A Real Time Budgeting Method for Module-Level-Pipelined Bus Based System using Bus Scenarios [*]

Tadaaki Tanimoto[†‡], Seiji Yamaguchi[‡§], Akio Nakata[‡], and Teruo Higashino[‡]
[†] Renesas Technology Corp., Kodaira, Tokyo 187–8588, Japan
[‡] Graduate School of Information Science and Technology,
Osaka University, Suita, Osaka 565–0871, Japan
tanimoto.tadaaki@renesas.com
{tanimoto, s-yamagt, nakata, higashino}@ist.osaka-u.ac.jp

ABSTRACT

In designing bus based systems with parallel and pipelined architecture, it is important to derive a real time budget (a specified execution time limit) for each task of a bus based system while satisfying given end-to-end real-time constraints of the entire system such as throughput and latency constraints. In this paper, we define a bus scenario representing a set of possible execution sequences of tasks and bus transfers executed in a bus based system. Then we propose a method for deriving real time budgets of all the tasks running in parallel and pipelined fashion from the pair of a system configuration (such as bus topology) and a bus scenario. In deriving such real time budgets, we consider computational complexity of each task, the amount of bus transfers and bus arbitration policies (e.g. fixed priority or time divided round robin based arbitration). We show that the proposed method is effective for designing several bus based systems such as MPEG decoders.

Categories and Subject Descriptors: B.8.2 [Hardware] Performance Analysis and Design Aids

General Terms: Design, Performance

Keywords: Bus based systems, Real-time systems, Pipelined processing, Multimedia processing, Cycle budgeting

1. INTRODUCTION

High performance parallel and pipelined architecture has been used in implementation of several types of image and sound processing with tight real time constraints in System-on-a-Chip (SoC) design [1, 2, 3, 11]. In a system with pipelined architecture, each task periodically receives input data from its predecessor tasks, performs its computation and sends output data to its successor tasks. The tasks that have no predecessors read the external inputs, and those that have no successors generate the external outputs. In designing such a system, bus based communication architecture is very popular to handle on-chip communications because it is simple to design and take up small areas. Here, we call such architectural systems as *module-level-pipelined bus based systems*.

In general, unified single memory architecture may make the performance of the entire system worse due to higher congestion in the memory access [2, 3]. Therefore, in order to design a highly efficient module-level-pipelined bus based system satisfying given real-time constraints, we must consider execution time of each task executed on each module composing the bus based system and bus data transfer time between those modules, where each module may execute multiple tasks (by mode switching, etc.) but at most one task at a time. Designers estimate several candidates of real time budgets very often from their experience, or by creating simulation models annotated with the detailed information based on experience. However, such an effort takes much time. As far as we know, there is no research result on the automatic derivation of a real time budget for each task of a module-level-pipelined bus based system from the top level real-time constraints.

As researches concerning with real time budgeting, [4, 5] have proposed an algorithm to derive a deadline and a period for each periodic software task on a single CPU where communication among tasks is executed asynchronously via buffers and is scheduled in a preemptive fashion. The derived deadline and period satisfy a real time constraint given as a deadline. Since the targets of [4, 5] are concurrent systems with asynchronous communication between periodic tasks through buffers and they restrict the communication model such that only one task can write to each communication buffer, we cannot apply their methods directly to bus based systems with bus arbitration.

In this paper, we propose a real time budgeting method for each task of a given module-level-pipelined bus based system where the input arrival rate is fixed and throughput and latency constraints are given. Our target concurrent system is a module-level-pipelined bus based system composed of several modules communicating via buses, where as in the image and/or sound processing system, the target bus based system satisfies the following conditions: 1) the input arrival rate to the bus based system is fixed, 2) each input data goes through the sequence of data manipulation by modules, memories, and data transfers among modules and memories in the pipelined fashion to the outputs (thus, several data can be concurrently processed by the multiple modules), 3) a data transfer between modules is always performed after completion of a module, 4) each bus based or direct module communication always requires data transfer between modules, and 5) in every direct module communication, the sender blocks when its receiver is busy.

We employ a directed graph representation (called a *bus scenario*) to express possible execution sequences of a module-level-pipelined bus based system with a real time constraint. The bus scenario consists of nodes representing tasks, arcs representing data transfer between tasks, and tokens on the nodes and/or arcs representing data. All nodes/arcs have information on time passages representing task execution time/data transfer time, respectively. The tokens on the nodes and/or the arcs indicate which tasks are in execution and/or data transfer. There may exist multiple tokens simultaneously indicating concurrent execution caused by pipelined processing.

Our goal is to derive a set of execution time limits (called *(real) time budgets*) for all tasks and bus transfers that satisfies the given latency and throughput constraints in the presence of bus/resource conflicts. The proposed real time budgeting method consists of two major phases: one is a *rough estimation phase* and the other is a *constraint validation phase*. First we *roughly estimate* a time budget for each bus transfer and task, and then we *validate* whether the estimated set of time budgets satisfy the given end-to-end real-time

[*]This work is partially supported by Renesas Technology, Corp.
[§]Presently with Canon, Inc.

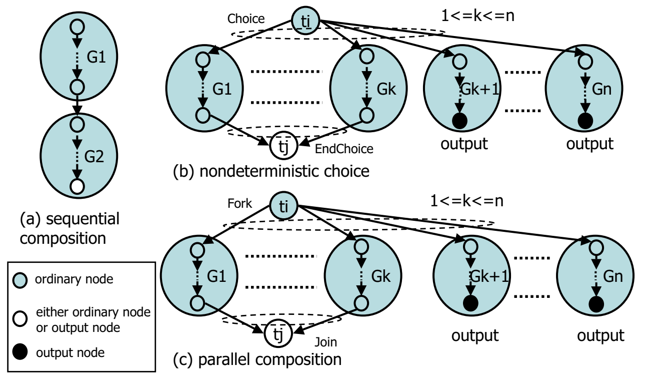


Figure 1: Inductive Definition of Bus Scenarios

constraints. If it does not, we consider that we must have *overestimated* time budgets. Thus, we reduce each task time budget based on certain heuristics and validate again until the constraints are satisfied. Here, we do not reduce bus time budgets and simply use the estimated worst case bus transfer time. Using the above technique, we obtain a *correct* but *possibly underestimated* set of time budgets for all tasks/bus transfers. Then, we perform a *binary search* between the least overestimated and the most underestimated time budgets so that we can obtain a more relaxed and correct set of time budgets.

In this paper, due to the space limitation, we only present a case study for a MPEG decoder bus based system with an unified single memory [9] and show the derived real time budget for each task.

2. MODELING MODULE-LEVEL-PIPELINED BUS BASED SYSTEMS

We explain how to model a module-level-pipelined bus based system with a single input stream and multiple output streams.

First, we define a **bus based system configuration** (BC for short) which represents a communication topology of modules of a bus based system.

Definition 1. *A* bus based system configuration (BC) *is an undirected connected graph $G_{BC} = G_{BC}(V, E)$, where each node is either a* module node *or a* bus node*, no two bus nodes are directly connected, and there are at most one path between any two module nodes. A module [bus] node is annotated with the module name [bus name, resp.].* □

If two module nodes in a BC are directly connected, we consider that there is a direct communication channel between the corresponding two modules. Otherwise, we consider that two modules communicate via a directly connected bus node between them. We write a bus node as a long horizontal/vertical line in the following figures. We omit a bus name if it is apparent from the context.

Next, we define a **bus scenario** (BS for short) which represents a set of possible timed execution sequences of tasks executed on modules in the bus based system for each input of the entire bus based system.

Definition 2. *A* bus scenario (BS) $G = G(T, D)$ *is a directed acyclic graph with an unique root (rooted-DAG, for short) which is inductively defined as follows:*

1. *A rooted-DAG containing only one node and no arcs is a BS.*
2. *If $G_1, \ldots, G_n$ are BSs, then the constructed rooted-DAGs as in Fig. 1 are also BSs.*

A node in a BS which has no outgoing arcs is called an output node*. Otherwise, it is called an* ordinary node*. The unique root node of a BS is called the* input node*. Each node of a BS represents a task executed by a module in a bus based system, denoted by $t_i \in T$. Each task t_i is annotated with a module name on which it is executed and a fixed execution time of the task.*[1] *The task on the input node takes external inputs of the entire bus based system. The tasks on the output nodes emit outputs of the entire bus based system to the external environment. Each directed arc $d_{ij} \in D$ between tasks t_i and t_j represents data dependencies. Each d_{ij} is annotated with a data transfer time, a label indicating either bus based or direct module communication. Moreover, each t_i has two attributes Incoming(t_i) and Outgoing(t_i) defined as follows:*

- Choice *and* Fork
  If Outgoing(t_i) = Choice [Outgoing(t_i) = Fork], t_i invokes one of the successor tasks nondeterministically [all of the successor tasks in parallel, resp.] when it is finished.
- EndChoice *and* Join
  If Incoming(t_i) = EndChoice [Incoming(t_i) = Join], t_i starts its execution when one of the predecessor tasks is finished [all of the predecessor tasks are finished, resp.]. □

[1] Each t_i can be a dummy node (a task whose execution time is zero) to describe a control flow dependency.

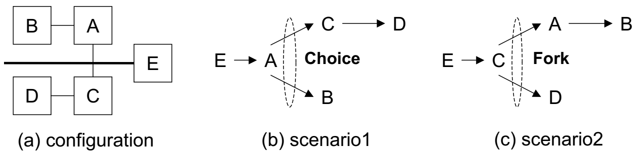


Figure 2: Bus Based System Configuration and Bus Scenarios of Example 1

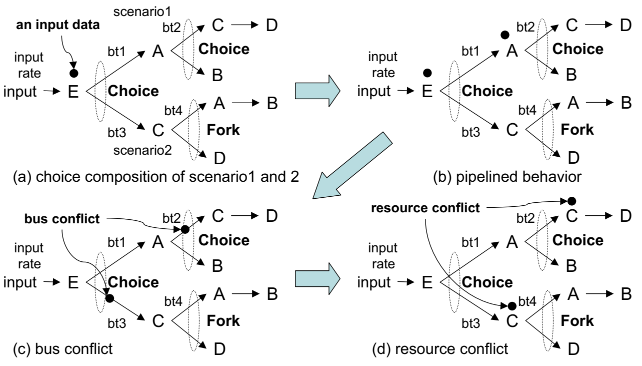


Figure 3: Scenario of Example 2 and its Pipelined Behavior with Bus/Resource Conflicts

We specify a behavior of a bus based system by a pair of BC and BS, as shown in the following example.

Example 1. *In the bus based system described in Fig. 2(a), only modules A, C, and E are connected to the bus, module A is connected only to B, and module D is connected only to C. For this bus based system, we may define the following BSs, Scenario1 and 2 (see Fig. 2(b),(c)). In Scenario1, A fetches data from E, performs some data manipulation, and then transfers data to* either *B or C. If C gets data from A, C manipulates data and then transfers them to D. On the other hand, in Scenario2, C fetches data from E, performs some data manipulation, and then transfers data to* both *A and D. Then A transfers data to B on getting data from C.* □

Example 2. *Fig. 3-(a) describes choice of the two BSs in Example 1 in a BS. The labeled arcs bt1,...,bt4 represent bus transfers.* □

In Fig. 3-(a), several tasks can be executed simultaneously due to the nature of a pipelined behavior (Fig. 3-(b)). But since a bus is shared among multiple modules, no two bus data transfers can occur at the same time (called a *bus conflict*, illustrated in Fig. 3-(c)). Moreover, since the bus based system has only one module C, two tasks in different paths cannot be executed by the module C at the same time (called a *resource conflict*, illustrated in Fig. 3-(d)). The same is true for A, B and D. In general, multiple tasks in a BS can be executed on a single module only once at a time.

It is therefore conceivable to employ a Time Petri Net (TPN) to define the execution semantics of a BS. A *TPN* is a bipartite directed graph consists of circles (called *places*), rectangles (called *transitions*), and directed arc between places and transitions, where each transition has a minimum and maximum time constraint (δ_{min}

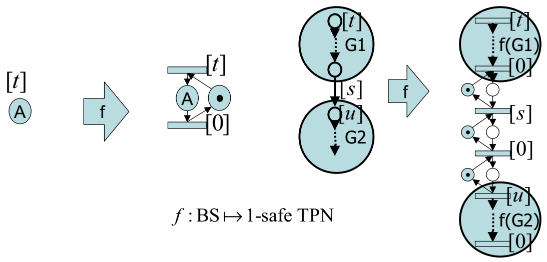


Figure 4: Transformation from BS to 1-Safe TPN (Root Node and Sequential Composition)

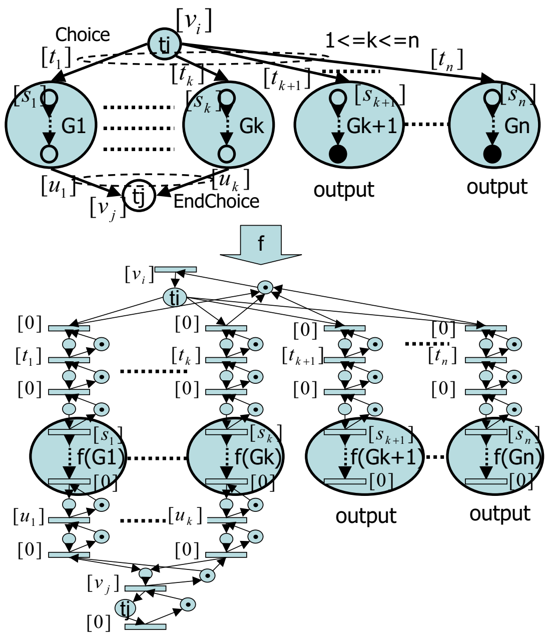


Figure 5: Transformation from BS to 1-Safe TPN (Nondeterministic Choice)

and δ_{max}, respectively), each arc has a *weight*, and each place can hold *tokens*. Each transition t is *enabled* when every predecessor place has tokens more than or equal to the weight of its outgoing arc to t. An enabled transition t may be *fired* after some time δ satisfying $\delta_{min} \leq \delta \leq \delta_{max}$ has elapsed since it was last enabled, and still remains enabled. It must be fired at the last time it can be fired unless it is disabled. If a transition t is fired, some tokens in all the predecessor places of t are removed, and some tokens are added to every successor place of t. The number of removed/added tokens for each place is equal to the weight of the arc connecting the place and the transition t. A TPN is called *live* if and only if any transition of the TPN is firable infinitely often. A TPN is called *bounded* (or *safe*) if and only if the number of tokens in any place is bounded to some fixed constant.[2]

We define the execution semantics of a BS as follows.

Definition 3. *The execution semantics of a BS is defined as that of Time Petri Net (TPN) constructed from the BS by the following two successive transformations.*

Transformation 1. *(Transformation from a BS to a 1-safe TPN)*
Each node in a BS is translated into a 1-safe TPN by the transformation function f in Figs. 4, 5, and 6. where circles and rectangles represent places and transitions in TPN, respectively, $[t]$ on a transition indicates that both the minimum and the maximum time passages of the transition are the non-negative real number t.

Transformation 2. *(Transformation of subnets for resolving resource conflicts)*
When different places are labeled with the same module name, we transform the subnet as in Fig. 7 to resolve a resource conflict of the place. □

[2]Due to the lack of space, we do not recall the formal definition of the syntax and operational semantics of TPN. For detail, refer to [12].

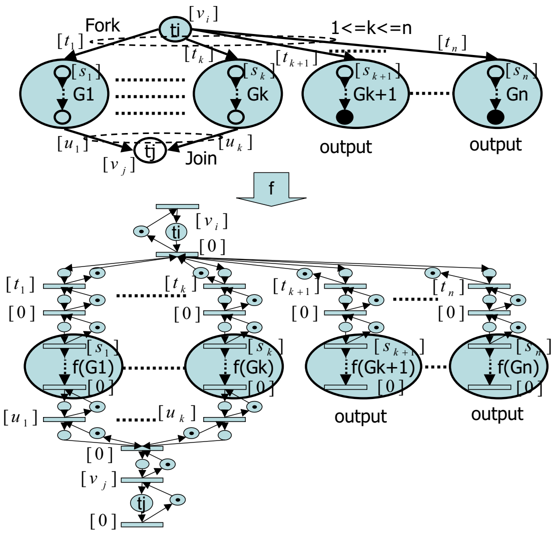


Figure 6: Transformation from BS to 1-Safe TPN (Parallel Composition)

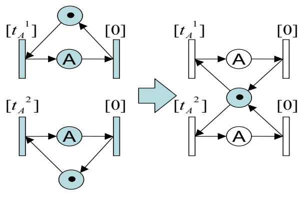


Figure 7: Transformation of Subnets for Resolving Resource Conflicts

Note that after applied Transformation 1. in Definition 3, according to its structure, the obtained 1-safe TPN is ensured to live. However, after Transformation 2. in Definition 3, the obtained TPN is not ensured to live, depending on each real time budget to be assigned to each task (transition). Therefore, we have to explore the appropriate value of real time budget for each task to ensure the liveness and given latency/throughput constraints.

3. REAL TIME BUDGETING PROBLEM

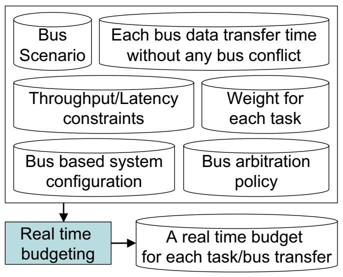
In this section, we define the real time budgeting problem as follows.

Definition 4. *The* real time budgeting problem *to a bus based system is to derive a real time budget on each task satisfying a given top-level latency/throughput constraint on the bus based system from the following inputs: 1) a BC describing the target architecture of the bus based system, 2) a BS describing the set of timed task execution sequences of the bus based system, 3) a bus data transfer time for each module under the condition there is no bus conflict, 4) a weight for each task, 5) a latency/throughput constraint, and 6) a bus arbitration policy(either a fixed priority or a time divided round robin based) (illustrated in Fig. 8).* □

In our method, we try to derive as much real time budgets as possible.

Example 3. *The bus based system configuration of a MPEG decoder bus based system[9] is depicted in Fig. 9.*

An input packet first goes to the coded video bit-stream interface(CVIF), is stored to an external memory(extDRAM) once, and then is read by the variable length decoder(VLD). After VLD processing, a conditional branching occurs according to the sort of the input frame. As for I frame, the output image is constructed



Bus Scenario
Each bus data transfer time without any bus conflict
Throughput/Latency constraints
Weight for each task
Bus based system configuration
Bus arbitration policy
Real time budgeting
A real time budget for each task/bus transfer

Figure 8: Real Time Budgeting Problem

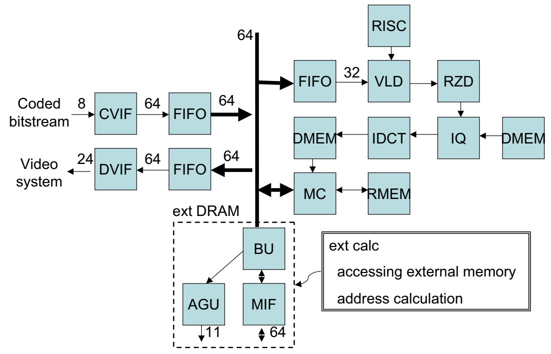


Figure 9: Bus Based System Configuration of MPEG Decoder LSI

by simply decoding the input packet. For P and B frames, the output images are constructed using the difference from the previous frames (case of P frame) or both the previous and next frames (case of B frame). In the I frame decoding process, the MPEG decoder performs variable length decoding(VLD), inverse quantization(IQ), inverse discrete cosine transformation(IDCT), motion compensation(MC), and then decodes the output image. In P or B frame decode processing, parameters including a motion vector are extracted from an input frame(VLD,RISC), a reference image is fetched(RMEM), motion compensator(MC) is executed, and then an image is decoded. The decoded image is stored to an external memory(extDRAM), transfered to a buffer(DispFIFO), sent to a format transformation(DVIF), and then is finally outputted.

We show the corresponding BS in Fig. 10, where t_k^i indicates a time budget for the k-th task on Path(i) and we omit local storages RMEM, DMEM, and DispFifo in the BS since their execution time can be characterized as 0 by regarding accesses to them as a part of execution of other tasks.

The arcs labeled with $bt1, \ldots, bt8$ represent bus data transfers. In this example, a fixed priority based bus arbitration is given as $bt0 < bt1 < \cdots < bt8$. Moreover, each bus data transfer time without any bus conflicts is given as in Table 1-(a). Note that each bus data transfer time without any bus conflicts can be obtained by dividing the given total amount of transferred data by the given bus transfer rate.

As for the weight of each node, we set 0 to nodes indicating bus data transfers, and roughly estimate a MOPS(Million Operation Per Seconds)/ (the number of available arithmetic components in the implementation) of each module.[3] *According to [11], we assume that the performance of MC and DCT is 1 pixel/cycle, respectively. Then we determine the number of available arithmetic components in the implementations of MC and DCT are 128 and 12, respectively. The weight for each task is depicted in Table. 2.*

We set the latency/throughput constraints on the pipelined MPEG decoder bus based system to be 4/30[sec.] and 1/30[sec.], respectively, where we assume the number of stages of the pipelined bus based system is 4 (the same as [11]).

In this case, the real time budgeting problem is to derive the set of real time budgets t_k^i for all tasks satisfying the given latency/throughput constraints, which is shown in Table 5. □

[3] In the calculation of the MOPS, we referred to [10].

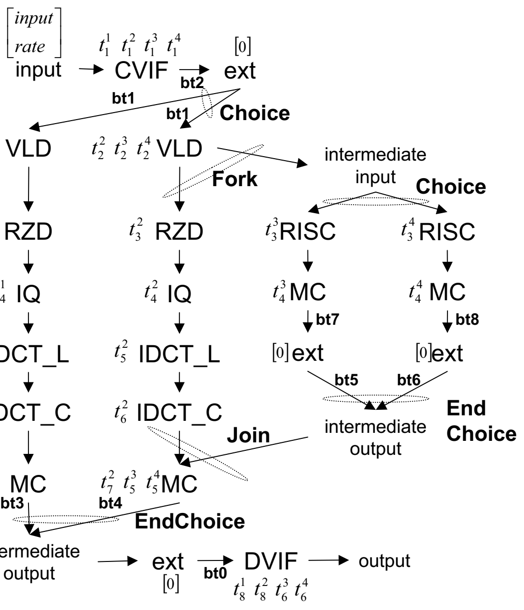


Figure 10: Bus Scenario of MPEG Decoder

Table 1: Worst Case Execution Time of Each Bus Data Transfer

Bus data transfer transition (in priority order)	(a) Transfer time[msec.]	(b) WCET[msec.]
bt0	1.08	1.08
bt1	1.08	2.16
bt2	1.08	3.24
bt3	1.08	4.32
bt4	1.08	5.4
bt5	1.08	6.48
bt6	2.16	8.64
bt7	0.030375	8.670375
bt8	0.06075	8.731125

4. REAL TIME BUDGETING METHOD

In this section, we describe the method to derive a set of real time budgets for all bus transfers and tasks while considering bus and resource conflicts. In the sequel, we denote each directed path from the input node to an output node on a BS by *Path*(i) (i is an index for each path). We also denote the set of all tasks on *Path*(i) by $T(Path(i))$.

4.1 Estimating Bus Time Budgets

In our proposed method, we will calculate the worst case execution time (WCET for short) for each bus data transfer with a fixed priority based or a time divided round robin based bus arbitration provided that each bus is mostly congested. Then, we assign the obtained WCET to each bus transfer time budget. We assume that if a bus transfer does not consume all the assigned bus time budget, the module that receives the bus transfer must wait until the rest of the time budget is consumed.

1. We assume that a BS has n bus data transfers $\{bt_1, bt_2, \ldots, bt_n\}$ and their data transfer time without any bus conflict are $\{x_1, x_2, \ldots, x_n\}$.

(a) Fixed priority based arbitration

In this case, a bus arbiter grants a bus access to the bus request module that has the highest priority among all the pending bus requests. Here we assume that a priority value $\sigma(i)$ has a higher priority than another $\sigma(j)$ when $\sigma(i) < \sigma(j)$. Thus, if the priority of a bus data transfer bt_i is $\sigma(i)$ $(1 \leq \sigma(i) \leq n)$, WCET x'_i of bus data transfer at bt_i is given as $x_i + \sum_{\sigma(j)<\sigma(i)} x_j$, where $\sigma : \{1, \ldots, n\} \mapsto \{1, \ldots, n\}$ is a permutation.

(b) Time divided round robin based arbitration

Table 2: Weights for Each Task Operation

k		$Path(1)$	$Path(2)$	$Path(3)$	$Path(4)$
1	CVIF	1.944	1.944	1.944	1.944
2	VLD	1.2	1.2	1.2	1.2
3	RZD	2.4	2.4		
4	IQ	4	4		
5	IDCT_L	7.7916	7.7916		
6	IDCT_C	7.7916	7.7916		
7	MC	0.030375	1.09375		
8	DVIF	10	10		
3	RISC			1.2	2.4
4	MC			0.030375	0.06075
5	MC			1.09375	1.09375
6	DVIF			10	10
Total		35.157575	36.220950	15.468125	16.698500

In this case, each module can access to each bus within a given time slice t_{slice} by turn. Therefore, if different n modules access the same bus at the same time, WCET x'_i of the bus data transfer bt_i is $\lceil x_i/t_{slice}\rceil \times t_{slice} \times n$.
We set the time constraint x'_i to bt_i. We perform this procedure to all bus data transfers in a BS.

2. For each path in a BS, we sum up WCETs of all arcs representing the bus data transfers, which are estimated by 1., and subtract the sum from the given latency constraint to obtain the total task time budget $Lw_i(0)$ for tasks on each path $Path(i)$.

Example 4. *The estimated WCETs for each bus data transfer of Example 3 is shown in Table 1-(b).* □

4.2 Estimating Task Time Budgets

In this section, we describe how to estimate a time budget for each task while considering resource conflicts. In the sequel, we use a vector of the total task time budgets $\overrightarrow{Lw(0)} = (Lw_1(0), \ldots, Lw_n(0))$ calculated in the previous subsection for each path on a BS.

Step1 (Estimating task time budget under the assumption that no resource conflicts happen)
For each path $Path(i)$ ($i \in \{1, \ldots, n\}$) of the BS, we calculate a weighted average of the total task time budget $Lw_i(0)$ w.r.t a weight specified for each task on $Path(i)$ and then set it as an estimated time budget for the corresponding task. For any task $t \in T(Path(i))$, a time budget $Val_i(Lw_i(0), t)$ for t is given as $Val_i(Lw_i(0), t) \overset{\text{def}}{=} Lw_i(0) \times Weight(t)/WS(i)$, where $Weight(t)$ represents the weight specified to the task $t \in T$ and $WS(i)$ is the sum of the weights specified to the tasks on the path $Path(i)$, i.e. $WS(i) \overset{\text{def}}{=} \sum_{t' \in T(Path(i))} Weight(t')$. A time budget $Val(\overrightarrow{Lw(0)}, t)$ for a task t is given as $Val(\overrightarrow{Lw(0)}, t) \overset{\text{def}}{=} \min_{i \in \{1,\ldots,n\}}\{Val_i(Lw_i(0), t)\}$.

Step2 (Estimating task time budget under the assumption that resource conflicts happen at most j times)
Initially, let $j := 1$. A task t might encounter resource conflict if some other task t' is executed on the same module (that is, t and t' are annotated with the same module name). Under the assumption that every task on each path $Path(i)$ always has at most j resource conflicts, we first calculate the worst case resource conflict resolution time $TConf(i, j)$ at each task on the same module. $TConf(i, j)$ is a time passage to wait for that all tasks on the same module in different paths on a BS acquire and release the module resource j times. In this calculation, we use a time budget for each task estimated at Step1. $TConf(i, j) \overset{\text{def}}{=} \sum_{t \in T(Path(i))} \sum_{k \in \{1,\ldots,n\}\setminus\{i\}} \sum_{t' \in T(Path(k)) \cap ConfTask(t)} (Val_i(Lw_i(j-1), t') \times j)$, where $ConfTask(t)$ is a function returning the set of tasks executed on the same module as t on the other paths, or empty when such tasks do not exist. Then we calculate a new total task time budget $Lw_i(j)$ by subtracting $TConf(i, j)$ from $Lw_i(j-1)$ of path $Path(i)$ and perform Step1 to recalculate a time budget for each task, i.e., $Lw_i(j) \overset{\text{def}}{=} Lw_i(j-1) - TConf(i, j)$.

Step3 (Validation of the estimated task time budgets at Step2)
We annotate the estimated task time budgets $\{Val_i(Lw_i(j), t)\}_{i \in \{1,\ldots,n\}, t \in T(Path(i))}$ to TPN representing the timed execution semantics of a BS. Then, we check whether the annotated TPN satisfies liveness and the given throughput/latency constraints (see Sect. 4.3). If all the properties are satisfied, go to Step4. Otherwise increment j and then go to Step2.

Table 3: $Lw(0)$ and $Lw(1)$ for Each Path

	Lw(0)[msec.]	Lw(1)[msec.]
$Path(1)$	122.533	32.468489
$Path(2)$	121.453	39.960259
$Path(3)$	106.311	78.467468
$Path(4)$	104.128	83.172884

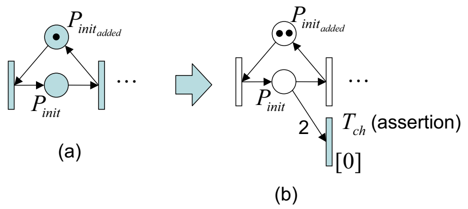


Figure 11: Initial Places of (a) 1-Safe TPN, and (b) 2-Safe TPN with Extra Transition as Throughput Assertion

Step4 (Interpolation between two estimated time budgets)
Intuitively, a better solution compared with the solution derived at Step3 $\{Val(\overrightarrow{Lw_i(j)}, t)\}_{t \in T}$ exists between $\{Val(\overrightarrow{Lw_i(j)}, t)\}_{t \in T}$ and $\{Val(\overrightarrow{Lw_i(j-1)}, t)\}_{t \in T}$.
We interpolate with K points between $Lw_i(j-1)$ and $Lw_i(j)$, where K is an arbitrarily determined number.

Step5 (Solution exploration via binary search)
To find the best solution among the solution candidates constructed at Step4 effectively, we perform the binary search. In the binary search, we iteratively apply Step3 while updating the total task time budgets. In this way, we will find the maximum time budget solution satisfying the given latency and throughput constraints among the interpolated K solution candidates.

Example 5. *For Example 3, the derived values of $\overrightarrow{Lw(j-1)}$ and $\overrightarrow{Lw(j)}$ are shown in Table 3. In this case, the time budget based on $\overrightarrow{Lw(1)}$ satisfies the liveness and given latency/throughput constraints. Thus, we can conclude that $j = 1$ and the best solution may exists between $\overrightarrow{Lw(0)}$ and $\overrightarrow{Lw(1)}$.* □

4.3 Validating End-to-End Constraints For Estimated Time Budgets

In this section, we will explain how to perform liveness analysis and throughput/latency constraint checking for TPN constructed from a given BS with real time budgets calculated at Sect. 4.2.

To check whether the given bus based system with the estimated real time budgets using the method in Sect. 4.2 satisfies liveness and throughput constraints, we annotate the real time budgets and the bus data transfer WCETs to corresponding transitions of TPN A derived from a given BS by Definition 3. Let B be the obtained TPN. Let P_{init} and $P_{init_{added}}$ be the initial place of TPN B and the inserted place to force the place P_{init} be 1-safe, respectively (see Fig. 11(a)). If B does not satisfy the given throughput constraint, and if we remove the restriction forcing P_{init} to be 1-safe, the number of tokens in P_{init} will eventually become more than two. To verify P_{init} is always 1-safe even without any additional subnets forcing it to be 1-safe, we modify B to construct C by changing the number of tokens in place $p_{init_{added}}$ to be two, adding a new transition T_{ch} and an arc with weight 2 from P_{init} to T_{ch} (see Fig. 11(b)). Obviously, C is a bounded (2-safe) TPN [4] due to its structure [8]. If all the transitions except T_{ch} of C are live, TPN B satisfies a given throughput constraint even without any additional subnets forcing P_{init} to be 1-safe. We apply a Time Petri Net analysis tool TINA [7] to check this.

To check whether the given bus based system with the real time budgets satisfies latency constraint, we apply TINA to perform a TPN simulation for C, to obtain transition firing sequences. We analyze the transition firing sequences to calculate the maximum time passage from a source transition, which corresponds to the data input node in the given BS, to sink transitions, which correspond to

[4] In general, the problem to check whether TPN is bounded or not is undecidable[13].

Table 4: Search Result for Real Time Budgets

search order	Interpolated point index	liveness check	throughput const. ($\leq 1/30?$)	latency [nsec.]	latency const. ($\leq 4/30?$)
1	0	NG	NG	421878817	NG
10	1024	NG	NG	390280520	NG
13	1152	NG	NG	137476760	NG
20	1153	OK	OK	132616461	OK
19	1154	OK	OK	132615839	OK
18	1156	OK	OK	132614597	OK
17	1160	OK	OK	132612109	OK
16	1168	OK	OK	132607134	OK
15	1184	OK	OK	132597187	OK
14	1216	OK	OK	132577291	OK
12	1280	OK	OK	132537498	OK
11	1536	OK	OK	132378334	OK
9	2048	OK	OK	132060003	OK
8	4096	OK	OK	130786680	OK
7	8192	OK	OK	128240028	OK
6	16384	OK	OK	123146731	OK
5	32768	OK	OK	112960133	OK
4	65536	OK	OK	92586938	OK
3	131072	OK	OK	57776175	OK
2	262144	OK	OK	43268489	OK

Table 5: Derived Real Time Budget t_k^i for Each Task

k		*Path*(1) [nsec.]	*Path*(2) [nsec.]	*Path*(3) [nsec.]	*Path*(4) [nsec.]
1	CVIF	6479998.129	6479998.129	6479998.129	6479998.129
2	VLD	4157695.707	3999998.845	3999998.845	3999998.845
3	RZD	8315391.414	7999997.691		
4	IQ	13858985.69	13333329.48		
5	IDCT	26995918.23	25971992.5		
6	IDCT	26995918.23	25971992.5		
7	MC	105241.6726	3645832.281		
8	DVIF	33333323.71	33333323.71		
3	RISC			8230496.1	14937640.19
4	MC			208334.4325	378109.0173
5	MC			3645832.281	3645832.281
6	DVIF			33333323.71	33333323.71

the data output nodes in the given BS, and then check the maximum time passage is less than or equal to the given latency constraint.[5]

5. EXPERIMENTAL RESULT

As a case study, we present the result of the proposed method applied to Example 3.

In this experiment, we set the number K of interpolation points for the binary search to 262,144 (2^{18}). The experimental result of real time budget exploration via binary search is shown in Table 4.

For the latency analysis, we simulate 10,000 frames for every analysis using TINA. As the result of binary search, the obtained most relaxed total time budget is 132.616461[msec.]. The obtained most relaxed real time budgets for all tasks satisfying both the latency and throughput constraints are shown in Table 5.

By using the result, if each task is implemented to each module while satisfying the derived real time budget, the entire system is ensured to satisfy the given end-to-end real-time constraints. The budgeting policy is fair since the computational complexity of each task is considered. Moreover, if some task does not consume the assigned budget entirely, we can slow down the task execution by either reducing the clock frequency of the executing module or stopping the clock when the task is finished, to save the power consumption. The same is true for bus transfer time budgets.

6. CONCLUSION

In this paper, we proposed a method to derive a real time budget for each task executed on each module of a given pipelined bus based system with a single input stream with a fixed data input rate and multiple output streams. In our method, a set of module level task execution and bus data transfer sequences of a bus based system is specified by our proposed model, a bus scenario. We present a case study on a MPEG decoder bus based system to show usefulness of our proposed method.

[5]The latency constraint can be formally checked by using a model checker for TPNs that is capable of checking time bounded liveness properties (e.g. "whenever some event A happens, some event B will eventually happen within time t"), but we had no such choice since there were no such model checker handy at the time of our experiment.

[14] proposed a real-time scheduling method to a single TDMA bus based system under the assumption that a fixed execution time for each task and each data transfer on the bus are given. However, resource conflicts due to pipelined behavior is not considered in [14]. On the contrary, our proposed method considers resource conflicts due to pipelined behavior. Therefore, we can relax a real time budget for each task by expanding end-to-end latency constraints while satisfying a given throughput constraint. This yields the lower power dissipation by reducing clock frequency and/or supply voltage.

We presented only MPEG2 decoder example as a case study, but we believe that we can apply our proposed method to other examples like a TCP/IP interface, a USB interface, and a digital baseband logic while ignoring their reactive behaviors. It is true that such an application does not always exhibit a pure pipelined behavior since a reactive behavior for such a system is usually required for error handling. However, if such an error handling sequence dose not have a real-time constraint, our method can be applied to derive a real time budget for each task in normal execution sequences as an initial solution.

The future work is to apply our proposed method to other pipelined bus based systems, to relax bus data transfer WCET by analyzing the actual bus data transfers, and then to relax a real time budget for each task executed in each module by taking advantage of the relaxed bus data transfer WCET.

Another future work is to extend our proposed method by incorporating existing bus access optimization methods[14, 15]. By changing the weight for each task and number of modules which cause resource conflicts, a rough estimation for each task and bus data transfer starting time can be derived on-the-fly to work with bus access optimization methods.

7. REFERENCES

[1] J. M. Fernandez, F. Moreno, and J. M. Meneses, "A High-Performance Architecture with a Macroblock-Level-Pipeline for MPEG-2 Coding", Real-Time Imaging, Vol. 2, Issue 6, pp. 331–340, 1996.

[2] J. H. Li and N. Ling, "Architecture and Bus-Arbitration Schemes for MPEG-2 Video Decoder", IEEE Trans. on Circuits and System for Video technology, Vol. 9, No. 5, pp. 727–736, 1999.

[3] C. W. Yoon, R. Woo, J. Kook, S. J. Lee, and H. J. Yoo, "An 80/20-MHz 160-mW Multimedia Processor Integrated With Embedded DRAM, MPEG-4 Accelerator, and 3-D Rendering Engine for Mobile Applications", IEEE Trans. of Solid-State Circuits, Vol. 36, No. 11, pp. 142–143, 2001.

[4] R. Gerber, H. Seongsoo, and M. Saksena, "Guaranteeing Real-Time Requirements with Resource-Based Calibration of Periodic Processes", IEEE Trans. on Softw. Eng., Vol. 21, No. 7, pp. 575–592, 1995.

[5] I. Shin, I. Lee, and S. L. Min, "Embedded System Design Framework for Minimizing Code Size and Guaranteeing Real-Time Requirements", Proc. of the 23rd IEEE Real-Time System Symp. (RTSS 2002), pp. 201–211, 2002.

[6] T. Murata, "Petri Nets: Properties, Analysis and Applications", Proc. of the IEEE, Vol. 77, No. 4, pp. 541–580, 1989.

[7] B. Berthomieu, P. O. Ribet, and F. Vernadat, "The tool TINA – Construction of Abstract State Spaces for Petri Nets and Time Petri Nets", Int. Journal of Production Research, Vol. 42, No. 14, pp. 2741–2756, 2004.

[8] A. Datta and S. Ghosh, "Synthesis of a Class of Deadlock-Free Petri Nets", Journal of the ACM (JACM), Vol. 31, Issue 3, pp. 486–506, 1984.

[9] T. Demura et. al., "A Single-Chip MPEG2 Video Decoder LSI", Proc. of ISSCC'94, pp. 72–73, 1994.

[10] The Institute of Image Information and Television Engineers, "Selected Writings of Comprehensive Information Technology on MPEG", Ohmsha,1996.(In Japanese).

[11] T. Onoye, T. Masaki, Y. Morimoto, Y. Sato, I. Shirakawa, and K. Matsumura, "Single Chip Implementation of MPEG2 Decoder for HDTV Level Pictures", IEICE Trans. Fundamentals, Vol. E79-A, No. 3, pp. 330–338, 1996.

[12] F. Cassez and O. H. Roux, "Structural Translation of Time Petri Nets into Timed Automata", Proc. of the Workshop on Automated Verification of Critical Systems (AVoCS'04), Electronic Notes in Theoret. Comput. Sci., Vol. 128, pp. 145–160, 2005.

[13] B. Berthomieu and M. Diaz, "Modeling and Verification of Time Dependent Systems Using Time Petri Nets", IEEE Trans. on Softw. Eng., Vol. 17, No. 3, pp. 259–273, 1991.

[14] W. Zheng, J. Chong, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli, "Extensible and scalable time triggered scheduling", Proc. of the 5th Int. Conf. on Application of Concurrency to System Design (ACSD 2005), pp. 132–141, 2005.

[15] P. Pop, "Analysis and synthesis of communication-intensive heterogeneous real-time systems", Ph.D. Thesis No. 833, Dept. of Computer and Information Science, Linkping University, 2003.

[16] T. Tanimoto, "Parametric Analysis, Design Space Exploration, and Scheduling for Real-Time Concurrent Systems with Finite Resources", Ph.D. Thesis, Graduate School of Information Science and Technology, Osaka University, 2006.