated into the formulation without changing our general approach. The execution time of a communication task $c$ with voltage $V_i$ is given by [16],

$$T_c = \kappa \frac{V_i}{(V_i - V_{th})^\alpha} \quad (4)$$

where $\kappa$ is a technology dependent constant, $\alpha$ is the saturation velocity ($1.4 < \alpha \leq 2$), $V_i$ is the supply voltage, and $V_{th}$ is the threshold voltage. The dynamic energy consumption of each task $c$ is given by [16],

$$E_c = \alpha_\tau \cdot C_{eff} \cdot V_i^2 \cdot T_c \quad (5)$$

where, $\alpha_\tau$ is the switching activity of the communication tasks and $C_{eff}$ is the effective switched capacitance for a data communication. The energy overhead, for switching from $V_i$ to $V_j$, is [16]

$$\varepsilon_{i,j}^{\Delta V} = C_r(V_i - V_j)^2 \quad (6)$$

where, $C_r$ is the capacitance of the power rail. The time overhead, for switching from $V_i$ to $V_j$, is given by [16]

$$\delta_{i,j}^{\Delta V} = \rho|V_i - V_j| \quad (7)$$

where $\rho$ is a constant.

## 3. MOTIVATIONAL EXAMPLE

In this section we give a motivation to scale voltage for on-chip communication bus synthesis by simultaneously performing voltage scaling, bus selection, scheduling and binding of communication tasks and illustrate that the *slack* of a communication task changes with the size of bus width and voltage. This *slack* can be exploited to scale the voltage and to share the communication bus, which ultimately increase the system efficiency in terms of number of buses/size of bus width and energy consumption. Consider a system that has been partitioned and mapped onto the on-chip modules of a SoC and the driver of each module is capable to scale the supply voltage while transferring data from one module to another module. As shown in Fig. 1(a) first, module $m_2$ executes task $\tau_1$ and its driver transfers data to $m_1$ and $m_3$ to execute tasks $\tau_2$ and $\tau_3$, respectively. After receiving the data from module $m_2$, module $m_1$ executes task $\tau_2$ and its driver transfers data to module $m_4$, which executes tasks $\tau_4$ and $\tau_5$. The task $\tau_5$ has to be finished before the deadline

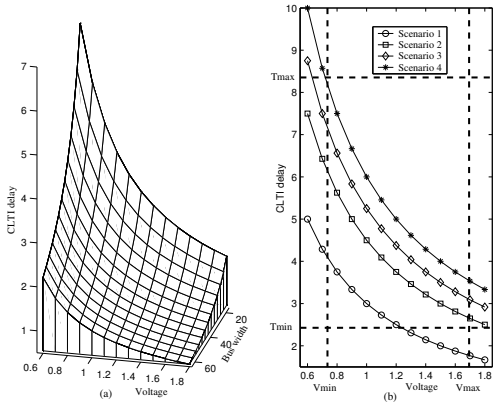


Fig. 3: Delay as a function of bus width and voltage for different scenarios

of 7ms. The ASAP and ALAP scheduling of communication task graph of above example with their start node and deadline node are shown in Fig. 1(c) and (d), respectively. Fig. 2(a) shows a scheduling of CLTIs with their ASAP and ALAP time of all the communication tasks $c_1$, $c_2$ and $c_3$, considering 16-bit wide bus and the nominal voltage settings (the highest supply voltage = 1.8V and body bias voltage = 0V), i.e., all the drivers run at their highest performance. This schedule of communication tasks for a 16-bit wide bus gives a *slack* (denoted by white rectangle) of 1 ms and needs two separate buses to meet the time constraint of 7ms. From the given power consumption at the nominal voltage as shown in Fig. 2(a), the total energy consumption of all the communication tasks can be calculated as 3·87mW·2ms = 522$\mu$J. Fig. 2(b) shows the scheduling of the same communication tasks $c_1$, $c_2$ and $c_3$ for 32-bit wide bus and the nominal voltage. This schedule gives the total *slack* of 4ms, which increases the mobility so that all the communication tasks can share a single bus. The total energy consumption at the nominal voltage can be calculated as, 169mW·3ms = 507$\mu$J. In order to reduce the energy consumption, we scale the voltage to exploit the *slack* of communication tasks so that all the tasks $c$ share the minimum number of bus as shown in Fig. 2(c) and (d). To make the problem simple, we assume in this example that the task processing time of each on-chip module is known to us, i.e, the operating voltages of modules are known. Further, we assume that the supply voltage of all the drivers can be varied continuously in the ranges [0.6, 1.8]V. In Fig. 2(c), communication tasks $c_2$ is scheduled with the supply voltage 1.4V to exploit the *slack* of 1ms, while tasks $c_1$ and $c_3$ are scheduled with the nominal voltage. The total energy consumption of the communication tasks is 39mW·3ms+(87+87)mW·2ms = 465$\mu$J, which is reduction in energy by 11% compared to the energy at the nominal voltages of 16-bit wide bus. In Fig. 2(d), the amount of *slack* is increased to 4ms by scheduling the communication tasks for 32-bit wide bus. This *slack* is exploited to scale the voltage and share the minimum number of bus. Hence, the supply voltage of the communication tasks $c_1$ and $c_3$ are scaled to 1.2V, while $c_2$ is kept to the nominal voltage to share a single bus. The total energy consumption is calculated as 174mW·1ms+ 2·56mW·2 = 398$\mu$J, which is reduction in energy consumption by 24% compared to the scheduling of Fig. 2(c).

In above example, optimal size of bus width, number of buses and voltages were found for a fixed size of data $NB_c$ of each task $c$. Due to the diversity of applications to be run is a single embedded system, a task $c$ can have a random size of data $NB_c(\zeta)$ to be transferred. For this scenario, above synthesized bus and voltage may not be optimal because of random data size of on-chip modules. Fig. 3(a) depicts $CLTI$ delay of a task $c$ as a function of voltage and bus width considering one scenario. Fig. 3(b) shows change in $CLTI$ delay of a task $c$ for different size of data to be transferred with time and voltage constraints $\{T_{min},T_{max}\}$ and $\{V_{min},V_{max}\}$, respectively. In Fig. 3(b) different size of data are plotted as a scenario, which has a certain probability. We saw in above example that the *slack* of task $c$ can be exploited to share the bus and to scale the voltage, however for the random data size of each task $c$ the size of *slack* is not deterministic. Hence, we model the communication bus synthesis together with the voltage scaling problem in a stochastic nonlinear programming and find an energy efficient optimal size of bus width and number of buses under the timing yield constraint.

## 4. PROBLEM FORMULATION

We assume that a set of tasks have been partitioned and mapped efficiently onto the appropriate modules of a SoC. Each module $m_i$ processes tasks and a communication task $c$ transfers data to another module $m_j$, which has the data dependency. The transfer of data from one module to another module takes place via a communication bus and this bus is driven by a driver which is capable to scale the voltage during each data transfer. Due to the diversity of applications that run within a single SoC, the workload offered on the embedded system is not uniform over the time. This introduces the randomness on the size of data to be transferred among the on-chip communication tasks. We model the size of data bit to be transferred by a communication task $c$ as a random variable $NB_c(\zeta)$ with a known probability distribution function. For each task $c$ its deadline $dl_c$, the distribution of random variable $NB_c(\zeta)$ and the switched capacitance $C_{eff}$ are given. Based on the mapped tasks $\tau$, a directed acyclic extended graph $G_E(T,E)$ is obtained as shown in Fig. 1. From the extended graph $G_E(T,E)$, the communication task graph $G_C(C,\Pi)$ is obtained with start node $S$ and deadline node $dl$. In the communication task graph $G_C(C,\Pi)$, $c \in C$ be a set of communicating tasks and their data dependency between the communication tasks is defined by a set $\Pi \subseteq (C \times C)$, consists of two-tuples $(c_i, c_j)$ where a successor $c_j$ depends on the results of the predecessor $c_i$. This data dependency between communication tasks is constrained by a set $Depn \subseteq (C \times C \times W)$ consists of 3-tuples $(c_i, c_j, w)$ such that $\forall i,j \in [1 \ldots N]$, $(c_i,c_j)_{i \neq j} \in \Pi | \Pi \subseteq C \times C$, a task $c_j$ can start transferring data no earlier than $w$ time units after the completion of transferring data by $c_i$.

We assume that the supply voltage $V_{dd}$ and the body bias voltage $V_{bs}$ of each data processing task $\tau_i$ are known and provided to calculate the execution time of task(s) in a module. Unlike this, the supply voltage $V$ and the body bias voltage $V_{bs}$ of each communication task $c \in C$ are unknown and to be identified. In this work for the sake of clarity, we consider only the supply voltage scaling however, adaptive body biasing (ABB) can easily be incorporated in our approach of communication bus synthesis. Each task $c \in C$ can vary its supply voltage $V$ within a certain continuous or discrete ranges.

## 5. STATISTICAL COMM. TASKS MODELING

The relation of the execution time $CLTI$ with the size of bus width $b_r$, supply voltage $V_i$, and a random size of data $NB(\zeta)$ of a communication task $c$ are given by the Eqs. (3) and (4). In this section, we assume that $CLTI$ is inversely proportional to $V_i$ ($V_{th} = 0, \alpha = 2$) to make the illustration of our point simpler, however the drawn conclusions are valid in general. After the simple algebraic manipulation of Eqs. (3) and (4) we get,

$$CLTI_{c,r,V} = \kappa \frac{NB_c(\zeta)}{b_r \cdot V_i} \tag{8}$$

We assume that $\eta$ be the timing yield of all communication tasks $c$. The yield constraint gives a limit to scale the voltage for the *slack* exploitation. We further assume that the timing yield is constrained by the range $0.5 < \eta \leq 1$. The overall

delay constraint of tasks $c$ can be written as,

$$\sum_{c \in C} P(dl_c - CLTI_{c,r,V} - \delta_{i,j}^{\Delta V} \geq 0) \geq \eta \quad (9)$$

This equation constraints the probability of all tasks $c$ having a delay less than the deadline of the tasks to be more than $\eta$. We assume that the random data model of all communication tasks $c$ are normally distributed. Hence, the mean $\mu$ and $\sigma$ of $CLTI$ as a function of $b_r$ and $V_i$ can be written as,

$$\sum_{c \in C} P((dl_c - CLTI_{c,r,V} - \delta_{i,j}^{\Delta V}) \sim N(\mu_{CLTI}(NB), \sigma_{CLTI}(NB))) \geq 0) \geq \eta \quad (10)$$

which can be rewritten as

$$\sum_{c \in C} -\mu_{CLTI}(NB) - \phi^{-1}(1-\eta)\, \sigma_{CLTI}(NB) \leq 0 \quad (11)$$

where $\phi^{-1}(\cdot)$ is the inverse of the error function. Eqs. (11) can be consider as a convex function under the condition that $\eta > 0.5$ [12]. Since the target yield for a given path is always much greater than 50%, this condition is easily satisfied.

## 5.1 Statistical Parameters Estimation

Applying the above random data model of each communication task to the stochastic programming, the optimal number of bus(es) and size of bus width are obtained. After an algebraic manipulation of Eqs. (8) the statistical parameters of voltage in terms of optimal size of bus width $b_r(opt)$ can be written as,

$$\hat{V}_i(\mu, \sigma) = \kappa \frac{NB_c(\zeta)}{CLTI_{c,r,V} \cdot b_r(opt)} \quad (12)$$

where, $NB \sim N(\mu_1(c), \sigma_1(c))$ and $CLTI \sim N(\mu_2(c), \sigma_2(c))$ are normally distributed random variables. Eqs. (12) is the quotient of two random variables, the statistical parameters of voltage can be estimated as [20],

$$\hat{\mu}_{V_i}(c) \cong \frac{\mu_1(c)}{\mu_2(c)} - \frac{1}{\mu_2^2(c)} \rho_{12}(c) \cdot \sigma_1(c)\sigma_2(c) + \frac{\mu_1(c)}{\mu_2^3(c)} \sigma_2^2(c) \quad (13)$$

similarly, the standard deviation of voltage can be estimated as,

$$\hat{\sigma}_{V_i}(c) \cong \frac{\sigma_1^2(c)}{\mu_1^2(c)} + \frac{\mu_1^2(c)}{\mu_2^2(c)} \sigma_2^2(c) - 2\rho_{12} \frac{\mu_1(c)}{\mu_2^3(c)} \sigma_1(c)\sigma_2(c) \quad (14)$$

where, $\mu_2^2(c)$ and $\mu_2^3(c)$ are the $2^{nd}$ and $3^{rd}$ central moments of each communication task $c$, respectively. In Eqs. (13) and (14), $\rho_{12}$ is the cross correlation between two random variables. Since, $CLTI$ and $NB$ are the dependent variables their cross correlation is equal to unity, i.e. $\rho_{12}$=1. The moments of random variables can be expressed as,

$$\mu_x^n = E\{(x-\mu)^n\} = \sum_{-\infty}^{\infty} (x-\mu)^n \cdot m(x) \quad (15)$$

where, function $m(x)$ is a probability mass function and $\mu$ is the mean of a random variable $x$. Similarly, mean of the *slack* $\hat{\mu}_{Slack}$ can be estimated as,

$$E[Slack] = dl_c - \frac{\kappa}{b_{opt} \cdot V_i} E[NB(\zeta)] - t_{ASAP_c} \quad (16)$$

# 6. BUS SYNTHESIS AND OPTIMIZATION

The stochastic nonlinear programming formulation for simultaneously voltage scaling, bus selection, scheduling, binding of communication tasks $c$ is given as follows:
Minimize:

$$\sum_{r \in R} Cost_r \cdot b_r \quad (17)$$

Where, $r \in R$ is a library of on-chip communication buses type, for example, bus of 16, 20, 24, $\cdots$, 128-bit wide. The $Cost_r$ of bus type $r$ is expressed in terms of bus width, like the cost of 32-bit wide bus is double the cost of 16-bit wide bus. The objective is to minimize total cost of buses by maximizing the sharing of buses among the communication tasks.
subject to,

$$\forall c \in C, \sum_{r \in R} \sum_{t=ASAP_c}^{\Psi} \sum_{V=V_{max}}^{V_{min}} X_{c,t,r,V} = 1 \quad (18)$$

$$\Psi = (ALAP_c + d_{min_c} - CLTI_{c,r,V} - \delta_{i,j}^{\Delta V}) \quad (19)$$

The variables those need to be determined in the formulation are the bus $b$ of type $r$, and binary decision variable $X_{c,t,r,V}$. The binary variable indicates scheduling of a communication task $c_i$ at time $t \in \{0, \cdots, \lambda\}$, bus type $r$ and with supply voltage $V$, respectively. Where the term $\delta_{i,j}^{\Delta V}$ is the time overhead delay due to switching of voltage from $V_i$ to $V_j$.

Each communication task $c \in C$ must be mapped to a single bus type $r$, operating at a single time $t$, with supply voltage $V$ as expressed in the Eqs. (18). The communication task $c \in C$ with ALAP time $ALAP_c$ cannot be executed later than $\Psi$, when the data is transferred through a bus $b$ of type $r$ with the communication lifetime interval of a task $CLTI_{c,r,V}$. The term $d_{min_c}$ is the minimum time to execute the communication task $c \in C$ to meet the deadline $dl_c$.

We introduce a set $\Omega$, which represents the set of all times that any communication task could possibly start at,

$$\Omega = \bigcup_{c \in C} \{ASAP_c, \cdots, ALAP_c\} \quad (20)$$

$$\forall t \in \Omega, \forall r \in R, \sum_{c \in C} \sum_{\substack{(t' \in \{t, \cdots, t+d_r-1\} \\ \cap \{ASAP_c, \cdots, \psi)}} \sum_{V=V_{max}}^{V_{min}} X_{c,t',r,V} \leq b_r \quad (21)$$

No communication bus $b$ of type $r$ can execute more than one communication task at a time $t$ with voltage $V$ is expressed in constraint Eqs. (21). The first summation is over all the communication tasks with bus type $r$ and the second summation is over a "time window" covering all start time $t^{'}$ for which the communication tasks could overlap.

$$\forall (c^{'}, c) \in \Pi, \ \forall (c^{'}, c, w) \in Depn, \sum_{r \in R} \sum_{\substack{t= \\ ASAP_c}}^{\Psi} \sum_{V=V_{max}}^{V_{min}} t \cdot X_{c,t,r,V} \geq \sum_{r \in R} \sum_{\substack{t= \\ ASAP_{c'}}}^{\Psi'} \sum_{V=V_{max}}^{V_{min}} (t + CLTI_{c',r,V} + w + \delta_{i,j}^{\Delta V}) \cdot X_{c',t,r,V} \quad (22)$$

$$\Psi^{'} = (ALAP_{c'} + d_{min_{c'}} - CLTI_{c',r,V} - \delta_{i,j}^{\Delta V}) \quad (23)$$

The data dependency between the communication tasks is expressed as in Eqs. (22). The term in the right hand side of the equation expresses that the communication task $c$ should be executed only after the execution time $\Psi^{'}$ of task $c^{'}$. The delay $w$ is a fixed time delay between two communication tasks to execute the data processing task $\tau_i$.

$$V = \{V_1, V_2 \cdots, V_n\} \quad (24)$$

The *slack* of each communication task $c \in C$ is exploited to share a communication bus and reduce the total energy consumption by scaling the supply voltage $V$ for each communication task from the nominal voltage. The possible discrete

voltages are constrained by the Eqs. (24).

$$\forall\ c \in C,\ P((\sum_{r \in R} \sum_{\substack{t= \\ ASAP_c}}^{\Psi} \sum_{V=V_{max}}^{V_{min}} (dl_c - t - CLTI_{c,r,V} - \delta_{i,j}^{\Delta V}) \cdot X_{c,t,r,V}) \geq 0) \geq \eta \quad (25)$$

The summation of deadline $dl_c$, start time $t$, the data transfer time (CLTI) and the delay overhead due to voltage switching $\delta_{i,j}^{\Delta V}$ of each communication task $c \in C$ should be greater than or equal to zero with a probability $\eta$ for on-chip bus $b$ of type $r$, supply voltage $V$ as shown in Eqs. (25).

The above communication bus optimization problem has constraints, which are nonlinear and formalized as a convex quadratic optimization problem. The problem of simultaneous communication bus synthesis and voltage scaling is similar to the discrete time-cost trade-off problem, which is known to be a NP-complete [7]. We also propose a heuristic using a linear relaxation method with limited number of discrete voltages so that the problem can be optimized with a quasi-polynomial time complexity [24]. Since the convex quadratic constraint is a convex function for $\eta > 50\%$, they guarantee a globally optimal solution [12].

## 7. EXPERIMENTAL RESULTS

We evaluate the effectiveness of the proposed techniques using generated benchmark as well as a real-life application, namely speech recognition system [1]. The automatically generated benchmark consists of 119 communication tasks $c$ and data to be transferred by all tasks $c$ are normally distributed with mean $\mu_c(NB)$=128 bit. Different level of variability in data size $NB_c(\zeta)$ were explored ranging from 2% to 30% of $3\sigma$. The deadline $dl_c$ of each task $c$ is deterministic and it is different for different values of $\sigma_c(NB)$. The data processing time $w$ of each task $\tau$ are given for each pair of communication tasks. We took the maximum value of $w$ of each task $\tau$ assuming that the on-chip modules are capable to scale the voltages for changing data traffic. Each communication task $c$ can scale the supply voltage ranging from 1.8V to 0.6V, to meet the desired timing yield. The on-chip communication buses are given as a library of buses with different size of bus width, which ranges from 16 to 128 bit wide with an increment of 4-bit. For the experiment purpose, we consider a bus with $4mm$ in length and its corresponding single line capacitance for $0.07\mu m$ technology is $609fF$. These technology dependent parameters were adopted from [16].

The first set of experiment was conducted to synthesize the minimum size of bus width and number of buses with reduced communication energy, using voltage scaling technique. We performed simultaneous voltage scaling (continuous), bus selection, scheduling and binding of communication tasks $c$ using the proposed algorithm. The algorithm was implemented in C as a pre-processing model to interface with a convex solver of MOSEK [2]. Table 1 shows the results of optimized size of bus width and number of buses for the automatically generated tasks $c$. The table compares the size of bus width and number of buses $b_r(opt)$, mean voltage $\hat{\mu}_V$ and mean *slack* $\hat{\mu}_{Slack}$ for the different timing yield $\eta$. The results show that optimized size of bus width and number of buses change with timing yield $\eta$. In column 2 and 5 of the table, the size of $b_r(opt)$ are constant for two different values of $\eta$, however, the mean voltage $\hat{\mu}_V$ and mean *slack* $\hat{\mu}_{Slack}$ decrease in column 6 and 7, respectively. This is because of increased in timing yield of communication tasks from 79% to 89%. In column 8 of the table, there are two buses with different bus width for all values of $3\sigma$. In this case, timing yield $\eta$ of all the tasks is set to 99%, so that voltage of all the tasks $c$ are scaled to the minimum possible value. This results very small amount of *slack* of communication tasks. Note that higher the amount of *slack*, more the mobility of communication tasks $c$, which maximizes the sharing of communication bus. Hence, at the timing yield of 99%, there is very less mobility of communication tasks $c$, which results the overlap among them so that two separate buses are needed to meet the real-time constraint. Fig. 4 depicts, the estimated cumulative distribution function (CDF) of voltage for $\eta = 79\%$, 89% and $b_r(opt) = 48$ bit wide. The estimated voltages have Cauchy distribution with mean $\hat{\mu}_V$ and $\hat{\sigma}_V$ from Eqs. (13) and (14). The results show that the mean of voltage $\hat{\mu}_V$ is high in case of $\eta = 79\%$ than the value of $\eta = 89\%$.

The second part of experiment was conducted on the CMU Sphinx [1] for speech recognition, which consists of three main components front end, training and recognition. The front end includes series of data processing tasks such as the pre-emphasis, hamming window, FFT (fast fourier transformation), mel frequency filter, IFFT, cepstral mean normalization, feature extraction to generate the features from the speech. The training 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. We use the American English lexicon consisting of 32 phonemes and a database of 17 different words (spelling out name of month, numbers and digits). The length and number of phonemes in a speech depends on application to application. After the partitioning, the front end was mapped to the dedicated hardware of FFT and filters. The tasks training and recognition were mapped to the one PowerPC processor. The co-simulation was carried out in Seamless CVE [3] from Mentor Graphics. The resulting co-simulation traces were used to capture the communication tasks $c$ and obtained the communication task graph $G_c(C, \Pi)$. We considered the speech length that varies from 1.06 to 11.8 sec and depending on length, the recognition time changes. To shorten the recognition time, the FFT was configured to 256, 512 and 1024 point and burst size of 3 to 6. The data model of the communication task of the FFT $NB_{fft}(\zeta)$ was approximated as a normal distribution, with mean $\mu_{NB_{fft}} = 248$ bit and standard deviation $\sigma_{NB_{fft}} = 44$. While, the data model of the communication tasks of a processor was kept deterministic. Fig. 5 shows the results of communication bus synthesis and mean energy consumption for timing yield ranging from 79% to 99%. In this part of experiment, we performed discrete voltage scaling with possible voltages {0.6V, 1.0V, 1.2V, 1,4V, 1.6V}. A constant single bus of 48 bit wide (cost 37.5) was obtained for $\eta$ ranging from 79% to 88% in Fig. 5(b), while the mean communication energy consumption was reduced upto 57% in Fig. 5(a) by scaling the voltage. For the timing yield $\eta > 88\%$, the amount of *slack* is less, which offers less mobility of communication tasks $c$ to share the same communication bus. Hence, for the $\eta > 88\%$, two buses of 24 and 32 bit wide (cost 43.7) was obtained as shown in Fig. 5(b).

Summarizing the experiments, the trade-off between minimization of buses and energy reduction was explored by varying the timing yield, during the communication synthesis. We have seen that increasing the timing yield $\eta$ can help to reduce the energy consumption, however, if the value of $\eta$ increases from a certain limit, the mobility of communication tasks will reduce and results additional some more buses to meet the real-time constraint. So, the timing yield $\eta$ can be used as a tuning factor to synthesize the minimum size of bus width and minimum number of buses with the reduced communication energy consumption.

## 8. CONCLUSION

In this paper, we have proposed an algorithm to synthesize an energy efficient minimum size of bus width and minimum number of buses in presence of random data size of on-chip communication tasks. The size of data to be transferred by communication tasks is modeled as a normally distributed random variable. The *slack* is exploited to maximize the sharing of buses and to reduce the energy by simultane-

| $3\sigma(NB)$ | Timing yield $\eta$=79% | | | Timing yield $\eta$=89% | | | Timing yield $\eta$=99% | | | Run time |
|---|---|---|---|---|---|---|---|---|---|---|
| | $b_r(opt)$ | $\hat{\mu}_V$ | $\hat{\mu}_{Slack}(\%)$ | $b_r(opt)$ | $\hat{\mu}_V$ | $\hat{\mu}_{Slack}(\%)$ | $b_r(opt)$ | $\hat{\mu}_V$ | $\hat{\mu}_{Slack}(\%)$ | (sec) |
| $3\sigma$=2% | 64 | 0.93 | 59.4 | 64 | 0.81 | 41.4 | (48,24) | 0.76 | 11.2 | $\sim$ 37 |
| $3\sigma$=5% | 64 | 0.97 | 58.2 | 64 | 0.81 | 38.7 | (48,24) | 0.76 | 9.5 | $\sim$ 37 |
| $3\sigma$=7% | 60 | 1.02 | 57.9 | 60 | 0.81 | 38.3 | (36,32) | 0.76 | 9.7 | $\sim$ 37 |
| $3\sigma$=10% | 60 | 1.07 | 57.3 | 60 | 0.89 | 35.9 | (36,32) | 0.79 | 9.3 | $\sim$ 37 |
| $3\sigma$=12% | 56 | 1.10 | 55.8 | 56 | 0.89 | 35.6 | (32,32) | 0.79 | 9.8 | $\sim$ 37 |
| $3\sigma$=15% | 56 | 1.15 | 54.4 | 56 | 0.92 | 34.5 | (32,32) | 0.84 | 7.9 | $\sim$ 37 |
| $3\sigma$=17% | 48 | 1.19 | 54.8 | 48 | 0.92 | 34.8 | (32,32) | 0.88 | 7.0 | $\sim$ 37 |
| $3\sigma$=20% | 48 | 1.24 | 51.1 | 48 | 1.12 | 32.6 | (32,32) | 0.92 | 7.1 | $\sim$ 37 |
| $3\sigma$=22% | 36 | 1.27 | 49.3 | 36 | 1.12 | 32.9 | (32,16) | 0.97 | 6.3 | $\sim$ 37 |
| $3\sigma$=25% | 36 | 1.32 | 49.7 | 36 | 1.23 | 31.1 | (32,16) | 1.03 | 6.4 | $\sim$ 37 |
| $3\sigma$=27% | 32 | 1.33 | 48.6 | 32 | 1.12 | 31.4 | (32,16) | 1.10 | 6.5 | $\sim$ 37 |
| $3\sigma$=30% | 32 | 1.35 | 48.1 | 32 | 1.27 | 30.2 | (32,16) | 1.13 | 6.6 | $\sim$ 37 |

Table 1: Synthesize bus(es) and mean voltage for different $\eta$ and $3\sigma$ with continuous voltage scaling

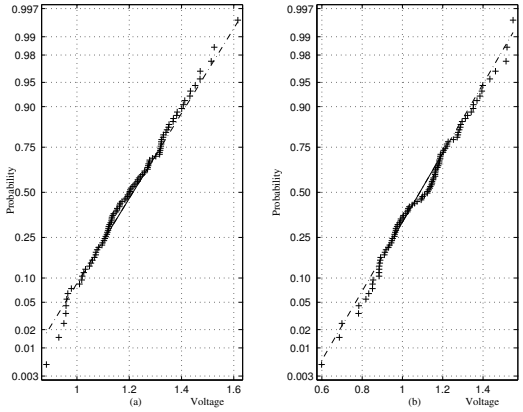


Fig. 4: Cumulative distribution function (CDF) of voltage (a) For $3\sigma$=20% and $\eta$=79%. (b) For $3\sigma$=20% and $\eta$=89%.

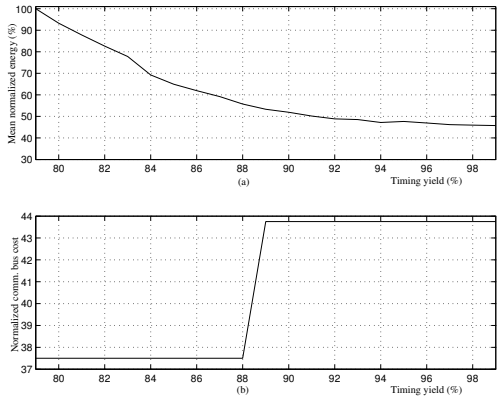


Fig. 5: Tuning of timing yield (a) Mean normalized energy (b) Normalized comm. bus cost as a function of timing yield.

ously scaling the voltage during the communication synthesis process, under the timing yield constraint. The experimental results carried out on automatically generated benchmark and a real-life application show that a significant amount of communication energy can be reduced by applying voltage scaling technique. However, the voltage scaling technique has some limitation to find the minimum size of bus width and minimum number of buses, if the timing yield is increased beyond a certain limit.

# 9. REFERENCES


[1] http://www.speech.cs.cmu.edu/sphinx/.

[2] http://www.mosek.com/documentation.html#manuals.

[3] http://www.mentor.com.

[4] A. Andrei and et al. Simultaneous communication and processor voltage scaling for dynamic and leakage energy reduction in time constrained systems. In *proc. of ICCAD*, 2004.

[5] L. Benini and et al. Address bus encoding techniques for system level power optimization. In *proc. of DATE*, 1998.

[6] J. Daveau and et al. Protocol selection and interface generation for hw/sw codesign. *In IEEE Tran. on VLSI Systems*, Vol. 5(No. 1), 1997.

[7] P. De, E. Dunne, J. Ghosh, and C. Wells. Complexity of the discrete time-cost trade off problem for project networks. *Operation research*, vol. 45(2):302–306, March 1997.

[8] W. Fornaciari, D. Sciuto, and C. Silvano. Power estimation for architecture exploration of hw/sw communication on system level buses. In *proc. CODES*, 1999.

[9] M. Gasteier and et al. Bus-based communication synthesis on system level. *In ACM Trans. design automation ele. sys.*, 1999.

[10] F. Gruian and K. Kuchcinski. Lenes: Task scheduling for low energy systems using variable supply voltage processors. In *proc. ASPDAC*, 2001.

[11] C. Hsieh and M. Pedram. Architectural energy optimization. *In IEEE TCAD of Integrated Circuits and Systems*, Vol. 21(No. 4), April 2002.

[12] P. Kall and S. Wallace. *Stochastic programming.*

[13] K. Lahiri, A. Raghunathan, and S. Dey. Design space exploration for optimizing on-chip communication architecture. *In IEEE TCAD of Integrated Circuits and Systems*, Vol. 23(No. 6):952–961, June 2004.

[14] D. L. Liu and et al. Power consumption estimation in cmos vlsi chips. *In IEEE Journal SSC*, vol. 29(6):1531–1549, June 1994.

[15] D. Lyonnard and et al. Automatic generation of application specific architectures for heterogeneous multiprocessor soc. In *proc. of DAC*, 2001.

[16] S. Martin and et al. Combined dynamic voltage scaling and adaptive body biasing for low power microprocessor under dynamic workloads. In *proc. of ICCAD*, pages 721–725, 2002.

[17] S. Narayan and D. Gajski. Synthesis of system level bus interfaces. In *proc. of DATE*, 1994.

[18] S. Pandey and et al. High level hw/sw communication estimation in shared memory architecture. In *proc. of Int. symposium on circuit and systems (ISCAS)*, 2005.

[19] S. Pandey and et al. Performance aware on-chip communication synthesis and optimization for shared multi-bus based architecture. In *ACM proc. of SBCCI*, 2005.

[20] A. Papoulis and S. U. Pillai. *Probability, random varibles and stochastic processes.* Mc Graw Hill, fourth edition.

[21] S. Pasricha and et al. Floorplan aware automated synthesis of bus based communication architectures. In *proc. DAC*, 2005.

[22] A. Pinto, L. P. Carloni, and A. V. Sangiovanni. Constraint driven communication synthesis. In *proc. of DAC*, June 2002.

[23] K. K. Rye and V. MooneyIII. Automated bus generation for multiprocessor soc design. *In IEEE TCAD of Integrated Circuits and Systems*, Vol. 23(No. 11):1531–1549, Nov. 2004.

[24] M. Skutella. Approximation algorithms for the discrete time-cost trade-off problem. *Mathematics of operation research*, vol. 23(4):909–929, Nov. 1998.

[25] L. Yan, J. Luo, and N. Jha. Joint dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real time embedded systems. *In IEEE TCAD of Integrated Circuits and Systems*, Nov. 2005.

[26] T. Y. Yen and W. Wolf. Communication synthesis for distributed embedded systems. In *proc. of ICCAD*, 1995.