Performance Aware On-Chip Communication Synthesis and Optimization for Shared Multi-Bus Based Architecture

Sujan Pandey*
Institute of Microelectronics Systems
Darmstadt University of Technology
Karlstr. 15, D-64283
Darmstadt, Germany
pandey@mes.tu-darmstadt.de

Manfred Glesner
Institute of Microelectronics Systems
Darmstadt University of Technology
Karlstr. 15, D-64283
Darmstadt, Germany
glesner@mes.tu-darmstadt.de

Max Mühlhäuser
Telecooperation Group
Darmstadt University of Technology
Hochschulstr. 10, D-64283
Darmstadt, Germany
max@informatik.tu-darmstadt.de

ABSTRACT

This paper presents a method of on-chip communication topology synthesis and optimization for a shared multi-bus based architecture. An assumption for the synthesis is that the system has already been partitioned and mapped onto the appropriate components of a SoC so that size of data to be transferred at each time by an on-chip module is fixed. We model the communication behavior of each module as a set of communication lifetime intervals (CLTIs), which are optimized in terms of number of overlaps among them, size of bus width and the minimum number of buses, using ILP (integer linear programming) formulation. We synthesize the communication topology and further optimize the architecture based on the intermodule communication statistics, which are obtained from the system level profiling of an application. The result of applying this approach to the Talking Assistant used in ubiquitous computing application demonstrates the utility of our techniques to synthesize the communication architecture for a complex system.

Categories and Subject Descriptors

B.7 [**Integrated Circuits**]: Design Aids; C.3 [**Special Purpose and Application System**]: Real-time and embedded systems

General Terms

Design, Algorithms

*The authors' work is supported by the Deutsche Forschungsgemeinschaft (DFG) as a part of the Ph.D. program "System Integration for Ubiquitous Computing in Information Technology" at the Darmstadt University of Technology, Darmstadt, Germany.

Keywords

On-chip communication architecture synthesis, Optimization, Algorithms

1. INTRODUCTION

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

We formulate the problem of synthesizing on-chip communication architecture for a partitioned and mapped Hw/Sw system into mainly three steps: 1) identifying an on-chip communication topology and the size of bus width; 2) defining communication protocol; 3) synthesis of communication interface. In this work, we focus on the first step of communication synthesis design flow, assuming that the size of data to be transferred from one module to other module is fixed (since the hardware is already synthesized). We model the communication behavior of an on-chip communicating module by its communication lifetime intervals (CLTIs), which consists of early start time and early finish time to transfer a certain size of data in a session. Where the term session is a periodic time interval that includes ASAP (As Soon As Possible) with hardware constraint scheduling of all the possible CLTIs of on-chip modules, which will be repeated in all the sessions as shown in Fig. 1(a) and (b). The CLTIs of modules in a session can overlap with each other, when one try to communicate while other is communicating through the bus. We minimize the number of overlapping among the CLTIs with a help of integer linear programming (ILP) formulation so that the optimal size of bus width and maximum utilization of bus among the on-chip modules will be achieved. After the ILP formulation, the optimized CLTIs of on-chip modules are given to the communication topology synthesis algorithm, which gives the number of bus and their connection with modules. To show the feasibility of our approach, we present a case study to synthesize on-chip communication topology for a mobile device, which is used for the ubiquitous computing application called Talking Assistant [3]. We evaluate CLTIs of each on-chip module for different value of bus width and optimize them to get the minimum number of overlaps among the CLTIs. From these optimized CLTIs, we synthesize the communication topology using the clique partitioning algorithm [9].

The remainder of this paper is organized as follow. In section 2, we describe related work and compare our approach with current approaches. In section 3, we give a brief explanation about the graph terminology. In section 4, we formulate the problem of communication topology synthesis for a shared memory architecture. In section 5, we present a mathematical formulation and optimization techniques for the on-chip communication modules. In section 6, we present an experimental results to approve our method of communication topology synthesis. In section 7, we give the conclusion of this paper and possible future research.

2. RELATED WORK

Recent approaches to on-chip communication synthesis and analysis mainly focus on to get an efficient communication architecture based on optimizing the available bus templates [5],[6],[7]. In [5] Lahiri et al. propose a method to find a communication architecture after the system has been partitioned into Hw/Sw. Their approach optimize the communication architecture by mapping a system into several available communication templates, and takes the one which fulfills the requirement best. This approach focus mainly the problem of finding the optimized communication architecture assuming bus topology has been already identified.

In [8] Ryu et al. describe an automatic bus generation for a multiprocessor SoC for real time applications. Their approach consider for five different types of bus, which can be generalized to shared bus, point-to-point and FIFO based architecture. They generate bus for a given size of bus width considering real time constraint. But, they do not consider the effect of different size of bus width to the communication synthesis process. In our approach, we synthesize the bus topology by finding an optimal value of bus width.

In [7] Pinto et al. present a method of communication synthesis based on the library elements and constraints graph. Where the library element is a collection of communication links and communication nodes. Their approach focus mainly to synthesize the communication topology for the point-to-point communication architecture.

In [10] Yen et al. propose a bus model for communication in embedded systems with arbitrary topologies in which point-to-point communication is a special case for the real time application. Their algorithm selects the number of buses, the type of each bus, the message transferred on each bus and the schedules the bus communication.

In [4] Gasteier et al. describe the automatic generation of a communication topology by using scheduling of data transfer operations to reduce the cost (e.g., area) of a bus architecture. However, their algorithm only supports a single bus topology. The aim of this work is to introduce on-chip communication synthesis and optimization techniques for multi-bus based shared memory architecture.

The aim of this work is to introduce on-chip communication synthesis and optimization techniques for multi-bus based shared memory architecture, considering intermodule communication statistics by profiling system at the system level. The presented work makes the following contributions:

- we demonstrate that the proper selection of the size of bus width influences the communication topology. For this, we evaluate and schedule (with as soon as possible) with hardware constraint) all the communication lifetime intervals of on-chip modules for different size of bus width and select one, which meets the real time constraint and gives the minimum overlap among the CLTIs.

- we model the dynamic communication behavior of on-chip modules with three parameters, which are average no. of communication, transition density and spatial correlation. These parameters are used later to refine the communication topology by moving or swapping the modules from one bus to the other bus.

3. BACKGROUND

In this section, we give a brief terminologies on graph theory, which are used in our approach of communication behavior analysis and synthesis.

3.1 Graph terminology

3.1.1 Definition 1

An overlap graph is a pair $G_o = (V, E_o)$, where a finite set $V = \{v_i | v_i \ \textit{represents an interval}\ I_i\}$, and a set $E_o = \{(v_i, v_j) | l_i < l_j < r_i < r_j\}$. The values l_i, l_j and r_i, r_j are left and right points of the interval i and j, respectively.

3.1.2 Definition 2

A containment graph $G_c = (V, E_c)$, where a finite set $V = \{v_i | v_i \ \textit{represents an interval}\ I_i\}$ and a set $E_c = \{(v_i, v_j) | l_i < l_j, r_j < r_i\}$. The values l_i, l_j and r_i, r_j are left and right points of the interval i and j, respectively.

4. PROBLEM DEFINITION

We define the communication behavior of on-chip modules by a set of their communication lifetime intervals (CLTIs) in a certain period of time, which we call a session. The CLTIs of a module is defined as a communication interval $[T_{s,m_i,k}, T_{e,m_i,k}]$, where $T_{s,m_i,k}$ is the early start time and $T_{e,m_i,k}$ is the early end time of bus usage to transfer the data by module m_i, with a generalized bus width k as shown in Fig.1(a). The bus width k is same for the whole system. This transfer can be either loading data from the shared memory or storing data to the shared memory. Intuitively, it can be seen in Fig.2 that the CLTI of each module depends on the size of data to be transferred and the size of bus width. Because of the different size of data of each module, ratio of change in communication lifetime is not the same for all. So, this makes the change in number of overlaps among the CLTIs for different size of bus width. Since on-chip modules are already synthesized and mapped, size of data to be transferred from a module is fixed for a session. The term session is a time interval, where all the possible communication events among on-chip modules are traced as CLTIs and these events will be repeated again at the same (relative) time instant in the next session. Fig.1(a) shows the ASAP (As Soon As Possible) with hardware constraint scheduling of CLTIs of all on-chip modules in a session by considering

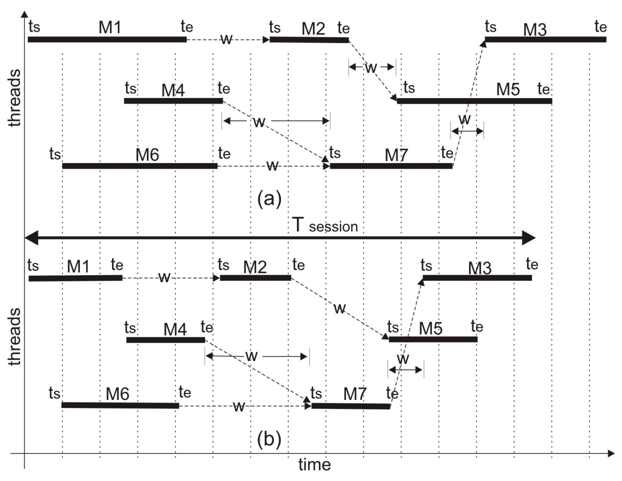

Figure 1: Communication life time interval (CLTI) of on-chip modules. (a) ASAP with hardware constraint schedule of CLTIs before applying ILP formulation; (b) schedule of CLTIs after applying ILP formulation.

size of data to be transferred and a size of bus width k (minimum of $k \in \{min, \cdots, max\}$). The dotted lines with arrow from one module to other modules in the figure show the data dependencies among them. The time interval between two communicating pairs of modules ($T_{s,m_j,k} - T_{e,m_i,k}$) gives the early start time constraint W_{m_i,m_j} for the successor m_j. This W_{m_i,m_j} is the duration of data processing time for a module, which is fixed in our case because hardware is already synthesized. Each CLTI in the figure uses communication bus to transfer data using the ASAP with hardware constraint and our objective is to minimize the number of overlaps among CLTIs so that all modules can transfer data by sharing the minimum number of buses. As we assume that communication lifetime interval is only the function of bus width, we try to shorten the size of CLTIs of each module by varying its size so that at certain size of bus width the minimum overlap and containment graph can be obtained. This gives the maximum utilization of communication resources sharing by several modules. We solve the problem of finding minimum overlap among the modules by using the integer linear programming (ILP), where the communication lifetime intervals of all modules are evaluated for different size of on-chip bus width and the optimal value of bus width is selected, which gives minimum overlap and containment graph among the CLTIs and fulfills the real time constraint $T_{session}$ as shown in Fig.1(b).

5. MODELING AND SYNTHESIS

In this section, we present a mathematical formulation for modeling the CLTI of communicating modules and their data dependencies with different size of on-chip bus width in a form of integer linear programming (ILP). After the ILP formulation of on-chip communication behavior, the minimum overlap among CLTIs and optimum size of bus width are obtained by meeting real time constraints. This optimized set of CLTIs of modules is then given to the communication topology synthesis algorithm, which synthesize the topology for on-chip communication architecture.

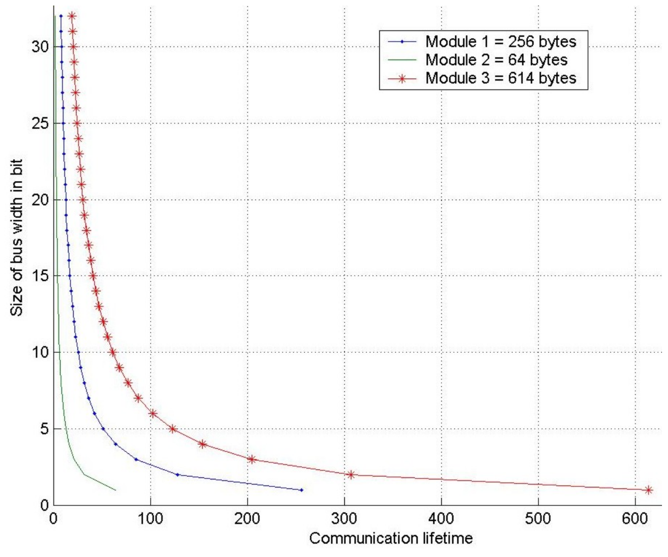

Figure 2: Relation between CLTI and the size of bus width

5.1 ILP formulation

The ILP formulation for communication synthesis approach takes a set of communicating modules with their sequence of execution (modules executes one after the other) ($Chain$), size of data to be transferred of each module for a session ($datasize(m_i)$), minimum and maximum size of bus width (k) and the data dependency of modules in terms of timing between predecessor and successor ($Depn$).

Let M be a set of communicating modules and their data dependencies between the modules is defined as a pair of modules $Pair \subseteq (M \times M)$, where a successor module depends on the output data of the predecessor module. This data dependency between modules is defined by a set $Depn \subseteq (M \times M \times W)$ consisting of 3-tuples (m_i, m_j, W_{m_i,m_j}) such that $\forall i, j \in [1 \ldots N]$, $(m_i, m_j)_{i \neq j} \in Pair | Pair \subseteq M \times M$, a module m_j can start transferring data no earlier than W time units after the completion of transferring data by m_i. We also define a set $Chain \subseteq M$ such that its elements are the set of sequential executing modules, which are derived by tracing the execution at the system level.

5.1.1 Variable

An integer variable $X_k = \{0, 1\}$ is defined such that $\forall m_i \in M$ and each bus width $k \in \{min, \cdots, max\}$, $X_k = 1$ iff the real time constraint, minimum overlap and containment graph are obtained with a certain value of bus width k.

5.1.2 Constraints

Exactly one size of bus width should be selected for all the modules in order to meet real time constraint, the minimum overlap graph and the minimum containment graph.

$$\forall m_i \in M, \sum_{k=min}^{max} X_k = 1 \qquad (1)$$

For each pair of modules (m_i, m_j), where m_i be the predecessor and m_j be the successor in terms of data dependency then an early start time $T_{s,m_j,k}$ to transfer data by m_j with size of bus width k should be not earlier than W_{m_i,m_j} time units after the completion of transferring data by m_i.

$$\forall(m_i,m_j)_{i\neq j}\in Pair,\ \sum_{k=min}^{max} T_{s,m_j,k}\cdot X_k \geq \sum_{k=min}^{max} T_{e,m_i,k}\cdot X_k + W_{m_i,m_j} \tag{2}$$

For each size of bus width $k\in\{min,\cdots,max\}$, summation of CLTI of each module $m_i \in Chain$ and summation of time constraint $W_{m_i,m_j}|(m_i,m_j)_{i\neq j}\in Pair$ should be less than or equal to the given real time constraint for a session. The start time and end time of each CLTI are bounded by their constraints as shown in eqn.(4). Where the start time $T_{s,m_i,k}$ of a module m_i should not be less than E_{m_i} and the end time $T_{e,m_i,k}$ of a module m_i should not be greater than L_{m_i}. While optimizing the number of overlaps among CLTIs, maximum number of synthesize bus should be less than or equal to the $MaxNumOfBus$ as shown in eqn. (5).

$$\forall m_i\in Chain,\ \sum_{k=min}^{max}\sum_{i=0}^{n} CLTI(T_{s,m_i,k},T_{e,m_i,k})\cdot X_k + W_{m_i,m_j}\cdot X_k \leq T_{session} \tag{3}$$

$$\forall m_i\in M,\ 1\leq i\leq n,\ T_{s,m_i,k}\not< E_{m_i}\quad T_{e,m_i,k}\not> L_{m_i} \tag{4}$$

$$\forall Chain\in M, \sum NumOfBus \leq MaxNumOfBus \tag{5}$$

The timing of each CLTI such as start time and end time for bus usages are evaluated as the expressions given bellow. All the communicating modules m_i, which do not have predecessor, are defined as a set of modules $m_i\in M$ called $StartModule$ and their start time $T_{s,m_i,k}$ is unchanged and given as input to the ILP formulation. Unlike this the modules $m_i\notin StartModule$, their timing are to be re-evaluated because of their data dependency among the predecessor and successor modules with changing size of bus width k. When a size of bus width is changed from k_1 to k_2, the timing of a module also changes because of data dependency among the modules.

Since Hw/Sw partitioning and mapping tasks have been already done, the size of data to be transferred by individual module is fixed and denoted by a variable $datasize(m_i)$. At this level of analysis we do not take into account the factors such as bus protocols overhead, bus request/grant overhead etc. for the sake of simplicity.

Where,

$$\begin{aligned}&\forall m_i\in StartModule.\\ &T_{s,m_i,k} = C\ (constant)\ which\ is\ given\\ &T_{e,m_i,k} = \frac{datasize(m_i)}{k}\end{aligned} \tag{6}$$

$$\begin{aligned}&\forall m_i\notin StartModule \land \forall(m_i,m_j)_{i\neq j}\in Pair.\\ &T_{s,m_j,k} = max(T_{e,m_i,k}+W_{m_i,m_j})_{\forall m_i\in Pair}\\ &T_{e,m_j,k} = T_{s,m_j,k}+\frac{datasize(m_j)}{k}\end{aligned} \tag{7}$$

If a communicating module m_j has more than one predecessors $m_{i\in[1\ldots N]}$ then its early start time $T_{s,m_j,k}$ to transfer data is the maximum of $T_{s,m_j,k}$ for all predecessors m_i.

5.1.3 *Objective functions*

The objective functions of the ILP formulation are to minimize the number of overlap (N_o) and the number of containment graph (N_c) of the CLTIs to maximize the sharing of communication bus among the communication modules. In our approach, we classify the type of overlapping among CLTIs into two classes containment graph and overlap graph. If the CLTIs of a module m_i is contained in the CLTIs of module m_j then there is not any possibility of sharing the communication bus between them. So, to meet the real time constraint we need to allocate a separate bus for modules m_i and m_j. But if the CLTIs of a module m_i is overlapped with the CLTIs of module m_j and their overlap interval is less than the given limit then the module m_i and m_j can share the same communication bus by storing data in a buffer of module until the bus is not available to transfer. So after having an information about the overlap and the containment graph of CLTIs, an efficient communication topology can be synthesized for a complex system. In our objective function for the communication topology synthesis we try to minimize both the containment and overlap graph of CLTIs.

$$\left\{\begin{aligned}\forall m_i\in M.\ N_c &= \sum_{c=0}^{n} E_c(v_i,v_j)\\ N_o &= \sum_{o=0}^{m} E_o(v_i,v_j)\end{aligned}\right\} \Rightarrow min(N_c,N_o)$$

5.2 Communication topology synthesis and optimization algorithm

After the ILP formulation for on-chip communication behavior of modules, the minimum overlap and containment of CLTIs are obtained for a session with a certain value of bus width k. We apply those minimum overlap and containment of CLTIs to the well known problem of graph partitioning called clique partitioning to synthesize the communication bus topology. Let $G=(V,E)$ denote a graph, where V is the set of vertices and E the set of edges. Each edge $e_{i,j}\in E$ links two different vertices v_i and $v_j\in V$. A subgraph SG of G is defined as (SV,SE), where $SV\subseteq V$ and $SE=\{e_{i,j}|e_{i,j}\in E, v_i,v_j\in SV\}$. A graph is complete if and only if for every pair of its vertices there exists an edge linking them. A clique of G is a complete subgraph of G. The problem of partitioning a graph into a minimal number of cliques such that each node belongs to exactly one clique is called clique partitioning.

```
/* create a super graph $G'(S,E')$ */
1  $S=\emptyset$;
2  $E'=\emptyset$;
3  **for** each $v_i\in V$ **do** $s_i=\{v_i\}$; $S=S\cup\{s_i\}$; **endfor**
4  **for** each $e_{i,j}\in E$ **do** $E'=E'\cup\{e'_{i,j}\}$; **endfor**
5  **while** $E'\neq\emptyset$ **do**
        /* find $S_{Num1}$, $S_{Num2}$ having most common node */
6       $MostCommons=-1$;
7       **for** each $e'_{i,j}\in E'$ **do**
8           $c_{i,j}=|COMMON\_NODE(G',s_i,s_j)|$;
9           **if** $c_{i,j}> MostCommons$ **then**
10              $MostCommons=c_{i,j}$;
11              $Num1=i$; $Num2=j$;
12          **endif**
13      **endfor**
14  $CommonSet=COMMON\_NODE(G',s_{Num1},s_{Num2})$;
15  $E'=EDGE\_REMOVE(E',S_{Num1})$;
16  $E'=EDGE\_REMOVE(E',S_{Num2})$;
```

```
    /* merge $S_{Num1}$ and $S_{Num2}$ into $S_{Num1Num2}$ */
17  $S_{Num1Num2} = S_{Num1} \cup S_{Num2}$;
18  $S = S - S_{Num1} - S_{Num2}$;
19  $S = S \cup \{S_{Num1Num2}\}$;
    /* add edge from $S_{Num1Num2}$ to super nodes*/
20  for each $s_i \in CommonSet$ do
21      $E^{'} = E^{'} \cup \{e^{'}_{i,Num1Num2}\}$;
22  endfor
23  endwhile
```

Figure 3: Algorithm clique partitioning of modules

Algorithm described in Fig.3 is a heuristic, which is based on the algorithm proposed in [8] to solve the clique-partitioning problem. Initially, each vertex $v_i \in V$ of G is moved in a separate super-node $s_i \in S$ of $G^{'}$ in steps 3-4. At each step, the algorithm finds the super-node of the graph, where each super node consists of all the nodes in connected nodes s_{Num1} and s_{Num2} with maximum number of common nodes. These two super-nodes are then merged into a single super-node, $s_{Num1Num2}$, which consists of all the vertices in s_{Num1} and s_{Num2}. The variable *CommonSet* consists of all the common nodes of s_{Num1} and s_{Num2}. All edges originating from s_{Num1} or s_{Num2} in $G\prime$ are deleted. New edges are added from $s_{Num1Num2}$ to all the super-nodes in *CommonSet*. Above steps are repeated until there are no edges left in the graph. As an end result of this algorithm, we obtain a set of super-nodes with no edge, where each super-node $s_i \in S$ forms a communicating bus, which can be shared by a set of modules m_i inside of it.

Since, the clique partitioning heuristic does not provide the best result, we optimize the communication topology by moving or swapping module(s) from one bus to other bus so that communication overhead delay and power consumption will be minimized by using the bridge seldom. The criteria for swapping modules between the buses is the communication cost which is the function of intermodule communication statistics of the modules (P_{comm}, δ *and* S) obtained by profiling a system at the behavioral level. Where P_{comm} is the average number of communication, δ is the transition density and S is the spatial correlation of communication behavior. The total cost due to the communication behavior of a module is evaluated as,

$$C(m_i) = K \cdot P_{comm}(m_i) + C_\delta \cdot \delta(m_i) + C_s \cdot S(m_i) \qquad (8)$$

where,
$K = 1$
C_δ = bus uses cost for the frequent bus request
C_s = bus uses cost for the contiguous transfer of data

A module is moved to the other bus only when the cost is improved and its CLTIs do not overlap with the rest. it is obvious that for a fixed amount of data to be transferred by a module m_i with higher transition density and lower spatial correlation (contiguous transfer of data) and a module m_j with lower transition density and higher spatial correlation, the total communication cost for m_i will be greater than the cost of m_j. In our case we assume that C_δ is at least two times the cost of C_s, which is $C_s = 1$.

6. EXPERIMENTS AND RESULTS

For an experimental purpose we chose the Talking Assistant [3] (TA), which is hand and eye free mobile device used

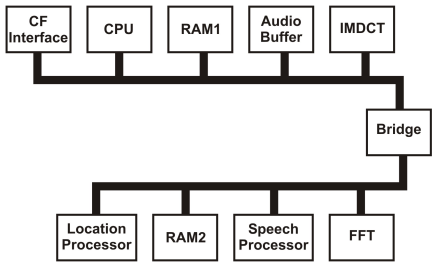

Figure 4: The synthesized communication architecture of Talking Assistant (TA)

for an ubiquitous computing applications. It has several embedded applications like Ogg Vorbis decoder for an audio decompression [2], voice communication, speech recognition unit [1] and IR and Compass based indoor location positioning to provide the context informations. Since our approach consider a system that has been already partitioned and mapped onto the hardware and software part of a system on-chip components, the overall on-chip communication modules are IMDCT (Inverse Modified Discrete Cosine Transformation), CF (Compact Flash)-interface for WLAN interfacing, a general purpose processor, audio buffer to I/O, speech processor, location processor, FFT and two data memories. These on-chip modules communicate with each other as a master and slave, where IMDCT, CF-interface, audio buffer, CPU (central processing unit), speech processor, FFT and location processor act as masters; and memories act as slaves in our example.

We profiled an architecture independent SystemC model of the Ogg Vorbis audio data with a length of 38 s and found that all the on-chip modules are called 8700704 times, periodically. The real time constraint for a session is evaluated as $T_{session} = 4.36$ μs for an audio application, which we keep as the overall constraint. At the start of each session the speech processor loads maximum of 41 frames of speech data and after this the location processor also loads 11 frames of location data from the shared memory. The frame size in these cases is 32 bit wide. After the speech processing, it will initiate the transfer of audio data from CF card to shared memory via the CF-interface. In each session the CF-interface transfers maximum of 64 bytes of data from CF card to memory. There can be less than 64 bytes data transfer from the CF card but we consider the worst case to meet the real time constraint. Once the audio data is available in the shared memory, CPU loads 64 bytes of data and performs first two steps of decompression like inverse quantization and channel decoupling. After this the IMDCT loads 48 bytes of data from the shared memory, which recovers data from frequency domain to time domain and stores its result of 16 bytes data back to the shared memory. The CPU loads that 16 bytes of data and does the fourth step of Ogg Vorbis decoding called reconstruct curve and stores the result back to the shared memory. After the completion of audio decompression, on-chip audio buffer loads 2 bytes of data at each time.

Based on the method described in section 5.1 for ILP formulation, we evaluated CLTIs of each transaction and scheduled them using ASAP with hardware constraint maximum of 3 buses. To find the minimum overlap among CLTIs, we applied ILP for different size of bus width ranging from 16

BusWidth(k)	$(\sum CLTI + W)_{ses.}$	N_o	N_c	$O.T\mu$s
16	9.04 μs	19	17	76.75
20	7.98 μs	21	16	62.63
24	7.26 μs	13	14	52.53
28	6.72 μs	13	12	44.40
32	6.13 μs	12	8	37.91
36	5.74 μs	10	6	32.72
40	5.14 μs	6	6	28.84
44	4.67 μs	4	3	25.85
48	**4.34 μs**	**4**	**3**	**23.32**
52	4.09 μs	3	5	21.16
56	4.07 μs	4	4	19.28
60	4.03 μs	7	2	17.64
64	4.03 μs	7	2	16.45

Table 1: Number of overlaps among the modules with different size of bus width

$Module$	P_{comm}	δ	S	$Cost$	$B1$	$B2$	$Swap$
IMDCT	0.63	0.21	0.67	0.88	$\surd$	–	–
A. buf.	0.24	0.24	0.13	0.85	$\surd$	–	$\surd$
G. proc.	0.48	0.47	0.43	1.85	$\surd$	–	–
S. proc.	0.83	0.86	0.27	1.1	–	$\surd$	–
FFT	0.87	0.72	0.53	2.84	–	$\surd$	$\surd$
CF-inte.	0.57	0.24	0.72	1.77	$\surd$	–	–
L. proc.	0.84	0.66	0.37	2.53	–	$\surd$	–
RAM1	–	–	–	–	$\surd$	–	–
RAM2	–	–	–	–	–	$\surd$	–

Table 2: The intermodule communication statistics of modules and their communication cost

bit to 64 bit. An early start time to transfer data by a successor module after receiving data from predecessor module is defined by the hardware resources of the successor. For example the IMDCT starts transferring data no earlier than 46 cycles after the receiving data from the shared memory. This 46 cycles is the amount of time needed to compute IMDCT of the loaded data from shared memory, which is 0.92 μs for 50 MHz of on-chip operating frequency.

In Table 1, it can be seen that changing the size of bus width number of overlap N_o and containment N_c of CLTIs also changes. These change in number N_o and N_c have the non-linear behavior with the size of bus width. This non-linearity is due to the time constraint W_{m_i,m_j} (minimum processing time of a processor) between predecessor(s) and successor(s) as shown in Fig.1. In addition to this, the column $O.T$ is the overlap time duration, which is in this case decreasing with increasing size of bus width. From the result of Table I, we choose the one which fulfills real time constraints and minimum number of N_o and N_c, which is 48 bit bus width. After this we applied the clique partitioning algorithm to find the communication topology.

In Table 2, we evaluated the total communication cost of each on-chip module, where the FFT has the maximum cost and the audio buffer has the minimum cost, in addition to this their CLTIs do not overlap with each other. Hence, we swap them and optimize the communication architecture by using the bridge seldom. The synthesized communication architecture is shown in Fig.4, where the CF-interface, CPU, IMDCT, audio buffer to I/O, and shared memory (RAM1) are assigned to the $Bus1$, similarly, the speech processor, location processor, FFT, memory (RAM2) are assigned to the $Bus2$. In between $Bus1$ and $Bus2$, there is a bridge to establish communication among modules of $Bus1$ and $Bus2$.

7. CONCLUSION

In this paper, we described a method of custom communication architecture synthesis and optimization techniques for a multi-bus based shared memory architecture. We modeled the communication behavior of on-chip modules as a set of communication lifetime intervals (CLTIs), which is the duration of time a module uses the bus to transfer a certain amount of data. We scheduled all the CLTIs using ASAP with hardware constraint maximum of 3 buses for a session and evaluated new overlap among them for different possible value of bus width. After obtaining the optimized number of overlaps among the CLTIs using ILP method, we synthesized the bus topology using the clique partitioning heuristic. This synthesized communication architecture is later optimized using the intermodule communication statistics. The results shown in this paper demonstrate that the selection of proper size of bus width influences the communication topology of a complex system.

8. ACKNOWLEDGEMENT

The authors would like to acknowledge and thank Dr. Leandro S. Indrusiak for his valuable suggestions.

9. REFERENCES

[1] The cmu sphinx group open source speech recognition engines. http://www.speech.cs.cmu.edu/sphinx/.
[2] Vorbis i specification. http://www.xiph.org/ogg/vorbis/doc/Vorbis_I_spec.html.
[3] E. Aitenbichler, J. Kangasharju, and M. Mühlhäuser.
[4] M. Gasteier and M. Glesner. Bus-based communication synthesis on system level. *In ACM Trans. design automation electron. syst.*, Vol. 4(No. 1):1–11, 1999.
[5] K. Lahiri, A. Raghunathan, and S. Dey. Design space exploration for optimizing on-chip communication architecture. *In IEEE Transactions on Computer Aided Design of Integrating Circuits and Systems*, Vol. 23(No. 6):952–961, June 2004.
[6] R. B. Ortega and G. Borriello. Communication synthesis for distributed embedded systems. In *In proc. Int. Conf Computer Aided Design*, 1998.
[7] A. Pinto, L. P. Carloni, and A. V. Sangiovanni. Constraint driven communication synthesis. In *In proc. of Design Automation Conference*, June 2002.
[8] K. K. Rye and V. MooneyIII. Automated bus generation for multiprocessor soc design. *In IEEE Transactions on Computer Aided Design of Integrating Circuits and Systems*, Vol. 23(No. 11):1531–1549, Nov. 2004.
[9] C. J. Tseng and D. P. Siewiorek. Automated synthesis of data paths on digital systems. *In IEEE Tran. on Computer Aided Design of Integrated Circuits and Systems*, Vol. CAD-5(No. 3):397–395, 1986.
[10] T. Y. Yen and W. Wolf. Communication synthesis for distributed embedded systems. In *In proc. Int. Conf. Computer Aided Design*, pages 288–294, 1995.