Evaluation and Improvements of Runtime Monitoring Methods for Real-Time Event Streams

BIAO HU, Technische Universität München
KAI HUANG, Sun Yat-Sen University and Technische Universität München
GANG CHEN, Technische Universität München
LONG CHENG, Technische Universität München
ALOIS KNOLL, Technische Universität München

Runtime monitoring is of great importance as a safeguard to guarantee the correctness of system runtime behaviors. Two state-of-the-art methods, dynamic counters and l-repetitive function, were recently developed to tackle the runtime monitoring for real-time systems. While both are reported to be efficient in monitoring the arbitrary events, the monitoring performance between them has not yet been evaluated. This article evaluates both methods in depth, to identify their strengths and weaknesses. New methods are proposed to efficiently monitor the many-to-one connections that are abstracted as AND and OR components on multiple inputs. Representative scenarios are used as our case studies to quantitatively demonstrate the evaluations. Both methods are implemented in hardware FPGA. The timing overhead and resource usages of implementing the two methods are evaluated.

Categories and Subject Descriptors: C.3 [**Special-Purpose and Application-Based Systems**]: Real-Time and Embedded Systems

General Terms: Evaluation, Improvements

Additional Key Words and Phrases: Runtime monitoring, event stream model, dynamic counters, l-repetitive function

1. INTRODUCTION

For the class of hard real-time embedded systems, meeting timing constraints, such as the deadline for every task execution, is a fundamental requirement. Therefore, a large amount of research has been devoted to the design-time schedulability analysis at different abstraction levels. The resulting schedule of an analyzed system, however, relies on the assumption that all system events conform to the specifications used by the design-time analysis. But with the increasing complexity of embedded systems, runtime events may not conform to the design-time specifications. For example, in mixed-criticality systems, the system may be overloaded by the low-critical tasks [Neukirchner et al., 2013b; Neukirchner et al., 2013a]. Therefore, runtime monitoring is important to further guarantee that the system timing properties comply with the design-time analysis. Runtime monitoring also helps to improve system performance. System events are often regulated by a designed shaper in order to

This work has been partly funded by China Scholarship Council, German BMBF projects ECU (grant number: 13N11936) and Car2X (grant number: 13N11933), and China SYSU 'the Fundamental Research Funds for the Central Universities' (grant number: 15lgjc32).
Authors' addresses: B. Hu, G. Chen, L. Cheng, and A. Knoll, Technical University Munich, Boltzmannstrae 3, 85748, Garching, Germany; K. Huang(corresponding author), School of Data and Computer Science, Sun Yat-sen University, Xiaoguwei Island, Panyu District, Guangzhou 510006, China. email: huangk36@mail.sys.edu.cn.

DOI:http://dx.doi.org/10.1145/0000000.0000000

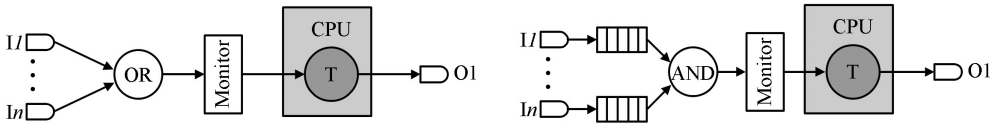
reduce the task response time or the interference on other tasks [Phan and Lee, 2013; Wandeler et al., 2012]. The shaping approach relies on an accurate monitoring on the arriving events.

To design an efficient monitoring scheme, there are two crucial requirements. Firstly, the verification performed by the monitor should be accurate. For the hard real-time system, verification must strictly exclude false positives (erroneous acceptance of the sequence), as this will threaten the safety of the overall system. At the same time, the number of false negatives (erroneous blocking of the sequence) needs to be kept to a minimum, i.e., the blocking of non-violation behavior should be avoided in order to increase the processor utilization as much as possible. This means the pessimistic estimate is often used to strictly rule out all the violations to guarantee the safety of the system, and should be lightweight due to its frequent activations and the tight resource budgets of the system [Lampka et al., 2011; Neukirchner et al., 2012]. It is not easy for a monitoring scheme to meet both aforementioned requirements as ensuring better accuracy often requires a complex monitoring scheme, which in-turn demands more computing power. Therefore, the monitoring scheme should be carefully designed in order to maintain both accuracy and efficiency.

Two state-of-the-art online monitoring algorithms have been recently developed: dynamic counters (DCs) monitoring [Lampka et al., 2011] and l-repetitive function (LRF) monitoring [Neukirchner et al., 2012]. Both monitoring algorithms are based on the arrival curve model that is capable of capturing arbitrary event arrival patterns in the time interval domain. The DCs monitoring assumes that an arrival curve can be conservatively approximated by a set of staircase functions and each staircase function can be monitored by a counter [Lampka et al., 2011]. The minimum of a set of dynamic counters gives the burst capacity of a system at (mission) time t. The LRF monitoring assumes that an l-repetitive function can be constructed in a maximum busy-window period, then a history of arrival time of most recent l events is kept to monitor coming events [Neukirchner et al., 2012]. The LRF is also a lightweight method of monitoring arbitrary event streams. Although both monitoring algorithms are reported to be efficient, their monitoring differences are unexplored.

Both DCs and LRF are designed to monitor the inputs with only one connection. With the exception of the one-to-one connection, many-to-one connections are also common in realistic embedded systems. To the best of our knowledge, there is no work in the literature that discusses how to monitor the many-to-one connections. In this article we discuss the monitoring method of many-to-one connections that are abstracted as OR and AND components [Jersak, 2005; Wandeler, 2006]. As shown in Fig. 1(a), OR component indicates that the output is generated whenever any input is available. An OR component is often used in the task callings. A task with a specific function is often called by many other tasks, and this task with the specific function will be activated no matter which task calls it. AND component indicates that the output is generated only when events are available in every input, as shown in Fig. 1(b). A common use of the AND component is the system synchronization. In parallel computing, a task is often activated only when all other parallel tasks return their results. The output trace from OR and AND components is complex, making it difficult to directly implement DCs and LRF to monitor these components. Therefore, a new monitoring mechanism is needed to monitor the OR and AND components.

This article investigates in depth both newly developed runtime monitoring methods, DCs monitoring and LRF monitoring, to evaluate their performance and identify their strengths and weaknesses with respect to different event arrival patterns. Better knowledge of the differences in the modeling scope, the corresponding monitoring accuracy, and computation or memory overhead for these two methods may be beneficial for using these techniques. We also develop a DCs-based method to



(a) Monitoring input with OR component. (b) Monitoring input with AND component.

Fig. 1: Diagram of OR and AND dependencies.

monitor the standard periodic burst event streams. Furthermore, for the input classes that are abstracted by AND and OR on multiple input event streams, new methods are proposed to complement the existing methods. The contributions of this article are summarized as follows.

— The performance of DCs and LRF monitoring are evaluated based on the event patterns. The strengths and weaknesses with respect to different event patterns are identified.
— The extension of DCs is explored to monitor the periodic events with jitter and periodic burst events.
— The methods of online monitoring output events with the component of AND or OR are proposed.
— All the monitoring methods are implemented in FPGA, and the timing and resource overheads of FPGA are reported.

The rest of this article is structured as follows. We review related work in the next section. In Section 3, basic background knowledge of the used techniques is described. In Section 4, we present the monitoring performance differences between the DCs method and the LRF method with different event arrival patterns. In Section 5, we present new approaches to monitor the output event with many-to-one components. A concrete case study is presented to show the DCs and LRF monitoring performance in Section 6. In Section 7, FPGA experiments show resource overhead and timing latencies of both methods. Section 8 concludes the paper.

2. RELATED WORK

Runtime monitors are used to ensure that the system runtime behaviors are constrained within a reliable range. There exists different kinds of monitoring architectures, targeted at different runtime behaviors. In the following, according to the system specifications, we group the monitoring into two aspects which are, the functionality monitoring and the timing monitoring.

In the functionality monitoring, a novel time-triggered approach was introduced in [Bonakdarpour et al., 2011; Bonakdarpour et al., 2013] to reduce the monitoring overhead and the system unpredictability. The time-triggered monitoring needs to find an optimal sampling period, so that the required auxiliary memory will be minimized and the monitor still correctly tracks the program state. In [Medhat et al., 2014], a control-theoretic approach was proposed to improve the time predictability, and ensures the soundness of verification by incorporating a maximally utilized bounded memory buffer. Experimental results of monitoring the functioning of a Toyota 2JZ engine showed that the overshoots were reduced and the time predictability was improved by applying this control-theoretic monitoring approach. The control-theoretic approach was extensively explored for coordinating time predictability and memory utilization in runtime monitoring of systems in [Medhat et al., 2015]. Three real-life cases were used to study the effect of this control-based monitoring in [Medhat et al., 2015]. Results of this work showed that the memory utilization and the time predictability were greatly improved with control-based monitoring.

The timing monitoring focuses on guaranteeing the system timing properties. A powerful model to analyze the system timing properties is Real-Time Calculus (RTC) [Thiele et al., 2000]. The arrival curves in RTC define the upper and lower bounds on the events number within a fixed interval. In [Lampka et al., 2009], a hybrid methodology was presented to analyze the system performance by using the timed automata to imitate the arrival curves. This methodology was extended to generalize more patterns, including the periodic with jitter, for the conversion of arrival curves to timed automata in [Lampka et al., 2010]. Based on this work, Lampka *et al.* subsequently proposed using DCs to predict future coming events in [Lampka et al., 2011]. Neukirchner *et al.* presented a lightweight monitoring method for arbitrary activation patterns in [Neukirchner et al., 2012]. The task activation patterns are modelled as minimum distance functions, which describe lower bounds on the temporal distance between consecutive activations. In order to reduce the monitoring overhead, an l-repetitive minimum distance function is constructed to represent the original minimum distance function.

Based on the both methods, several monitoring methods and task scheduling schemes are proposed. Huang et al. [Huang et al., 2012] prototyped DCs in FPGA to conform the runtime inputs for hard real-time systems. The LRF monitoring is extended to monitor the group of low-criticality events in mixed-criticality systems [Neukirchner et al., 2013b; Neukirchner et al., 2013a]. By using LRF to guarantee sufficient temporal independence among partitions, the interrupt latency is greatly improved in a real-time hypervisor. Although DCs and LRF are reported to be efficient, their differences have not yet been explored. In this article, based on the real-time events pattern, the monitoring differences of DCs and LRF are investigated.

Also in this article, the new schemes in monitoring AND and OR component are proposed. In [Jersak, 2005; Jersak and Ernst, 2003], Jersak *et al.* proposed to use the standard activation pattern resulting from the boolean operation on the various input event streams to embed many-to-one connections within their compositional framework [Richter et al., 2003a]. However, the analysis is not tight because the compositional framework must use a limited set of classical arrival patterns to present the input activation pattern. Haid *et al.* [Haid and Thiele, 2007] presented a tighter analysis on the delay and backlog for OR-activations and AND-activations in modular performance analysis by directly using the input activation patterns. Research to date on the many-to-one connections deals with the offline analysis on the system timing properties. In this article, the DCs method is extended to monitor the output events from the AND and OR components.

3. BACKGROUND

Both DCs monitoring and LRF monitoring share the same event model inherent to the Real-Time Calculus (RTC) [Thiele et al., 2000], and the SymTA/S approach [Henia et al., 2005], respectively. The event model is used as the connection between different resources or tasks in real-time systems. In this section, we first provide the basic definitions about event models then introduce the DCs and LRF monitoring methods.

3.1. Event models

Event models in real-time systems describe how often events (or function calls) arrive and data provided as input to the system. In Network Calculus [Le Boudec and Thiran, 2001], the concept of arrival curve was proposed to abstract the data flow. RTC extends the concepts of Network Calculus to the domain of real-time embedded systems. In RTC, the event streams are abstracted as a tuple $\alpha(\Delta) = [\alpha^u(\Delta), \alpha^l(\Delta)]$ of upper and lower arrival curve that provide an event stream model, representing all possible traces of an event stream [Wandeler, 2006].

Definition 3.1. (Arrival Curve) Let $R[s,t)$ denote the number of events arriving on an event stream in the time intervals $[s,\ t)$. Then, R, α^u and α^l are related to each other by the following inequality

$$\alpha^l(t-s) \leq R[s,t) \leq \alpha^u(t-s), \forall t \geq s \geq 0 \tag{1}$$

with $\alpha^l(0) = \alpha^u(0) = 0$.

The upper arrival curve $\alpha^u(\Delta)$ provides an upper bound on the number of events seen on the event stream in any time interval of length Δ, and analogously, the lower arrival curve $\alpha^l(\Delta)$ provides a lower bound on the number of events in a time interval Δ. In RTC, arrival curve, together with the resource model, is used to analyze system performances.

In the SymTA/S method, events are bounded by the minimal distance function and maximum distance function and are defined as follows [Henia et al., 2005]:

Definition 3.2. (Minimum Distance Function) The minimum distance function $d^{min}(n)$ specifies the minimum distance between $n+1$ consecutive events in an event stream.

Definition 3.3. (Maximum Distance Function) The maximum distance function $d^{max}(n)$ specifies the maximum distance between $n+1$ consecutive events in an event stream.

At the component level, SymTA/S uses the minimum distance function to get the busy window and based on the busy window, the system behaviors are analyzed. From these definitions, one can see that the minimum/maximum distance function and arrival curve provide the events bound with respect to time. Minimum distance function also limits the event number within a time interval, which is the same as upper arrival curve. The maximum distance function requires that the event number cannot be less than a specification, which is the same as lower arrival curve. For example, for periodic events with jitter, the arrival curves are

$$\begin{aligned} &(a)\ Upper\ Arrival\ Curve : \alpha^u(\Delta) = \left\lfloor \frac{\Delta + J}{\delta} \right\rfloor \\ &(b)\ Lower\ Arrival\ Curve : \alpha^l(\Delta) = max\left\{0, \left\lfloor \frac{\Delta - J}{\delta} \right\rfloor\right\} \end{aligned} \tag{2}$$

where δ is the period, and J is the jitter. If the periodic events are expressed with the minimum/maximum distance functions, they should be

$$\begin{aligned} &(a)\ Minimum\ Distance\ Function : d^{min}(n) = \max\{0, n \cdot \delta - J\} \\ &(b)\ Maximum\ Distance\ Function : d^{max}(n) = n \cdot \delta + J \end{aligned} \tag{3}$$

where $d^{min}(n)$ and $d^{max}(n)$ describe a lower bound and an upper bound on the time interval between the occurrences of the first and the last event in any sequence of $n+1$ consecutive events in the event trace.

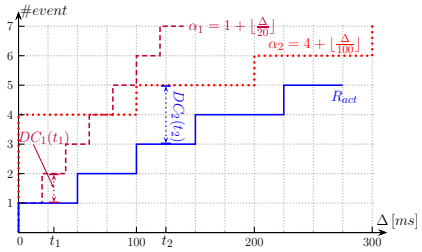
Since events should be bounded by arrival curve or minimum/maximum distance function at runtime, the monitor is used to verify that all events conform to the designed bound. Based on the arrival curve model, the DCs monitoring is proposed. Based on the minimum distance function, the LRF monitoring is proposed.

3.2. DCs monitoring

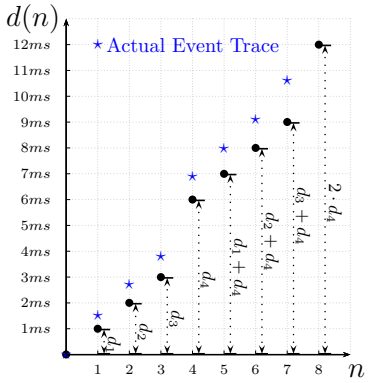
The DCs are used to predict the upper bound of future events, based on which the power mode is adaptively managed in [Lampka et al., 2011]. In this case, the constructed arrival curve should be larger than the real arrival curve. However, since

ALGORITHM 1: Implementing a dynamic counter to monitor a staircase function

Input:
signal s, $\triangleright tuple < DC_i, CLK_i >$;
Output:
1: **if** s = $CLK_i\_timeout$ **then**
2: $DC_i \leftarrow \min(DC_i{+}1, N_i^u)$
3: reset_timer(CLK_i, δ_i^u)
4: **end if**
5: **if** s = event_arrival **then**
6: **if** $DC_i = 0$ **then**
7: report_exception
8: **else**
9: **if** $DC_i = N_i^u$ **then**
10: reset_timer(CLK_i, δ_i^u)
11: **end if**
12: $DC_i \leftarrow DC_i - 1$
13: **end if**
14: **end if**



(a) An example for an arrival curve as the combination of staircase functions. The arrival curve is $\alpha_{min} = \min(\alpha_1, \alpha_2)$. R_{act} is the actual event arrival trace. $DC_1(t_1)$ denotes the potential burst capacity with the constraint of staircase function α_1 at the time t_1. $DC_2(t_2)$ denotes the potential burst capacity with the constraint of staircase function α_2 at the time t_2.



(b) An example of an l-repetitive minimum distance function. The black dot denotes the minimum distance function. The blue star denotes the actual event trace. The interval of successive events should not be smaller than the minimum distance function

Fig. 2: The two types of approximation curve.

false positives are not allowed if the monitor is used to monitor events, the constructed arrival curve for the monitor to guard should be less than the real arrival curve.

In principle, any monotonous and time-invariant arrival curve can be conservatively approximated as the minimum on a set of staircase functions with the form $\alpha_i^u(\Delta) = N_i^u + \lfloor \frac{\Delta}{\delta_i} \rfloor$, where N_i^u is the initial value of this staircase function and δ_i is its stair length [Lampka et al., 2009].

$$\forall \Delta \in \mathbb{R}_{\geq 0} : \alpha^u(\Delta) \geq \min_{i=1..n} (\alpha_i^u(\Delta)). \tag{4}$$

An example is shown in Fig. 2(a). An upper arrival curve is the minimum of two staircase functions. For every staircase function (α_i^u), a dynamic counter (DC_i) and a timer (CLK_i) are used to conform events to it. The detail algorithm is shown in Algo. 1 [Lampka et al., 2011]. The use of DC_i in Algo. 1 is to update the potential burst the system can accept during the runtime. The maximum value of DC_i is N_i^u, which indicates that the burst events should not exceed $\min_{i=1..n} N_i^u$. The timer CLK_i is used to check the updating time of DC_i. The violation happens when $DC_i = 0$.

3.3. LRF monitoring

ALGORITHM 2: Implementing l-repetitive function to monitor an event stream

Input:
current time, trace buffer$[l]$, $d[l]$;
Output:
1: **for** $i \in [0, l-1]$ **do**
2: **if** current time - trace buffer$[i] < d[i]$ **then**
3: report_exception
4: **end if**
5: **end for**
6: right shift trace buffer
7: trace buffer$[0]$ = current time

Arrival events are also bounded by the minimum distance function. An l-repetitive distance function is a special minimum distance function that satisfies the following condition:

$$d(n) = \begin{cases} d_n(given), & n \leq l, \\ \max\limits_{\omega \in [1,l]} (d(\omega) + d(n-\omega)), & n > l. \end{cases} \tag{5}$$

Figure 2(b) shows an l-repetitive minimum distance function with $l = 4$. For such l-repetitive function, it has been shown that the arrival time of the most recent l events is sufficient to verify whether or not future events will conform to the predefined minimum distance function. The detail monitoring algorithm with l-repetitive function is shown in Algo. 2 [Neukirchner et al., 2012]; the arrival time of most recent l events is maintained. Everytime a new event arrives, the time gap between the current event and the past l events is used as a comparison to the minimum distance function. If the time gap is smaller than the designed minimum distance, the exception is reported.

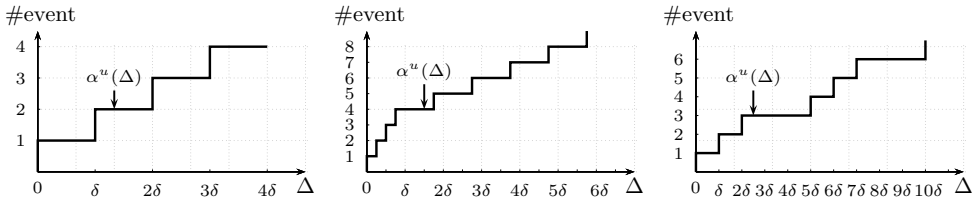
In [Neukirchner et al., 2012], authors propose to use a part of minimal distance function to represent distance function. This proposal is based on two assumptions:

ASSUMPTION 1. *For verification of timing constraints, only a part of the minimal distance function is relevant. The relevant domain is* $[1, n_{max}]$*, where* n_{max} *depends on the scheduling policy of the resource.*

ASSUMPTION 2. *The execution sequence of tasks on a resource only depends on the number of pending activations (events) of all tasks and the state of the scheduler (if it is stateful).*

As implied by these two assumptions, the distance between two events, which are sufficiently far apart, does not influence the scheduling on a resource. For the schedulers that can be analyzed with busy-window approach, such as static priority preemptive (SPP), rate monotonic scheduling (RMS) and earliest deadline first (EDF), the above two assumptions hold. At runtime, when the system reaches an idle state, i.e., no pending events, the events history can be neglected, as the past events no longer impact the system.

From these assumptions, it is deduced that only a part of the d function is relevant for verification of timing constraints. The relevant domain is $[1, n_{max}]$, where n_{max} is the number of events in the maximum system busy-window period [Tindell et al., 1994]. Then we only need to use the part of $[1, n_{max}]$ of minimum distance function in constructing the l-repetitive function. The detailed method of constructing l-repetitive function is explained in [Neukirchner et al., 2012].



(a) Periodic events (b) Periodic events with an initial burst (c) Periodic burst events

Fig. 3: Arrival curves of three types of event stream.

4. MONITORING ALGORITHM COMPARISON BASED ON REPRESENTATIVE ARRIVAL CURVES

As presented in the previous section, DCs monitoring works by using a set of staircase functions to approximate the upper arrival curve, while LRF monitoring works by using the repetitive segment to approximate the minimum distance function. The accuracy of using these two methods for monitoring different typical event streams depends on the approximated degree.

In this section, according to the specifications of arrival curve, the event streams are categorized into three types. The first type is the periodic events, whose arrival curve is the staircase function. The second type is the periodic or sporadic events with an initial burst, whose arrival curve is the staircase function with an inital burst. The third type is the periodic burst events. The arrival curves of these three types of event streams are shown in Fig. 3. In the following, the monitoring performance is investigated towards monitoring the three types of event streams. The DCs monitoring method is improved towards some specific event arrival curves.

4.1. Periodic events

The upper arrival curve of the periodic or sporadic events can be modeled as the staircase function, as shown in Fig. 3(a). It indicates that the minimum temporal gap between two successive events is δ. For this kind of arrival curve, either DCs monitoring with one counter or LRF monitoring with $l = 1$ can be used to monitor the periodic event. As one staircase function is guarded by one counter, there is no error for DCs monitoring. For LRF monitoring, by keeping the arrival time of the most recent event, the minimum temporal gap between the coming event and the previous event can be strictly guaranteed. Therefore, there is no error.

4.2. Periodic events with an initial burst

The assumption of periodic events is overly restrictive. Many periodic events exhibit a so called initial burst that captures the distortion that a periodic event stream might experience. Figure 3(b) shows the upper arrival curve of a periodic event stream with an initial burst. A widely used model to specify the event stream with an initial burst is the PJD model, where the arrival curve is characterized by period p, jitter j, and minimal interarrival distance d. In the PJD model, the upper arrival curve can be determined as $\alpha^u(\Delta) = \min\{\lceil \frac{\Delta+j}{p} \rceil, \lceil \frac{\Delta}{d} \rceil\}$ and can be represented as the minimum of two staircase functions. The parameters of the two staircase functions can be computed as follows [Lampka et al., 2009]:

$$\begin{cases} N_1^u = \lceil \frac{j}{p} \rceil + 1,\ \delta_1^u = p; & if\ d = 0 \vee d \leq p - j, \\ N_1^u = \lceil \frac{j}{p} \rceil + 1,\ \delta_1^u = p;\ N_2^u = 1,\ \delta_2^u = d; & if\ d > 0 \wedge d > p - j. \end{cases} \tag{6}$$

ALGORITHM 3: Implementing a dynamic counter to monitor the periodic event with jitter

Input: signal s, $\triangleright tuple < DC_i, CLK_i >$

```
1: if s = CLK_i_timeout then
2:    if DC_i = N_i then
3:       reset_timer(CLK_i, δ_i − J'_i)
4:    else
5:       DC_i ← min(DC_i+1,N_i)
6:       reset_timer(CLK_i, δ_i)
7:    end if
8: end if
9: if s = event_arrival then
10:   if DC_i = 0 then
11:      report_violation
12:   else
13:      if DC_i = N_i ∧ CLK_i < δ_i − J'_i then
14:         reset_timer(CLK_i, δ_i − J'_i)
15:      end if
16:      DC_i ← DC_i-1
17:   end if
18: end if
```

This approximation of Eq. 6 is not conservative. For the hard real-time system, the approximated arrival curve is not allowed to exceed the actual upper arrival curve. Otherwise, some violated events will not be verified as the non-violation event, which harms the safety of the system. To make the approximated arrival curve lower than the real arrival curve, $N_1^u = \lceil \frac{j}{p} \rceil + 1$ should be changed to $N_1^u = \lceil \frac{j}{p} \rceil$. The DCs and LRF method in monitoring periodic events with an initial burst are discussed as follows.

4.2.1. DC*s monitoring.* The DCs method proposed in [Lampka et al., 2011] can only guard the arrival curve represented as the overlay of staircase functions. There are also some arrival curves, like periodic event stream with jitter, that cannot be exactly represented as the overlay of staircase functions. The arrival curve of periodic event stream with jitter is represented as a shifted staircase function. In [Lampka et al., 2010], the timed automata that can generate the periodic event trace is extended to generate the periodic event trace with jitter by using a factor to refine the period and offset. It is theoretically possible to use DCs to accurately monitor the periodic events with jitter by using the similar approach as extending the timed automata from [Lampka et al., 2010]. However, in this article, we propose another new approach which can also take the jitter into account.

For a periodic event stream with jitter, if the jitter is greater than the period, there will be an initial burst [Wandeler, 2006].

$$\alpha_i(\Delta) = \left\lceil \frac{\Delta + J_i}{\delta_i} \right\rceil = N_i + \left\lfloor \frac{\Delta + J_i'}{\delta_i} \right\rfloor, \ (0 < J_i' < \delta_i), \tag{7}$$

where N_i is the initial burst. In this case, DCs can monitor the arrival curve with $J_i' = 0$. If $J_i' \neq 0$, J_i' will be skipped by the DCs monitoring. Being aware that this kind of arrival curve is the staircase function left shifted by jitter, the idea of monitoring this type of event stream is to include the jitter in the DCs monitoring algorithm. Before introducing the new algorithm, we provide a definition of initial phase. The initial phase is used to introduce an offset of J_i' in the DCs monitoring.

Definition 4.1. (Initial Phase) For a periodic event stream with jitter, suppose the period is δ_i and the jitter is J_i, the initial phase is referred as $\delta_i - J_i'$, where $J_i' = J_i \bmod \delta_i$.

The improved DCs algorithm is shown in Algo. 3. Compared to the monitoring method of DCs in [Lampka et al., 2011], there are two more considerations of using DCs to monitor the periodic events with jitter. The first consideration is that the initial phase of DCs monitoring is used to reduce the temporal gap between the burst events and the following event. Since the next event after the initial burst is allowed after $\delta_i - J_i'$, the initial value of a timer is already set as $\delta_i - J_i'$. The second consideration is

the judge of the renewal point. The renewal point is the time the potential burst DC_i and the timer returns back to the initial burst and initial phase, respectively. Since there exists an offset of J_i' of arrival curve compared to staircase function, the timer of a renewal point should be equal to or less than the initial phase.

There are two triggering signals in Algo. 3. One is the timeout and the other is the event arrival. If $J_i' = 0$, Algo. 3 is the same with DCs monitoring in Algo. 1. If $J_i' \neq 0$, the timer starts from the phase of $\delta_i - J_i'$. The timer goes down with the system clock. When the timer comes to 0, a timeout happens. This is a sign that one more event can be accepted. The renewal point in Algo. 3 is referred to as the state $DC_i = N_i$ and $CLK_i < \delta_i - J_i'$ (line 3 and line 14). If a renewal point is found by the Algo. 3 when timeout happens, the timer phase is reset to $\delta_i - J_i'$. If not, the timer phase is reset to δ_i. The event arrival will trigger the violation verification. If $DC_i = 0$, the new arrival event is a violation. If not, the event is accepted by the system, and DC_i should be reduced by 1.

Based on the Algo. 3, for any time interval $(s, t]$, DC_i can be computed as:

$$DC_i(t) = \min\left(DC_i(s) + \left\lfloor \frac{t - s + J_i'}{\delta_i} \right\rfloor - R(s,t),\ N_i\right) \tag{8}$$

where $R(s,t)$ denotes the number of arrival events in the time interval $[s,t)$. The Eq. 8 can guarantee that arrival events will never run against the arrival curve. Besides, the jitter will not be skipped by Algo. 3. Since the improved DCs method can consider the case when the jitter exists, the PJD type whose arrival curve is $\min\{\lceil\frac{\Delta+j}{p}\rceil, \lceil\frac{\Delta}{d}\rceil\}$ can be fully guarded by DCs method.

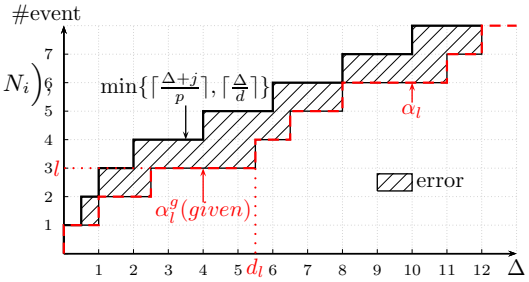


Fig. 4: Arrival curve of periodic events with jitter (black solid line) that is the minimum of two staircase functions. The red dash line represents a more conservative arrival curve that is segmentally repetitive and is used in LRF monitoring. The shadow area shows the error between the arrival curve and the approximated curve.

4.2.2. LRF *monitoring.* For the LRF monitoring, a repetitive segment should be constructed. Suppose the first repetitive segment is α_l^g as given in the offline analysis. The arrival curve for the l-repetitive minimum distance function can be expressed as follows:

$$\alpha_l(\Delta) = \begin{cases} \alpha_l^g(\Delta)(given), & if\ \Delta \leq d_l, \\ k \cdot l + \alpha_l^g(\Delta - k \cdot d_l), & if\ \Delta > d_l, \end{cases} \tag{9}$$

where k is the largest integer that makes $\Delta - k \cdot d_l < d_l$ when $\Delta > d_l$.

If the arrival curve of periodic events with an initial burst is modeled as $\min\{\lceil\frac{\Delta+j}{p}\rceil, \lceil\frac{\Delta}{d}\rceil\}\,(p > d > 0)$, there is a gap between α_l and α^u, as shown in Fig. 4. Since the monitoring only allows the false negatives for the hard real-time system, the approximated arrival curve of repetitive segment should always be lower than the actual arrival curve.

The monitoring accuracy by LRF monitoring depends on the closeness of the approximated arrival curve and the original arrival curve, i.e., how much the error area in Fig. 4 is. We therefore define the monitoring error to represent the monitoring accuracy of verifying events based on the approximated arrival curve.

Definition 4.2. (Monitoring Error) Suppose the actual arrival curve is α^u and the approximated arrival curve is $\widetilde{\alpha}$. For the hard real-time system, $\alpha^u \geq \widetilde{\alpha}$. The

monitoring error (ME (Δ)) of verifying arrival events based on $\widetilde{\alpha}$ is referred to that within an interval of Δ,

$$ME(\Delta) = \left| \frac{\int_0^{\Delta} (\alpha^u(t) - \widetilde{\alpha}(t)) \mathrm{d}t}{\widetilde{\alpha}(\Delta)} \right|. \tag{10}$$

From Fig. 4, it can be found that the ME reflects the worst-case conforming delay for every event. Suppose events arrive as early as possible, that is, the runtime arrival pattern is exactly the same as $\alpha^u(t)$. If the monitor conforms all events to comply with $\widetilde{\alpha}(t)$, the unnecessary delay per event within Δ is represented by the ME of Eq. 10. The smaller the ME is, the closer to $\alpha^u(t)$ the $\widetilde{\alpha}(t)$ is. Hence, the ME represents the monitoring accuracy.

We can see that the ME is a function with respect to two arrival curves and the monitoring interval. To reflect the monitoring accuracy in a long time, the monitoring interval should be large.

4.3. Periodic burst events

In contrast to periodic events with an initial burst, events burst happens periodically for the periodic burst event stream. The LRF method is capable of monitoring periodic burst events. As the arrival curve of periodic burst events (as shown in Fig. 3(c)) is segmentally repetitive, the LRF method can monitor the events by keeping the trace of the repetitive segment. Specifically, suppose the length of repetitive segment is d_l and contains l events. Keeping most recent l events is sufficient to conform events as the whole arrival curve for the LRF monitoring.

The arrival curve of periodic burst events is the non-convex pattern. According to [Lampka et al., 2010], the non-convex pattern can be handled by making use of subsets of convex patterns and local synchronization for obtaining local minima and maxima. Following this indication, we provide a detailed method for monitoring the periodic burst events.

There are two kinds of periodic burst events: standard and nonstandard. In the following, we discuss how to apply DCs method to monitor the standard and nonstandard periodic burst events.

4.3.1. Standard periodic burst events. The authors in [Neukirchner et al., 2012] claim that DCs algorithm is invalid to monitor periodic burst events. In this article, a new scheme using DCs is developed to monitor standard periodic burst events.

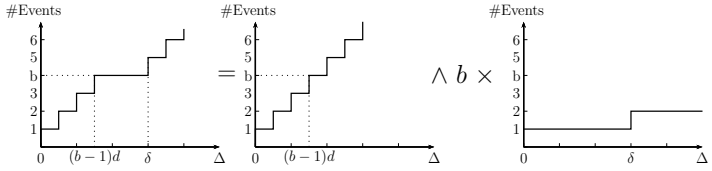


Fig. 5: The diagram of periodic burst pattern equivalence.

The standard periodic burst events model is characterized by three parameters, which are, an interval δ, a minimum timing separation d between successive events, the maximum events b within the interval δ, as shown in Fig. 5. The upper arrival curve can be expressed as follows [Richter et al., 2003b],

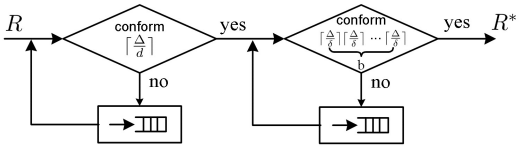


Fig. 6: The flow of monitoring periodic burst arrival events.

$$\alpha^u(\Delta) = \left\lfloor \frac{\Delta}{\delta} \right\rfloor b + \min\left(\left\lceil \frac{\Delta - \lceil \frac{\Delta}{\delta} \rceil b}{d} \right\rceil, b \right). \tag{11}$$

ALGORITHM 4: Online monitoring of a periodic burst arrival event pattern.

Input:
signal s_d, s_δ, $tuple < DC^d, CLK^d >$, $tuple < DC_i^\delta, CLK_i^\delta >$ and event queue q_1, q_2;
Output:
1: Algorithm 1 $\leftarrow$ (s_d, $tuple < DC^d, CLK^d >$)
2: Algorithm 1 $\leftarrow$ (s_δ, $tuple < DC_i^\delta, CLK_i^\delta >$)
3: **if** report_exception(s_d) **then**
4: q_1.enqueue();
5: **end if**
6: **if** report_exception(s_δ) **then**
7: q_2.enqueue();
8: **end if**
9: **while** q_1.length()>0 $\wedge$ $DC^d > 0$ **do**
10: q_1.dequeue
11: $DC^d = DC^d - 1$
12: **end while**
13: **while** q_2.length()>0 $\wedge$ $\max(DC_i^\delta) > 0$ **do**
14: q_2.dequeue
15: **for** $i \leftarrow 1$ to n **do**
16: **if** $DC_i^\delta > 0$ **then**
17: $DC_i^\delta = DC_i^\delta - 1$
18: break
19: **end if**
20: **end for**
21: **end while**

Although the periodic burst arrival curve cannot be approximated as the minimum of a set of staircase functions, it can be equivalent to a special logic composition of staircase functions. As shown in Fig. 5, periodic burst arrival curve is decomposed to a staircase function with a period of d and b staircase functions with a period of δ. A counter corresponding to the period d and b counters corresponding to the period δ are used to guarantee that the minimal distance between two events is larger than d and the number of events within any δ interval is smaller than b.

The detailed procedures are shown in Fig. 6 where R represents the events that are monitored and R^* represents the output events trace. The monitor first checks whether or not the arrival events comply with $\lceil \frac{\Delta}{d} \rceil$. If not, a buffer is used to store the arrival events. If yes, the monitor checks whether or not the arrival events comply with the b staircase functions. b staircase functions are assumed to be equivalent, i.e., any event is acceptable if it does not violate any of the respective staircase functions. If no counters can accept an event, this event is a violation and will be delayed by buffer. Note that the two buffers in Fig. 6 are separated and the buffer obeys first-in-first-out principle. The pseudo code of this monitoring algorithm is shown in Algo. 4.

4.3.2. Nonstandard periodic burst events. For nonstandard periodic burst events, the events burst is periodic but the timing separation between two events within the burst is different. If DCs monitoring is used in conforming events as periodic burst events, the minimum of a set of staircase functions ($\min\limits_{i=1..n}\{\alpha_i\}$, $\alpha_i = N_i^u + \lfloor \frac{\Delta}{\delta_i} \rfloor$) is assumed to approximate α_l, where $\delta_{max} = \max\limits_{i=1..n}\{\delta_i\}$, as shown in Fig. 7. In the case that $\Delta = +\infty$, we have $\min\limits_{i=1..n}\{\alpha_i\} = N_n^u + \lfloor \frac{\Delta}{\delta_n} \rfloor$, where $\delta_n = \delta_{max} = \max\limits_{i=1..n}\{\delta_i\}$. If we require

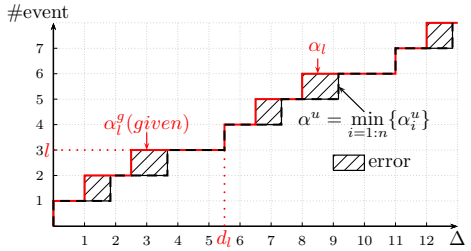


Fig. 7: Arrival curve of periodic burst events (red solid line) that is segmentally repetitive. The black dash line represents a more conservative arrival curve that is the minimum of a set of staircase functions and is used in DCs monitoring.

$$\lim_{\Delta=+\infty}\left\{\alpha_l(\Delta) - \min_{i=1..n}\{\alpha_i(\Delta)\}\right\} = \frac{\Delta}{d_l}l - N_x^u - \left\lfloor \frac{\Delta}{\delta_{max}} \right\rfloor \geq 0, \quad (12)$$

we have to set that $d_l \leq l \cdot \delta_{max}$. The ME (defined in Eq. 10) can also be used to represent the monitoring accuracy.

5. MONITORING MANY-TO-ONE CONNECTIONS

In the many-to-one connection that is abstracted as the OR or AND dependency, the output arrival curve is often irregular. For the irregular arrival curve, the DCs and LRF methods rely on using more constraint arrival curves to approximate OR or AND component. But this approximation may not be tight, especially in cases where the difference between the approximated arrival curve and the irregular arrival curve is large. Therefore, another more accurate monitoring method is needed for monitoring the output event with the OR or AND component.

From [Haid and Thiele, 2007], we found that the output arrival curve of OR or AND component is an algebraic combination of input arrival curves. The logical relationship inherent to this type of algebraic combination can be explored by using DCs to verify whether the output events of OR or AND components comply with the output arrival curve. It is not easy to apply the LRF method to monitor output events. There are two main reasons for this. On the one hand, the algebraic combination between input arrival curve and output arrival curve cannot be easily explored. The LRF method relies on the minimum distance function. Although an algebraic combination exists between input arrival curve and output arrival curve, no such relationship exists between input minimum distance function and output minimum distance function. On the other hand, the LRF method relies on constructing the l-repetitive function. Since the output minimum distance functions may be very irregular, it is difficult to find a tight l-repetitive function that can closely represent the output minimum distance function. Based on the two reasons, we only focus on using DCs to monitor the output events of OR or AND component.

5.1. OR component

The definition of OR component implies that the output event is generated each time an event is available from any input. So the following theorem holds [Haid and Thiele, 2007].

THEOREM 5.1. *Assume an output with the OR dependency on n event inputs. The input event streams are modeled as abstract event streams with arrival curve α_1^u, α_2^u, ..., α_n^u. Then, the output arrival curve can be modeled by the arrival curve α_{or}^u that*

$$\alpha_{or}^u = \alpha_1^u + \alpha_2^u + ... + \alpha_n^u. \tag{13}$$

From Eq. 13, the output arrival curve of OR component is the sum of input arrival curve indicating that output events can be accepted only if the sum of arrival events is not over α_{or}^u. This is a kind of group monitoring, which is different from setting a monitor in every input. The monitoring method that attributes a monitor in every input requires that the input events cannot exceed the respective input arrival curve. In the real case, there is only a restriction on the sum events instead of a restriction in every input. By setting only one monitor in the OR component, the system utilization can be improved as one input may exceed its budget at the cost of another input.

The problem of monitoring output events with OR component is formulated as follows. Given an OR component with n periodic-with-jitter inputs, how do you guarantee that the output events will not exceed the output arrival curve by monitoring the output event. As the periodic-with-jitter inputs are common in real-time systems [Phan and Lee, 2013], we concentrate on the monitoring methods with all periodic-with-jitter inputs.

The critical idea of monitoring event stream constrained by α^u_{or} is that DCs are used to bound every input arrival curve and the sum of input arrival curve can be bounded by the DCs. According to the subsection 4.2, the monitoring of a periodic event stream with jitter only needs one dynamic counter. For the n periodic event streams with jitter, n DCs are needed. Since the connection is boolean OR dependency, only one counter is needed to be responsible for a coming event. The coming event is the violation only when all counters are 0. Then the problem is how to choose the responsible counter that can strictly bound the output events by α^u_{or}.

Before providing the solution of finding out the responsible counter, an example of monitoring OR component is first introduced.

Example 5.2. In an OR connection with two inputs, the two input arrival curves are $\alpha_1 = 1 + \lfloor \frac{\Delta}{3} \rfloor$, $\alpha_2 = 1 + \lfloor \frac{\Delta}{2} \rfloor$. Then the output arrival curve is $\alpha^u_{or} = 2 + \lfloor \frac{\Delta}{3} \rfloor + \lfloor \frac{\Delta}{2} \rfloor$. Two DCs are used to bound the output events, where DC_1 is for α_1 and DC_2 is for α_2. When the first event is checked at the output, the system capacity is reduced by 1, no matter which counter is responsible for this event. From α^u_{or}, the system capacity after 2 time units will return to 2. If DC_1 is responsible for the first event, it will take 3 time units for the system capacity to go back to 2, which is contradicted with α^u_{or}. So DC_2 should be responsible for the first event.

From Ex. 5.2, the responsible counter for an event is the counter that can give the system the largest capacity as quickly as possible. From the Eq. 13, the system capacity at any time t is:

$$capacity(t) = \sum_{k=1}^{n} DC_k(t). \tag{14}$$

LEMMA 5.3. *For any coming event, if a counter can be found as a responsible counter that makes the system capacity the greatest in the shortest time, the output events can be strictly bounded as α^u_{or} by using n DCs.*

PROOF. There are two requirements that the output events with α^u_{or} are strictly bounded by using n DCs, i.e., there are no false negatives and no false positives.

No false negatives: From α^u_{or}, the output events can be accepted in the worst case: all input events arrive as the designed input arrival curve. As for every output event, n DCs can give the system the greatest capacity, which indicates that worst-case output events can be accepted. If one event is checked as the violated event, then this event violates the worst case, which also violates the α^u_{or}.

No false positives: Since n DCs are used to bound the sum of n periodic event streams with jitter. Every counter can bound an input event stream, the total input event streams are thus bounded by using n DCs. □

THEOREM 5.4. *Assume $DC_i(t)$ and $CLK_i(t)$ are the values of counters and timers at the time t in monitoring output events with OR component. If at the time t_0, an output event is checked, where t_0^- and t_0^+ denote the time before and after the monitoring trigger. The responsible counter is the counter with the smallest Δ', where Δ' satisfies*

$$\left\lfloor \frac{\Delta' + \delta_i - CLK_i(t_0^+)}{\delta_i} \right\rfloor = N_i^u + 1 - DC_i(t_0^-). \tag{15}$$

The proof is seen in Them. 9.1 in the Appendix.

Δ' in Eq. 15 is called the capacity influence time (CIT) in this article. The CIT can be directly calculated by Eq. 15. The Algo. 5 is used to obtain the counter with the smallest CIT. Due to the fact that in Algo. 3 a renewal point is checked whenever an event is accepted, and $CLK_i(t_0^+)$ may be different from $CLK_i(t_0^-)$ in a renewal point,

in line 2 of Algo. 5 $CLK_i(t_0^+)$ is first checked. In line 3, CIT is calculated. In line 5, the counter with the smallest CIT is selected.

ALGORITHM 5: the counter with the smallest CIT

Input: $[DC_i, CLK_i]$.

1: **for** $i \leftarrow 1$ **to** n **do**
2: $\quad CLK_i^\dagger \leftarrow (DC_i = N_i^u \wedge CLK_i > J_i')?J_i' : CLK_i$
3: $\quad CIT_i \leftarrow (N_i^u - DC_i) * \delta_i^u + CLK_i^\dagger$
4: **end for**
5: **return** i, where $CIT_i = \min_{k=1}^{n}\{CIT_k\}$

5.2. AND component

AND component indicates that the output event is generated only if all events are available in each input. Thus the following theorem holds [Haid and Thiele, 2007].

THEOREM 5.5. *Assume a component by AND connection with two input event streams that are modeled as abstract event streams with arrival curve* $[\alpha_1^u, \alpha_1^l]$ *and* $[\alpha_2^u, \alpha_2^l]$. *Then, the output event streams can be modeled with the arrival curve*

$$\alpha_{and}^u = \max\left\{ \min\{\alpha_1^u \oslash \alpha_2^l + B_1^0 - B_2^0, \alpha_2^u\}, \min\{\alpha_2^u \oslash \alpha_1^l + B_2^0 - B_1^0, \alpha_1^u\}\right\}, \tag{16}$$

where B_1^0 *and* B_2^0 *denote the initial buffer fill level of the two input buffering ports, with the constraint that* $\min\{B_1^0, B_2^0\} = 0$*, i.e., one of the buffers must initially be empty.*

The idea of monitoring output events with AND component is that, the output arrival curve from AND component is derived as the maximum of a set of periodic-with-jitter models. Dynamic counters can then be used to monitor the set of periodic-with-jitter models.

LEMMA 5.6. *For two periodic event streams with jitter (e.g.,* $\alpha_1 = [\alpha_1^u, \alpha_1^l]$, $\alpha_2 = [\alpha_2^u, \alpha_2^l]$, *where* $\alpha_1^u = \lceil \frac{\Delta + J_1}{\delta_1} \rceil$, $\alpha_1^l = \max(0, \lfloor \frac{\Delta - J_1}{\delta_1} \rfloor)$, $\alpha_2^u = \lceil \frac{\Delta + J_2}{\delta_2} \rceil$, $\alpha_2^l = \max(0, \lfloor \frac{\Delta_2 - J_2}{\delta_2} \rfloor)$, $(\delta_1 < \delta_2)$*, then* $\forall \Delta$, $\alpha_1^u \oslash \alpha_2^l(\Delta) = +\infty$, *and* $\alpha_2^u \oslash \alpha_1^l(\Delta) = \alpha_2^u(\Delta + \delta_1 + J_1)$.

The proof is seen in the Appendix.

With Lemma 5.6, the following theorem holds:

THEOREM 5.7. *For two input streams with AND component, the output arrival curve is* $\alpha_{and}^u = \max\{\alpha_2^u, \min\{\alpha_{2'}^u, \alpha_1^u\}\}$, *where* $\alpha_{2'}^u = \alpha_2^u(\Delta + \delta_1 - J_1) + B_2^0 - B_1^0$.

PROOF. From Lemma 5.6, it can be concluded that $\min\{\alpha_1^u \oslash \alpha_2^l + B_1^0 - B_2^0, \alpha_2^u\} = \alpha_2^u$, and $\min\{\alpha_2^u \oslash \alpha_1^l + B_2^0 - B_1^0, \alpha_1^u\} = \min\{\alpha_2^u(\Delta + \delta_1 - J_1) + B_2^0 - B_1^0, \alpha_1^u\} = \min\{\alpha_{2'}^u, \alpha_1^u\}$, where $\alpha_{2'}^u = \alpha_2^u(\Delta + \delta_1 - J_1) + B_2^0 - B_1^0$. Then from Eq. 16, $\alpha_{and}^u = \max\{\alpha_2^u, \min\{\alpha_{2'}^u, \alpha_1^u\}\}$. □

Since α_2^u, $\alpha_{2'}^u$ and α_1^u are staircase functions with jitter, 3 DCs may be needed to monitor the event streams with α_2^u, $\alpha_{2'}^u, \alpha_1^u$. Note that, if periods of all input event streams are the same, the needed counter number may be only one. Let DC_2, $DC_{2'}$, and DC_1 denote the counter values w.r.t. α_2^u, $\alpha_{2'}^u$ and α_1^u. From $\max\{\alpha_2^u, \min\{\alpha_{2'}^u, \alpha_1^u\}\}$, it is deduced that every counter is responsible for the arrival events, which means all counters should be reduced by one for accepting one event. To constrain that an event will not violate $\max\{\alpha_2^u, \min\{\alpha_{2'}^u, \alpha_1^u\}\}$, it is required that $\max\{DC_2, \min\{DC_{2'}, DC_1\}\}$ is not less than 0. The details about $\min$ and $\max$ in monitoring can be seen in [Lampka et al., 2011; Huang et al., 2012].

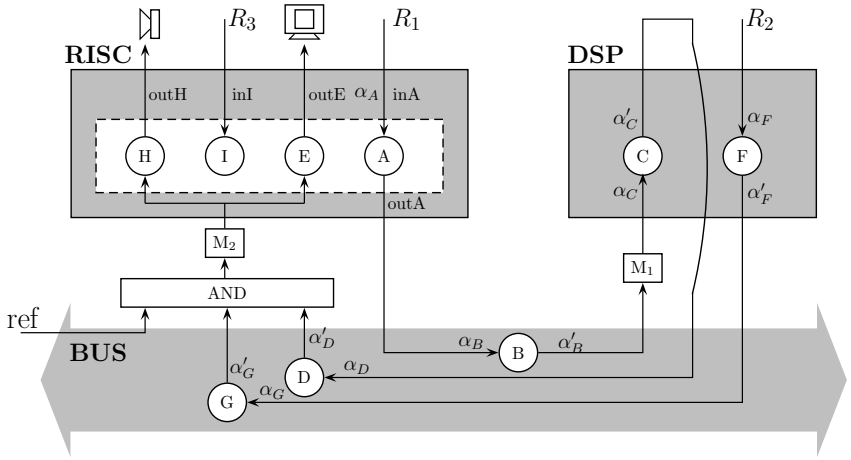


Fig. 8: Analysis model of the MPEG-2 decoder mapped onto an MpSoC platform

The aforementioned approach is not limited to monitoring the output events by AND connection with only two input event streams. For more input event streams, this approach can also be applied, as stated in the following theorem.

THEOREM 5.8. *DCs can also be used to monitor the output event stream by AND connection with more than 2 periodic event inputs with jitter.*

PROOF. Assume the number of inputs is $n (n \geq 2)$, and the arrival curve in every input is denoted as α_i (where $\delta_1 < \delta_2 < ... < \delta_n$, δ_i is the period). The output arrival curve is derived as

$$\alpha^u_{and} = \max\{\alpha'_1,\ \alpha'_2,\ ...,\ \alpha'_n\}, \tag{17}$$

where $\alpha'_i = \min\{\bigcup_{j=1,j\neq i}^{n} \{\alpha^u_j \oslash \alpha^l_i\},\ \alpha^u_i\}$. Assume $\delta_1 < \delta_2 < ... < \delta_n$. Then for $\alpha^u_j \oslash \alpha^l_i$, if $j < i$, $\alpha^u_j \oslash \alpha^l_i$ is infinity. If $j > i$, then $\alpha^u_j \oslash \alpha^l_i$ is left shifted $\delta_i - J_i$ of α^u_j. So $\alpha'_i = \min\{\bigcup_{j=1,j\neq i}^{n} \{\alpha^u_j \oslash \alpha^l_i\},\ \alpha^u_i\}$ is the minimum of a set of staircase functions with jitter. Therefore, α^u_{and} is the maximum of a set of minimum of staircase functions with jitter. DCs can be used to monitor the event stream constrained by α^u_{and}. □

5.3. Effectiveness

In this section, we extend the DCs method into monitoring the OR and AND components. It is theoretically proven that, if the event streams that flow into the OR and AND components are periodic-with-jitter patterns, the presented DCs method is sufficient. The arrival curve that the DCs method guards is the same as the original arrival curve. In the case that the input event streams are not periodic-with-jitter patterns, the presented DCs method in this section is invalid. A conservative arrival curve that is the minimum of a set of staircase functions or is a repetitive function may need to be constructed so that the DCs or LRF monitoring methods can be applied, where the monitoring accuracy lies on the ME defined in Eq. 10.

6. CASE STUDY: MPEG-2 DECODER

The MPEG-2 Decoder is used as a case study in this section to demonstrate the applicability and effectiveness of our proposed monitoring methods.

Table I: Tasks and Their Specifications

Stream	Task	Function	Best/Worst case execution time (10^3 cycles)	Priority
video	A	VLD, IQ, IS	[5, 10]	2
	B	data transfer	[1, 1]	2
	C	IDCT, MC	[3, 4]	1
	D	data transfer	[1, 1]	1
	E	assemble video-frames	[0.1, 0.1]	3
audio	F	DEC, IMDCT, SYN	[500, 1000]	2
	G	data transfer	[100, 100]	3
	H	assemble audio-frames	[10, 10]	1
-	I	play back control	[2, 2]	4

MPEG-2 decoder is used to decode the video and audio. This application is processed in a heterogeneous platform with a RISC processor (750 MHz) and a DSP processor (250 MHz). The two processors communicate with each other by a BUS (125 MHz). The specific subtasks of MPEG-2 decoder are mapped to the two processors as shown in Fig. 8, which is the same as [Haid and Thiele, 2007]. The specifications of the subtasks are seen in Table. I, where all parameter values come from [Haid and Thiele, 2007]. An AND component is used to synchronize the assembly of video stream, audio stream, and a referred basis line. All tasks are scheduled by the non-preemptive fixed-priority scheduling policy.

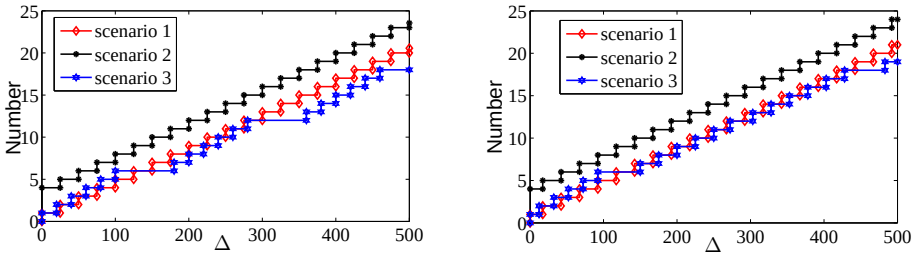
In Fig. 8, R_1 represents the input video, and R_2 represents the input audio. There are three scenarios about the arrival patterns of R_1 and R_2, which are shown as follows:

— Scenario 1: the input arrival patterns of R_1 and R_2 are set as the strictly periodic pattern. The arrival speed is set as 40 $frames/s$.
— Scenario 2: the input arrival patterns of R_1 and R_2 are set as periodic pattern, while with 3-period jitter. The arrival speed is set as 40 $frames/s$.
— Scenario 3: the input arrival patterns of R_1 and R_2 are set as periodic burst pattern whose specific parameters are that $\delta = 180\,ms,\ b = 6,\ d = 20\,ms$, as shown in Eq. 11.

Scenario 1 is a very common situation that many videos and audios need to comply with. Scenario 2 is also a possible situation, because there might be other workload on the second bus that delivers video and audio streams to RISC and DSP. Such workload may lead to the jitter of R_1 and R_2. Scenario 3 is the artificial scenario where R_1 and R_2 are conformed to the periodic burst pattern on purpose. In these scenarios, we set one monitor named M_1 to ensure the correctness of video frames passing from BUS to DSP, and another monitor named M_2 to ensure the correctness of the AND component. The ref input is set as the strictly periodic frame stream with the speed of 40 $frames/s$. Under these three scenarios, the arrival curves that the monitor M_1 and M_2 should comply with can be known by applying the Real-Time Calculus analysis in [Perathoner et al., 2009; Haid and Thiele, 2007].

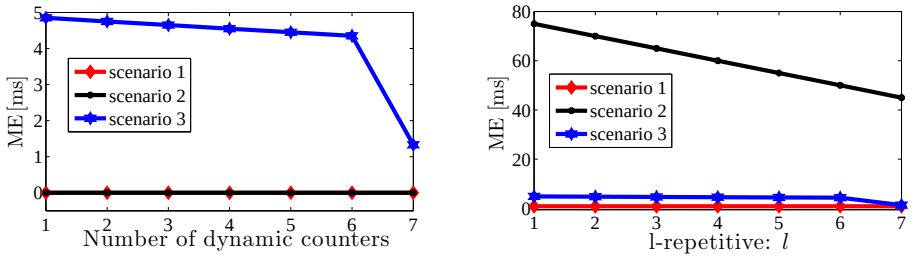
6.1. Results

The arrival curves w.r.t., the three scenarios at the two monitors are seen in Fig. 9. It can be found that, the arrival curves at the monitor M_1 are the same as the arrival curves of R_1 with a slight jitter. Since one dynamic counter is sufficient to monitor the periodic-with-jitter event stream, there will be no error of applying DCs to monitor events at M_1 in the scenarios 1, 2, as shown in Fig. 10(a). As regard to the scenario 3, the arrival curve at M_1 is the standard periodic burst pattern with a slight jitter. Both DCs and LRF methods can be applied to effectively monitor the periodic burst event stream. To use the DCs method presented in Section 4.3.1, the number of counters



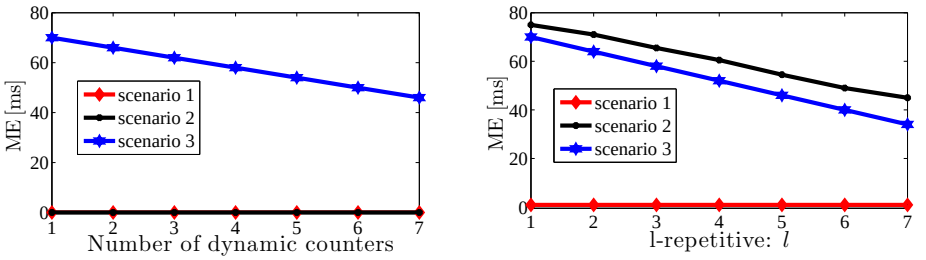
(a) Arrival curve of video frames passing from BUS to DSP (b) Arrival curve of event stream from the AND component

Fig. 9: The arrival curve that the two monitors need to guarantee



(a) DCs (b) LRF

Fig. 10: The monitoring accuracy of DCs and LRF at M_1 w.r.t., the number of counters and the number of events in the repetitive segment, respectively



(a) DCs (b) LRF

Fig. 11: The monitoring accuracy of DCs and LRF at M_2 w.r.t., the number of counters and the number of events in the repetitive segment, respectively

should be 7. However, the slight jitter of periodic burst arrival pattern is not able to be considered by using DCs and LRF methods.

The metric ME is used to evaluate the monitoring accuracy of the two methods at M_1, where Δ of ME is set to be sufficiently long. As shown in Fig. 10(a), for the scenarios 1, 2, one counter is sufficient. For the scenario 3, 7 counters are needed to increase the monitoring accuracy. It can also be seen that, one counter can guard the periodic and the periodic-with-jitter event stream. And even for scenario 3, the monitoring of using one counter is also accurate. Fig. 10(b) shows the accuracy of using LRF method at M_1. Similar to one counter, $l = 1$ can guard the periodic event stream, i.e., the staircase function. For the scenarios 1 and 3, by using a staircase function to conservatively represent the arrival curve, the ME is small. While for the scenario 2, the LRF is not so effective because the jitter of the arrival curve is large.

The arrival curves at M_2 are seen in Fig. 9(b). Although there are 3 input event streams, the arrival curves of scenario $1, 2$ are also staircase functions with jitter because periods of the 3 input event streams are the same. Hence, one counter is sufficient to monitor it. As regard to the scenario 3, the arrival curve is neither staircase function nor periodic burst function. Nevertheless, a set of staircase functions can still be used to conservatively represent it. The ME of DCs is shown in Fig. 11(a), where one counter is sufficient in the scenarios $1, 2$. The ME decreases with the increase of counters in the scenario 3. The ME of using LRF is seen in Fig. 11(b). For the scenario 1, $l = 1$ is accurate enough because the jitter is very small. For the scenario 2, since the jitter is not small, the ME is also not small. The arrival curve in the scenario 3 is not periodic burst, thus the ME of using LRF decreases with the increase of l, but cannot be 0.

6.2. Insights gained from the results

In the first two scenarios, the ME of applying the DCs method is very small in the long run, due to that the DCs method is sufficient to monitor the periodic or periodic-with-jitter event streams. The effectiveness of the LRF method on the periodic-with-jitter arrival pattern relies on the jitter size. If the jitter is large, the monitoring may not be effective. If the jitter is small, the LRF method may still be effective. Besides, for the monitoring of AND component, the output arrival curves in scenarios $1, 2$ are also periodic-with-jitter patterns because the periods of the three input event streams are the same. In the scenario 3, the arrival curve that the monitor needs to guard is irregular, the DCs or LRF method can only rely on constructing an arrival curve to represent the original arrival curve, which results in a large ME.

7. IMPLEMENTATION OVERHEAD EVALUATION

It can be found in Algo. 1 that the DCs method relies on a lot of timers and frequent timer interrupts. Because of this, it is not easy to implement this method as a software in a normal processor with a limited number of timers. To evaluate the implementation overhead of the DCs and LRF monitoring methods, both methods are implemented in FPGA because the FPGA provides numerous timers and the timer interrupt is not necessary for a hardware IP. We investigate the following three metrics of implementing the DCs and LRF methods.

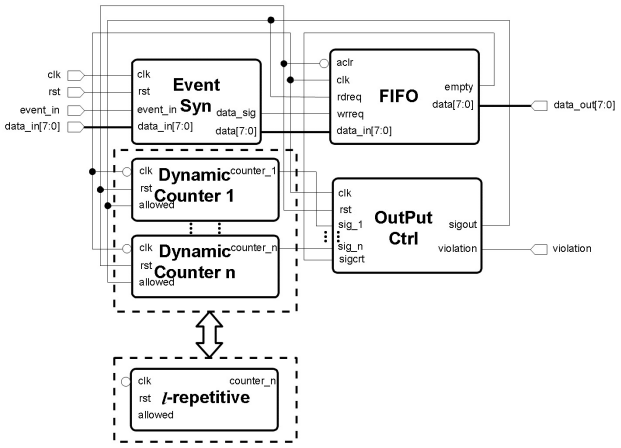


Fig. 12: Block diagram of the FPGA testbed

— Resource Overhead.
— Monitoring Latency.
— Maximum Running Frequency.

We first consider the effect of the number of used DCs and the number of segments within LRF on the three metrics. Then, we investigate the influence of our proposed methods of monitoring OR or AND component on the three metrics.

7.1. Implementation setup

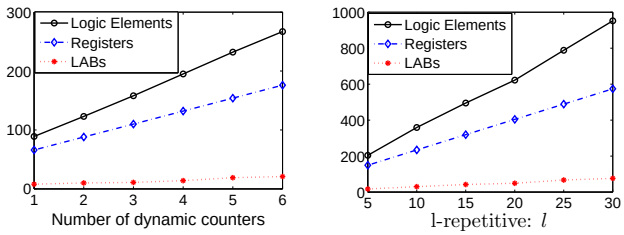
The monitor is configured as an FPGA IP module. As shown in Fig. 12, this FPGA IP module contains four submodules. The EventSyn module checks the coming event signal and data, and transmits them to FIFO. The FIFO module is used to buffer regulated events. The OutPutCtrl module controls the release of data. Dynamic Counter module or l-repetitive module is used to regulate the output of event streams. After this configuration, the correctness of the configured IP is verified in ModelSim.

7.2. The results of the DCs monitoring and the LRF monitoring

7.2.1. Resource overhead. After testing the correctness of our Verilog HDL code in ModelSim, we synthesize the implementations in Quartus using ALTERA Cyclone III EP3C120F780 device. In this test, we only concentrate on the resource overhead of each monitoring method.

For the DCs monitoring, the resource overheads mainly depend on the number of used DCs. For the LRF monitoring, the resource overheads also depend on the l within LRF. Hence, we investigate the influence of the two factors on the resource overhead. For the DCs monitoring, we choose six dynamic counters to be 1 to 6 with the increment of 1. For the l-repetitive monitoring scheme, we choose six l values. l values are 5 to 30 with increments of 5. The resource usage of the configured IP is represented by three components in the FPGA, which are logic elements, registers, and logic array blocks (LABs).

As depicted in Fig. 13, the resource usage is linear with increasing l or number of dynamic counters. Resource usage of 5-repetitive monitoring is almost the same as 4 dynamic counters, which counts less than 0.3% of the total resources (total logic elements is 119,008 in this FPGA board). This illustrates that the resource overhead of the two monitoring methods is very low.



(a) DCs monitoring (b) LRF monitoring

Fig. 13: Results of resource usage after synthesis

7.2.2. Monitoring latency. As for the monitoring timing latency, it is observed in ModelSim that 5 clock cycles ($1\ \mu s$) are needed to transfer an event from the input to the output if data is not prevented from being sent out. The timing latency is the same in both monitoring methods and keeps constant with different l or number of dynamic counters because both methods are able to be executed in parallel in FPGA hardware. This result indicates that the timing overhead of both monitoring methods is considerably small, which has little influence on the system timing behavior.

Table II: Frequency Specifications (Unit: MHz)

#counters of DCs	1	2	3	4	5	6
DCs	165.36	168.21	157.07	159.13	149.26	132.09
l of LRF	5	10	15	20	25	30
LRF	159.12	155.33	143.46	145.87	132.18	127.67

7.2.3. Maximum working frequency. The maximum working frequency of the DCs and LRF methods relies on the complexity of the arrival curve that both methods need to guard. Under the Model slow 1200mV 85C, the maximum working frequencies w.r.t.,

the needed #counters of the DCs and l of the LRF are listed in Tab. II. It can be seen that, the maximum frequency decreases when the #counters or l increases generally.

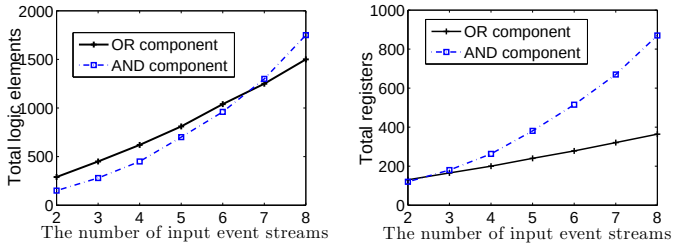
7.3. The results of monitoring methods on OR and AND components

We apply the monitoring methods on OR or AND component with FPGA and evaluate the monitoring effectiveness by inspecting the resource usage, monitoring delay, and runtime maximum frequency. Eight event streams are chosen as the basic streams to study the FPGA implementation. The eight event streams are periodic-with-jitter type with the arrival curve $\alpha_i^u = N_i^u + \lfloor \frac{\Delta + J_i}{\delta_i} \rfloor$, where the specifications of N_i^u, δ_i and J_i are shown in Tab. III.

Table III: Parameter Specifications

	α_1^u	α_2^u	α_3^u	α_4^u	α_5^u	α_6^u	α_7^u	α_8^u
N_i^u	1	3	5	7	9	11	13	15
$\delta_i[\mu s]$	20	100	180	260	340	420	500	580
J_i	10	50	90	130	170	210	250	290

7.3.1. Resource overhead. The resource results are shown in Fig. 14. As depicted in the figure, the increasing speed of the curve of monitoring output events from AND component is greater than that of monitoring output events from OR component. For the OR component, the resource overhead increases linearly with the number of input event streams. For the AND component, from Lem. 5.6 and Them. 5.8, we can derive that the needed counter number is $\frac{n(n+1)}{2}$, where n is the number of input event streams. Therefore, the needed resource usage is quadratic with the number of inputs.



(a) Total logic elements (b) Total registers

Fig. 14: Resource usage of different monitoring method IPs

7.3.2. Timing latency. It is observed that the timing latency of monitoring the OR component is $n + 7$ cycles, where n is the number of input event streams, due to that picking up CIT in Algo. 5 relies on serial execution. However, the timing latency of monitoring the AND component is still 5 cycles because there is no dependency in running the program to monitor AND component. Program runs in parallel so that 5 cycles is still enough.

7.3.3. Maximum working frequency. We then investigate the runtime maximum frequency of using this IP. From the compiling report, we get the maximum frequency on Model slow 1200mV 85C, as shown in Tab. IV. It can be found that the runtime maximum frequency is not strictly monotonous with input stream number. In general, the larger the input stream number, the smaller the maximum frequency will be.

8. CONCLUSIONS

The DCs and LRF monitoring methods are both proposed to verify the single predefined event arrival curve. The DCs method relies on the assumption that any monotone and time-invariant arrival curve can be conservatively approximated as the minimum of a set of staircase functions, while the LRF method assumes only n_{max}

Table IV: Frequency Specifications (Unit: MHz)

Number of streams	1	2	3	4	5	6	7	8
OR composition	N/A	140.45	111.06	128.21	101.52	105.08	92.44	90.84
AND composition	N/A	156.25	134.99	117.07	109.7	123.21	82.17	83.51

activated events are relevant for verification of timing constraints. Based on these assumptions, another arrival curve is constructed to represent the original pre-defined arrival curve. The DCs and LRF methods will verify whether or not the coming events comply with the constructed arrival curve.

Since the coming events are verified based on the constructed arrival curve, the monitoring effectiveness depends on two factors: the difference between original pre-defined arrival curve and the constructed arrival curve, and the counter number for DCs method or l for LRF method. The ME is defined to represent the difference between original pre-defined arrival curve and the constructed arrival curve. ME =0 means that the monitoring method is sufficient because the arrival curve that the monitoring method guards is exactly the same as the pre-defined arrival curve.

In this article, the event arrival patterns are categorized into three types, which are, periodic, periodic events with an initial burst, and periodic burst. Besides, the monitoring methods in verifying the output events from the OR and AND components are explored. Towards different event streams, we now have the following conclusions.

— Periodic event stream: both DCs and LRF methods are sufficient. Only one counter or $l = 1$ is needed.
— Periodic event stream with an initial burst: if this burst is only a jitter of a periodic event stream, only one counter is needed. If not, the counter number depends on how many staircase functions that are needed for representing the arrival curve. The effectiveness of the LRF method depends on the burst. If the burst is small, the LRF will be effective by setting $l = 1$ to verify the arriving events as the periodic events. If not, the LRF may not be effective.
— Periodic burst event stream: for the standard periodic burst event stream, the DCs method is sufficient. The needed counters is $b + 1$, where b is defined in Eq. 11. The LRF method is sufficient, where $l = b + 1$. For the nonstandard periodic burst event stream, the DCs method is not sufficient.
— OR component: if the event streams that come into the OR component are periodic-with-jitter arrival pattern, the DCs method is sufficient. The needed counter number is the same as the number of input event streams. The LRF method is not discussed.
— AND component: if the event streams that come into the AND component are periodic-with-jitter arrival pattern, the DCs method is sufficient. If all periods are the same, the needed counter number may be only 1. If no periods are the same, the needed counter number is $\frac{n(n+1)}{2}$, where n is the number of input event streams. The LRF method is also not discussed.

Furthermore, the MPEG-2 decoder is studied to present the monitoring effectiveness of using DCs and LRF methods. In this case, three scenarios are considered, where the DCs and LRF methods are shown to be effective in some specific scenarios. By implementing both monitoring methods in FPGA, we show that the resource usage of implementing the DCs and LRF methods is quite low, and the monitoring latency is negligible. The implementation of monitoring methods on OR and AND components are also investigated. The results show that, the monitoring latency on OR component is linear with the number of input event streams and the monitoring latency of AND component is still 5 cycles. It is also found that the maximum frequency can be over 100 MHz if there are less than 7 input event streams under Model slow 1200mV 85C.

9. APPENDIX

THEOREM 9.1. *Assume $DC_i(t)$ and $CLK_i(t)$ are the values of counters and timers at the time t in monitoring output events with OR component. If at the time t_0, an output event is checked, where t_0^- and t_0^+ denote the time before and after the monitoring trigger. The responsible counter is the counter with the smallest Δ', where Δ' satisfies*

$$\Big\lfloor \frac{\Delta' + \delta_i - CLK_i(t_0^+)}{\delta_i} \Big\rfloor = N_i^u + 1 - DC_i(t_0^-) \tag{18}$$

PROOF. Assume DC_i is the responsible counter. Then, at the time $t_0^+ + \Delta$, the following equation holds,

$$\begin{aligned} &DC_i(t_0^+ + \Delta) = DC_i(t_0^+) + q_i(\Delta, t_0^+),\ where \\ &q_i(\Delta, t_0^+) = \min\Big(N_i^u - DC_i(t_0^+), \Big\lfloor \frac{\Delta + \delta_i - CLK_i(t_0^+)}{\delta_i} \Big\rfloor\Big) \end{aligned} \tag{19}$$

The system capacity at the time $t_0^+ + \Delta$ is:

$$capacity(t_0^+ + \Delta) = \sum_{k=1}^{n} DC_k(t_0^+) + \sum_{k=1}^{n} q_k(\Delta, t_0^+). \tag{20}$$

$\sum_{k=1}^{n} DC_k(t_0^+) = \sum_{k=1}^{n} DC_k(t_0^-) - 1$, no matter which counter is responsible for the event. The system capacity at the time $t_0^+ + \Delta$ is only decided by $\sum_{k=1}^{n} q_k(\Delta, t_0^+)$.

For DC_i as the responsible counter,

$$\begin{aligned} \sum_{k=1}^{n} q_k^i(\Delta, t_0^+) = &\sum_{k=1, k\neq i,j}^{n} q_k(\Delta, t_0^+) + \min\Big(N_i^u + 1 - DC_i(t_0^-), \Big\lfloor \frac{\Delta + \delta_i - CLK_i(t_0^+)}{\delta_i} \Big\rfloor\Big) + \\ &\min\Big(N_j^u - DC_j(t_0^-), \Big\lfloor \frac{\Delta + \delta_j - CLK_j(t_0^+)}{\delta_j} \Big\rfloor\Big) \end{aligned} \tag{21}$$

where $DC_i(t_0^+) = DC_i(t_0^-) - 1$, $DC_j(t_0^+) = DC_j(t_0^-)$.

If a different counter DC_j is chosen as the responsible counter,

$$\begin{aligned} \sum_{k=1}^{n} q_k^j(\Delta, t_0^+) = &\sum_{k=1, k\neq i,j}^{n} q_k(\Delta, t_0^+) + \min\Big(N_i^u - DC_i(t_0^-), \Big\lfloor \frac{\Delta + \delta_i - CLK_i(t_0^+)}{\delta_i} \Big\rfloor\Big) + \\ &\min\Big(N_j^u + 1 - DC_j(t_0^-), \Big\lfloor \frac{\Delta + \delta_j - CLK_j(t_0^+)}{\delta_j} \Big\rfloor\Big) \end{aligned} \tag{22}$$

where $DC_i(t_0^+) = DC_i(t_0^-)$, $DC_j(t_0^+) = DC_j(t_0^-) - 1$.

Set Δ_i and Δ_j satisfy that:

$$\Big\lfloor \frac{\Delta_i + \delta_i - CLK_i(t_0^+)}{\delta_i} \Big\rfloor = N_i^u + 1 - DC_i(t_0^-), \Big\lfloor \frac{\Delta_j + \delta_j - CLK_j(t_0^+)}{\delta_j} \Big\rfloor = N_j^u + 1 - DC_j(t_0^-). \tag{23}$$

For $\Delta < \Delta_i$,

$$\min\Big(N_i^u + 1 - DC_i(t_0^-), \Big\lfloor \frac{\Delta + \delta_i - CLK_i(t_0^+)}{\delta_i} \Big\rfloor\Big) = \min\Big(N_i^u - DC_i(t_0^-), \Big\lfloor \frac{\Delta + \delta_i - CLK_i(t_0^+)}{\delta_i} \Big\rfloor\Big). \tag{24}$$

For $\Delta < \Delta_j$,

$$\min\Big(N_j^u + 1 - DC_j(t_0^-), \Big\lfloor \frac{\Delta + \delta_j - CLK_j(t_0^+)}{\delta_j} \Big\rfloor\Big) = \min\Big(N_j^u - DC_j(t_0^-), \Big\lfloor \frac{\Delta + \delta_j - CLK_j(t_0^+)}{\delta_j} \Big\rfloor\Big). \tag{25}$$

Then, if $\Delta_i \le \Delta_j$, for $\Delta \le \Delta_i \le \Delta_j$, it can be deduced that

$$\sum_{k=1}^{n} q_k^i(\Delta, t_0^+) \ge \sum_{k=1}^{n} q_k^j(\Delta, t_0^+). \tag{26}$$

DC_i should be chosen as the responsible counter. If $\Delta_j \le \Delta_i$, for $\Delta \le \Delta_j \le \Delta_i$, it can be deducted that

$$\sum_{k=1}^{n} q_k^j(\Delta, t_0^+) \ge \sum_{k=1}^{n} q_k^i(\Delta, t_0^+). \tag{27}$$

DC_j should be chosen as the responsible counter.

Then, for any counters, the responsible counter is the counter with the smallest Δ', where Δ' satisfy the Eq. 15. □

LEMMA 9.2. *For two periodic event streams with jitter (e.g., $\alpha_1 = [\alpha_1^u, \alpha_1^l]$, $\alpha_2 = [\alpha_2^u, \alpha_2^l]$, where $\alpha_1^u = \lceil \frac{\Delta+J_1}{\delta_1} \rceil$, $\alpha_1^l = \max(0, \lfloor \frac{\Delta-J_1}{\delta_1} \rfloor)$, $\alpha_2^u = \lceil \frac{\Delta+J_2}{\delta_2} \rceil$, $\alpha_2^l = \max(0, \lfloor \frac{\Delta_2-J_2}{\delta_2} \rfloor)$, $(\delta_1 < \delta_2)$, then $\forall\Delta$, $\alpha_1^u \oslash \alpha_2^l(\Delta) = +\infty$, and $\alpha_2^u \oslash \alpha_1^l(\Delta) = \alpha_2^u(\Delta + \delta_1 + J_1)$.*

PROOF. First, we prove $\alpha_1^u \oslash \alpha_2^l$ is an infinite to any Δ.

$$\begin{aligned}\alpha_1^u \oslash \alpha_2^l(\Delta) &= \sup_{\lambda\ge 0}\{\alpha_1^u(\Delta+\lambda) - \alpha_2^l(\lambda)\} = \sup_{\lambda\ge 0}\Big\{\Big\lceil \frac{\Delta+\lambda+J_1}{\delta_1}\Big\rceil - \max\Big(0, \Big\lfloor \frac{\lambda - J_2}{\delta_2}\Big\rfloor\Big)\Big\} \\ &\ge \sup_{\lambda\ge 0}\Big\{\Big\lfloor \frac{\lambda}{\delta_1}\Big\rfloor - \Big\lfloor \frac{\lambda}{\delta_2}\Big\rfloor - \Big\lceil \frac{J_2}{\delta_2}\Big\rceil\Big\}.\end{aligned}$$

Since $\delta_1 < \delta_2$, we can set $\delta_1 = c * \delta_2,\ (0 < c < 1)$. Therefore,

$$\sup_{\lambda\ge 0}\Big\{\Big\lfloor \frac{\lambda}{\delta_1}\Big\rfloor - \Big\lfloor \frac{\lambda}{\delta_2}\Big\rfloor\Big\} \ge \sup_{\lambda\ge 0}\Big\{\Big(\frac{1}{c} - 1\Big)\frac{\lambda}{\delta_2} - 1\Big\},$$

where $(\frac{1}{c} - 1)\frac{\lambda}{\delta_2}$ is a wide increasing function. So $\sup_{\lambda\ge 0}\{\lfloor \frac{\lambda}{\delta_1} \rfloor - \lfloor \frac{\lambda}{\delta_2} \rfloor\}$ is infinity, and $\alpha_1^u \oslash \alpha_2^l(\Delta)$ is also infinity.

Now we prove $\alpha_2^u \oslash \alpha_1^l$ is left shifted $\delta_1 + J_1$ of α_2^u.

$$\alpha_2^u \oslash \alpha_1^l(\Delta) = \sup_{\lambda\ge 0}\{\alpha_2^u(\Delta+\lambda) - \alpha_1^l(\lambda)\} = \sup_{\lambda\ge 0}\Big\{\Big\lceil \frac{\Delta+\lambda+J_2}{\delta_2}\Big\rceil - \max\Big(0, \Big\lfloor \frac{\lambda - J_1}{\delta_1}\Big\rfloor\Big)\Big\}.$$

Consider $0 \le \lambda < \delta_1 + J_1$, then $\max(0, \lfloor \frac{\lambda - J_1}{\delta_1} \rfloor) = 0$. So

$$\alpha_2^u \oslash \alpha_1^l(\Delta) = \sup_{0\le\lambda<\delta_1+J_1}\{\alpha_2^u(\Delta+\lambda) - \alpha_1^l(\lambda)\} = \sup_{0\le\lambda<\delta_1+J_1}\Big\{\Big\lfloor \frac{\Delta+\lambda+J_2}{\delta_2}\Big\rfloor\Big\} = \Big\lceil \frac{\Delta+J_2}{\delta_2} + \frac{\delta_1+J_1}{\delta_2}\Big\rceil.$$

For $\lambda \ge \delta_1 + J_1$, $\max(0, \lfloor \frac{\lambda - J_1}{\delta_1} \rfloor) = \lfloor \frac{\lambda - J_1}{\delta_1} \rfloor$, and $\lceil \frac{\Delta+\lambda+J_2}{\delta_2} \rceil - \lfloor \frac{\lambda - J_1}{\delta_1} \rfloor = \lfloor \frac{\Delta+\lambda+J_2+\delta_2}{\delta_2} \rfloor - \lfloor \frac{\lambda - J_1}{\delta_1} \rfloor$ is a decreasing function because $\delta_1 < \delta_2$. So

$$\alpha_2^u \oslash \alpha_1^l(\Delta) = \sup_{\lambda\ge\delta_1+J_1}\Big\{\Big\lceil \frac{\Delta+\lambda+J_2}{\delta_2}\Big\rceil - \Big\lfloor \frac{\lambda - J_1}{\delta_1}\Big\rfloor\Big\} = \Big\lceil \frac{\Delta+J_2}{\delta_2} + \frac{\delta_1+J_1}{\delta_2}\Big\rceil.$$

Therefore, $\alpha_2^u \oslash \alpha_1^l = \lceil \frac{\Delta+J_2}{\delta_2} + \frac{\delta_1+J_1}{\delta_2} \rceil$. □

References

Bonakdarpour, B., Navabpour, S., and Fischmeister, S. (2011). Sampling-based runtime verification. In *FM 2011: Formal Methods*, pages 88–102. Springer.

Bonakdarpour, B., Navabpour, S., and Fischmeister, S. (2013). Time-triggered runtime verification. volume 43, pages 29–60. Springer.

Haid, W. and Thiele, L. (2007). Complex task activation schemes in system level performance analysis. In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 173–178. IEEE.

Henia, R., Hamann, A., Jersak, M., Racu, R., Richter, K., and Ernst, R. (2005). System level performance analysis - the symta/s approach. In *Computers and Digital Techniques*, pages 148–166. IEEE.

Huang, K., Chen, G., Buckl, C., and Knoll, A. (2012). Conforming the runtime inputs for hard real-time embedded systems. In *Design Automation Conference (DAC)*, pages 430–436. ACM.

Jersak, M. (2005). *Compositional performance analysis for complex embedded applications.* PhD thesis, University of Braunschweig-Institute of Technology.

Jersak, M. and Ernst, R. (2003). Enabling scheduling analysis of heterogeneous systems with multi-rate data dependencies and rate intervals. In *Design Automation Conference (DAC)*, pages 454–459. ACM.

Lampka, K., Huang, K., and Chen, J.-J. (2011). Dynamic counters and the efficient and effective online power management of embedded real-time systems. In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 267–276. ACM.

Lampka, K., Perathoner, S., and Thiele, L. (2009). Analytic real-time analysis and timed automata: A hybrid method for analyzing embedded real-time systems. In *ACM International Conference on Embedded Software (EMSOFT)*, pages 107–116. ACM.

Lampka, K., Perathoner, S., and Thiele, L. (2010). Analytic real-time analysis and timed automata: A hybrid methodology for the performance analysis of embedded real-time systems. volume 14, pages 193–227. Springer.

Le Boudec, J.-Y. and Thiran, P. (2001). *Network calculus: a theory of deterministic queuing systems for the internet*, volume 2050. Springer.

Medhat, R., Bonakdarpour, B., Kumar, D., and Fischmeister, S. (2015). Runtime monitoring of cyber-physical systems under timing and memory constraints. pages 14:1–14:29. ACM.

Medhat, R., Kumar, D., Bonakdarpour, B., and Fischmeister, S. (2014). Sacrificing a little space can significantly improve monitoring of time-sensitive cyber-physical systems. In *Cyber-Physical Systems (ICCPS)*, pages 115–126. IEEE.

Neukirchner, M., Axer, P., Michaels, T., and Ernst, R. (2013a). Monitoring of workload arrival functions for mixed-criticality systems. In *Real-Time Systems Symposium (RTSS)*, pages 88–96. IEEE.

Neukirchner, M., Michaels, T., Axer, P., Quinton, S., and Ernst, R. (2012). Monitoring arbitrary activation patterns in real-time systems. In *Real-Time Systems Symposium (RTSS)*, pages 293–302. IEEE.

Neukirchner, M., Quinton, S., Ernst, R., and Lampka, K. (2013b). Multi-mode monitoring for mixed-criticality real-time systems. In *Hardware/Software Codesign and System Synthesis (CODES-ISSS)*, pages 1–10. ACM.

Perathoner, S., Wandeler, E., Thiele, L., Hamann, A., Schliecker, S., Henia, R., Racu, R., Ernst, R., and GonzÃąlez Harbour, M. (2009). Influence of different abstractions on the performance analysis of distributed hard real-time systems. pages 27–49. Springer.

Phan, L. and Lee, I. (2013). Improving schedulability of fixed-priority real-time systems using shapers. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 217–226. IEEE.

Richter, K., Jersak, M., and Ernst, R. (2003a). A formal approach to mpsoc performance verification. volume 36, pages 60–67. IEEE.

Richter, K., Racu, R., and Ernst, R. (2003b). Scheduling analysis integration for heterogeneous multiprocessor soc. In *Real-Time Systems Symposium (RTSS)*, pages 236–245. IEEE.

Thiele, L., Chakraborty, S., and Naedele, M. (2000). Real-time calculus for scheduling hard real-time systems. In *International Symposium on Circuits and Systems*, pages 101–104. IEEE.

Tindell, K. W., Burns, A., and Wellings, A. J. (1994). An extendible approach for analyzing fixed priority hard real-time tasks. pages 133–151. Springer.

Wandeler, E. (2006). *Modular performance analysis and interface-based design for embedded real-time systems*. PhD thesis, ETH Zurich, Swiss.

Wandeler, E., Maxiaguine, A., and Thiele, L. (2012). On the use of greedy shapers in real-time embedded systems. pages 1–22. ACM.