Improved Schedulability Analysis of EDF Scheduling on Reconfigurable Hardware Devices*

Nan Guan[1], Zonghua Gu[2], Qingxu Deng[1], Weichen Liu[2] and Ge Yu[1]
[1]Dept of Computer Science and Engineering
Northeastern University
Shenyang, China
[2]Dept of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China

Abstract

Reconfigurable devices, such as Field Programmable Gate Arrays (FPGAs), are very popular in today's embedded systems design due to their low-cost, high-performance and flexibility. Partially Runtime-Reconfigurable (PRTR) FPGAs allow hardware tasks to be placed and removed dynamically at runtime. Hardware task scheduling on PRTR FPGAs brings many challenging issues to traditional real-time scheduling theory, which have not been adequately addressed by the research community compared to software task scheduling on CPUs. In this paper, we consider the schedulability analysis problem of HW task scheduling on PRPR FPGAs. We derive utilization bound tests for two variants of global EDF scheduling, and use synthetic tasksets to compare performance of the tests to existing work and simulation results.

1 Introduction and Background

A reconfigurable device, such as a FPGA, consists of a rectangular grid of Configurable Logic Blocks (CLBs) and the interconnects between them. FPGAs are inherently parallel, that is, two or more HW tasks can execute on a FPGA concurrently as long as they can both fit on it. FPGAs can be *1D reconfigurable*, where each task occupies a contiguous set of columns, or *2D reconfigurable*, where each task occupies a rectangular area. Partially Runtime-Reconfigurable (PRTR) FPGAs allow part of the FPGA area to be reconfigured while the remainder continues to operate without interruption. In other words, HW tasks can be placed and removed dynamically at runtime. In this paper, we use the word *FPGA* to refer to *PRTR FPGA*.

HW task scheduling on FPGAs shares many similarities with global SW task scheduling on identical multiprocessors [1], where all processors in the system have identical processing speed and different task invocation instances may run on different processors. But it is actually a more general and challenging problem since a HW task may occupy a different area size on the FPGA while a SW task always occupies one and only one CPU. In fact, we can view multiprocessor scheduling as a special case of task scheduling on 1D reconfigurable FPGAs where all tasks have width equal to 1.

For multiprocessor and FPGA scheduling, we cannot rely on worst-case response time calculation for schedulability analysis like [2], since there may not be a *critical instant*, i.e., it is generally unknown what task release phase offsets will cause any task to achieve its worst-case response time. Therefore, we are forced to rely on utilization bound tests for schedulability analysis. For EDF-based multiprocessor scheduling, several authors have presented utilization bound tests. Goossens et al [3] presented a utilization bound test, referred to as **GFB**, assuming that tasks have relative deadlines equal to the period. Baker [4] presented another one, referred to as **BAK1**, that can handle relative deadlines less than or equal to the period. Baker [5] extended **BAK1** to include tasks with post-period deadlines. Bertogna et al [6] presented an improved test, referred to as **BCL**, and showed that **GFB** and **BAK1** are incomparable to each other, and each test can accept tasksets that the other test rejects. For tasksets with different timing characteristics, they have different performance in terms of acceptance ratio. Baker [7] further showed that all three tests, **GFB**, **BCL** and **BAK1**, are generally incomparable to each other. **GFB** performs better than **BCL** if the taskset only consists of tasks with low time utilization (time-light tasks), while **BCL** performs better if there are tasks with high time utilization (time-heavy tasks) [4]. Baker [8] further improved upon **BCL**, and presented another utilization bound test, referred to as **BAK2**, which combines **BCL** with the busy-interval analysis of **BAK1** to obtain a tighter bound than either method could achieve alone.

For EDF-based global scheduling on FPGAs, Danne et al [9] presented a schedulability bound test (referred to as **DP**) based on Goossens et al [3] (**GFB**). Danne et al [10] also discussed partitioned scheduling for FPGAs, where each task is restricted to executing on a given partition of the FPGA, and task execution on each partition is serialized, so the problem is reduced to task allocation followed by single-processor schedulability analysis. We focus on global EDF scheduling in this paper.

*this work is partially supported by National Basic Research Program of China (973 Program) under Grant No.2006CB303000 and the Cultivation Fund of the Key Scientific and Technical Innovation Project, Ministry of Education of China, under Grant No.706016.

There are two possible variants of global EDF scheduling for FPGAs, as discussed in [9]:

Definition 1 *(**EDF-FkF**) Let Q be the queue of all active jobs sorted by non-decreasing deadlines (sorted by release time in ties of deadlines). Let J_i denote the ith job in Q. The scheduling algorithm* EDF-First-k-Fit *(EDF-FkF) selects at any time the first k jobs R of Q for execution, with the largest k for which $\sum_{J_i \in R} A_i \leq A(H)$ holds.*

Definition 2 *(**EDF-NF**) Let Q be the queue of all active jobs sorted by non-decreasing deadlines (sorted by release time in ties of deadlines). Let J_i denote the ith job in Q. The scheduling algorithm* EDF-Next-Fit *(EDF-NF) determines the set of running tasks R with the following algorithm: start with an empty set R and visit all active jobs $J_i \in Q$ in order of non-decreasing deadlines. Add J_i to R if and only if $\sum_{J_k \in R \cup J_i} A_k \leq A(H)$.*

Perhaps EDF-NF is a misnomer because EDF-NF does not allocate and schedule tasks strictly in deadline order. Danne et al [9] showed that EDF-NF is superior to EDF-FkF in the sense that if a taskset Γ is schedulable using EDF-FkF, then it is also schedulable using EDF-NF. Intuitively, the reason is that EDF-FkF may leave some hardware resources idle, if there are ready jobs that can fit on the FPGA but are blocked by a job J_k with a shorter deadline but cannot fit on the FPGA, but EDF-NF can exploit these idle resource by skipping J_k and place the jobs behind it in the queue first.

We derive two schedulability bound tests for EDF-NF and EDF-FkF in Sections 4 and 5, respectively. Since EDF-NF is superior to EDF-FkF, both tests are also applicable to EDF-NF. Since the FPGA scheduling problem is the generalized case of multiprocessor CPU scheduling problem, the derivation processes of FPGA schedulability bounds are based on those of multiprocessor schedulability bounds. Specifically, Danne et al [9] (**DP**) for EDF-FkF is based on Goossens et al [3] (**GFB**); Theorem 2 (**GN1**) in Section 4 for EDF-NF is based on Bertogna et al [6] (**BCL**); Theorem 3 (**GN2**) in Section 5 for EDF-NkF is based on Baker [8] (**BAK2**). The derivation processes of Theorems 2 and 3 closely follow the respective references (Bertogna et al [6] and Baker [8]), with consideration of the FPGA-specific issue of task area size.

We make the following assumptions:

- The FPGA is 1D reconfigurable, and each job occupies a contiguous set of columns.
- The entire FPGA area is homogeneous without any pre-configured cells or columns. In reality, some parts of the FPGA may be pre-configured as memory blocks or soft-core CPUs, hence are not available for task placement. If we take this factor into account, then any schedulability bound test will not be generally applicable anymore and become less interesting.
- Reconfiguration overhead on the FPGA is zero. In reality, FPGA reconfiguration carries a significant overhead in the range of milliseconds that is proportional to the size of area reconfigured, unlike CPU scheduling, where task context-switch overhead is typically small enough to be ignored. We make this assumption to simplify the theorem derivation process, but it is easy to take into account the overhead by adding it to the execution time, similar to response time analysis in fixed-priority CPU scheduling [2].
- We allow *unrestricted migration* [1] of jobs, i.e., a job can be preempted and migrated to another position of the FPGA with zero overhead. In other words, the FPGA can be defragmented by rearranging active jobs to different positions on the FPGA with zero overhead. As a result, a job can fit on the FPGA as long as its area size is less than the remaining free area size. If we did not permit task migration, then we would need to adopt a task placement strategy such as first-fit, best-fit or worst-fit, and only accept a task only when there is enough contiguous free space on the FPGA. We leave this issue as part of our future work. (This issue does not arise in multiprocessor CPU scheduling since each SW task occupies one and only one CPU.)
- We only consider priority-based scheduling, but not optimal multiprocessor scheduling algorithms such as Pfair [11], which involves frequent task context-switches and is not likely to be suitable for HW task scheduling on FPGAs due to the large reconfiguration overhead. (Even though we assume zero reconfiguration overhead in this paper, our goal is to construct an analysis framework that can be easily extended to handle large reconfiguration overhead without much difficulty.)

This paper is structured as follows: Section 2 presents the problem definition and terminology. Section 3 discusses the work-conserving concept on FPGAs, which forms the foundation for the theorem derivation in later parts of this paper. Section 4 presents the schedulability bound test for EDF-NF, and section 5 presents the schedulability bound test for EDF-FkF. Section 6 presents experimental evaluation results, and Section 7 discusses conclusions and possible future work.

2 Problem Definition and Terminology

We consider a 1D reconfigurable FPGA H with $A(H)$ columns, also referred to as its *area size*. We consider a taskset Γ consisting of N periodic or sporadic tasks. Each task $\tau_k = (C_k, D_k, T_k, A_k), k \in 1, \ldots, N$ is characterized by its execution time C_k, period or minimum inter-arrival time T_k , a relative deadline D_k and an area size A_k, which represents the amount of contiguous columns that τ_k occupies. A task τ_k consists of a sequence of jobs J_k^j.

For multiprocessor CPU scheduling, each task occupies one processor when it is executing, so we evaluate the work done by a job only based on its execution time. But for FPGA scheduling, each task can occupy a different area size, so we must evaluate the work done by a job based on its area size in addition to its execution time. We define two notions of *work*

done by a task. The *time work* $W_i^T(t-\delta,t)$ done by task τ_i over a time interval $[t-\delta,t)$ is the sum of the lengths of all subintervals during which a job J_i^j executes. The *system work* $W_i^S(t-\delta,t)$ done by task τ_i over a time interval $[t-\delta,t)$ is the product of the time work of the interval and the area of the task: $W_i^S(t-\delta,t) = W_i^T(t-\delta,t) * A_i$ The time utilization of a taskset Γ is $U^T(\Gamma) = \sum_{i=1}^{N} C_i/T_i$, and system utilization is $U^S(\Gamma) = \sum_{i=1}^{N} C_i * A_i/T_i$.

The *interference* $I_k(t-\delta,t)$ of a task τ_k over a time interval $[t-\delta,t)$ is the sum of the lengths of all the sub-intervals in $[t-\delta,t)$ during which τ_k is preempted. The *interference contribution* $I_{i,k}(t-\delta,t)$ *of a task* τ_i to $I_k(t-\delta,t)$ is the amount of interference caused by τ_i to τ_k. The *block busy interval* is any time interval during which the idle area of the FPGA is no larger than $A(H) - A_{max} + 1$, where A_{max} is largest area size of any task in Γ. The *block busy time* $B(t-\delta,t)$ of a time interval $[t-\delta,t)$ is the sum of the length of all block busy intervals in $[t-\delta,t)$. The *block busy time* $B_i(t-\delta,t)$ *of* τ_i is the total amount of time during which τ_i is executing in the block busy time $B(t-\delta,t)$. The τ_k*-busy interval* is the interval during which τ_k always has active jobs executing or waiting to execute.

3 The Work Conserving Concept for FPGA

If a multiprocessor CPU scheduling algorithm is *work-conserving*, it means that it never leaves any processor idle when there are any jobs in the ready queue. For example, global EDF scheduling is *work-conserving* on identical multiprocessor systems [3]. This fact is the basis for derivation of the utilization bound tests of global EDF scheduling.

Unlike multiprocessor CPU scheduling, it is possible for parts of the FPGA area to be idle when there are jobs in the ready queue, because the idle area may not be large enough to fit any of the jobs in the ready queue. Therefore, we need an extended notion of work-conserving algorithms. Danne et al [9] defined the concept of α*-work-conserving* for FPGA scheduling, which guarantee that at least $\alpha * A(H)$ area of the FPGA is occupied (busy) when there are jobs in the ready queue. Next, we present two definitions of α-work-conserving algorithms for EDF-FkF and EDF-NF.

Definition 3 *A scheduling algorithm is* global-α-work-conserving *if at least* $\alpha * A(H)$ *area of the FPGA* H *with total area* $A(H)$ *is occupied whenever there are jobs in the ready queue.*

Lemma 1 *EDF-FkF is a* global-α-work-conserving *algorithm with*

$$\alpha = 1 - (A_{max} - 1)/A(H)$$

where A_{max} *is the largest area of any possible job.*

Our definition of global-α-work-conserving is the same as the α work-conserving concept in Danne et al [9], in which a task τ_i's area A_i is assumed to be a real number instead of an integer for the purpose of generality, and α is determined to be $1 - A_{max}/A(H)$. We believe it is more reasonable to assume A_i is an integer, since it refers to the number of columns that τ_i occupies. In this case, α should be $1-(A_{max}-1)/A(H)$. Intuitively, if an area of size A_{max} is idle, then it is still possible to fit another task on the FPGA; but if an area of size $(A_{max}-1)$ is idle, then it may not be possible to fit another task. Therefore, in an overload situation, i.e., when the task queue is not empty, at least $A(H) - (A_{max} - 1)$ area of the FPGA must be occupied, hence $\alpha = 1 - (A_{max} - 1)/A(H)$. Fig. 1(a) illustrates this point graphically. With the integer task area assumption, the **EDF-FkF schedulability condition** in [9] should be modified as follows, which is the formula used in Section 6 for performance evaluation:

Theorem 1 ***(DP)*** *Any periodic taskset* Γ *can be feasibly scheduled by EDF-FkF on a FPGA* H *with area* $A(H) \geq A_{max}$ *if:*

$$\forall \tau_k \in \Gamma : U^S(\Gamma) \leq (A(H) - A_{max} + 1) * (1 - U^T(\tau_k)) + U^S(\tau_k)$$

where A_{max} *is the largest area of any task in* Γ.

Next, we define another notion of work-conserving algorithm in order to obtain tighter schedulability bounds:

Definition 4 *A scheduling algorithm is* interval-α-work-conserving *if at least* $\alpha * A(H)$ *area of the FPGA* H *with total area* $A(H)$ *is occupied during a time interval* $[a,b]$.

A *global-*α*-work-conserving* algorithm guarantees a lower bound of system utilization whenever there are jobs in the ready queue, but an *interval-*α*-work-conserving* algorithm only guarantees a lower bound of system utilization during certain time intervals. We define this concept in order to get a tighter α bound for EDF-NF:

Lemma 2 *EDF-NF is an* interval-α-work-conserving *algorithm with*

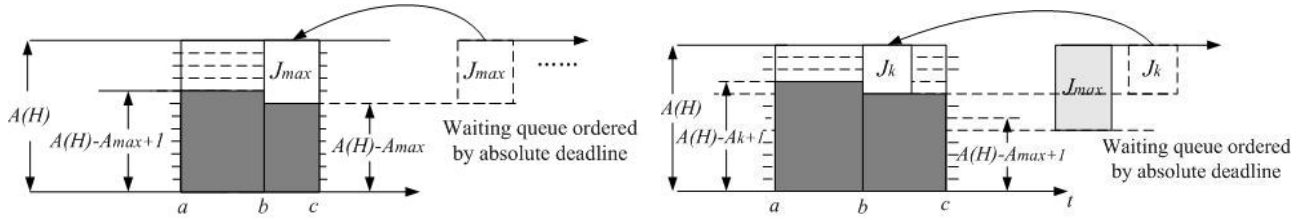
$$\alpha = 1 - (A_k - 1)/A(H)$$

in any time interval during which the job J_k *is in the ready queue.*

For EDF-NF, when a job cannot fit on the idle area of the FPGA, we can first allocate some other job with a longer deadline and smaller area that can fit on the FPGA. Therefore, at least $(A(H) - (A_k - 1))$ area of the FPGA must be occupied when a job J_k with area A_k is in the ready queue. But for EDF-FkF, we must allocate jobs in the order of non-decreasing deadlines, therefore a job with a large area can block other jobs behind it in the wait queue from being allocated, so we must be pessimistic and use A_{max} instead of A_k. Fig. 1(b) illustrates the situation for EDF-NF.

4 Schedulability Bound Test for EDF-NF

In this section, we derive a schedulability bound test based on Bertogna et al [6] (referred to as **BCL**). The **BCL** test is



(a) Global-α-work-conserving for EDF-FkF.

(b) Interval-α-work-conserving for EDF-NF.

Figure 1. The work-conserving concept for FPGA scheduling.

derived by analyzing the upper bound of the interference time of a given task during its execution. If the interference that a task τ_i can impose on task τ_k in the time interval $[r_k^j, d_k^j)$ is greater than $D_k - C_k$, then it is sufficient to only consider the portion $D_k - C_k$ in the response time calculation of task τ_k. The **BCL** bound is tighter than Goossens et al [3], which assumes τ_k can be preempted by all the work done by other tasks.

Here are the key steps of the derivation: suppose a job J_k^j of task τ_k misses its deadline d_k^j, then we can find the lower bound of the interference I_k that the job must suffer in interval $[r_k^j, d_k^j)$ to cause the deadline miss. Since the precise interference in any interval is hard to calculate, we derive an upper bound to the interference using the workload in the interval, and then derive a sufficient schedulability bound test.

For a job to meet its deadline, the total interference on the task must not be larger than its slack $D_k - C_k$. We define the worst-case interference for task τ_k as: $I_k^* = \max_j(I_k(r_k^j, d_k^j)) = I_k(r_k^{j*}, d_k^{j*})$, where $j*$ is the job instance in which the total interference is maximal. We also define $I_{i,k}^* = I_{i,k}(r_k^{j*}, d_k^{j*})$.

Lemma 3 *The taskset is schedulable if the following condition is true for each τ_k:*

$$\sum_{i \neq k} A_i \min(I_{i,k}^*, D_k - C_k) \leq (A(H) - A_k + 1)(D_k - C_k).$$

Proof 1 *The proof is by contradiction. Suppose the taskset is not schedulable, then there must exists a job J_k^j that misses it deadline at time t. The total interference on the task $I_k(t - \delta, t) > D_k - C_k$.*

Let $x = D_k - C_k$, $\xi = \sum_{i:I_{i,k} \geq x} A_i$, $A_{bnd} = (A(H) - A_k + 1)$, $\varpi(i,x) = \sum_i A_i \min(I_{i,k}(t-\delta, t), x)$, then $\varpi(i,x) = \xi x + \sum_{i:I_{i,k} < x} A_i I_{i,k}(t - \delta, t)$

If $\xi > A_{bnd}$, obviously $\varpi(i,x) > (A(H) - A_k + 1)(D_k - C_k)$.

If $\xi \leq A_{bnd}$. The system work done by all the tasks in $\tau_k's$ interference interval of $[t-\delta, t)$ is $\sum_{i=1}^N A_i * I_{i,k}(t - \delta, t)$. From Lemma 2, we know that for EDF-NF, the occupied area cannot be less than $A(H) - A_k + 1$ in any given time point when τ_k is in the ready queue. Therefore, the system work done by all the tasks in $\tau_k's$ interference is no less than $(A(H) - A_k + 1) I_k(t - \delta, t)$. So we have

$$\varpi(i,x) > \xi x + A_{bnd} I_k(t-\delta, t) - \sum_{i:I_{i,k} \geq x} A_i I_{i,k}(t - \delta, t)$$

Because $I_k(t-\delta, t) > I_{i,k}(t-\delta, t)$, then $\varpi(i,x) > \xi x + A_{bnd} I_k(t-\delta, t) - \sum_{i:I_{i,k} \geq x} A_i I_k(t - \delta, t)$
$\varpi(i,x) > \xi x + A_{bnd} I_k(t-\delta, t) - \xi I_k(t - \delta, t)$.

Because $\xi \leq A_{bnd}$, so $\varpi(i,x) > A_{bnd} * x$, i.e. $\sum_{i=1}^N A_i \min(I_{i,k}(t-\delta, t), x) > (A(H) - A_k + 1)(D_k - C_k)$ It also contradicts the lemma. So the assumption cannot hold.

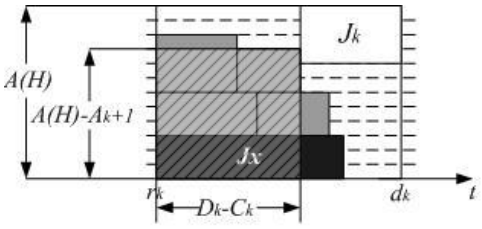


Figure 2. Illustration of Lemma 3

Fig. 2 shows the intuitive meaning of Lemma 3. The gray and black blocks represent the jobs that preempt J_k. It is obvious that if one can guarantee the whole system work done by these jobs to be less than the size of the shadowed part, the interference certainly cannot cause J_k to miss deadline. Furthermore, if the interference contribution of some job is larger than $D_k - C_k$ (the black job), we only need to consider the$(D_k - C_k)$ part.

To apply the schedulability test of Lemma 3, the most straightforward approach is to compute the interference $I_i, k(r_k^j, d_k^j)$ for each task τ_i in every interval $[r_k^j, d_k^j]$ until the end of the hyper-period. This is not possible without running a simulation of the system. To avoid the complexity of this approach, we will derive an upper bound on the interference. Obviously, we know that the interference $I_{i,k}(r_k^j, d_k^j)$ cannot be larger than the time work $W_i^T(r_k^j, d_k^j)$, so the upper bound of $W_i^T(r_k^j, d_k^j)$ is also the upper bound of $I_{i,k}(r_k^j, d_k^j)$.

In the context of multiprocessor scheduling, the worst case for this time work is when the deadlines of job J_k^j and its interfering task τ_i are aligned, because in this case the number of instances of τ_i that interfere with τ_k is maximized [4]. This conclusion also holds true for FPGA scheduling, since the inter-

ference that J_k^j suffers from task τ_i in some given time interval is only related to their execution times, not to their area sizes.

Lemma 4 *An upper bound on the time work of τ_i in the interval $[r_k^j, d_k^j)$ is:*

$$W_i(r_k^j, d_k^j) \leq N_i C_i + \min(C_i, \max(D_k - N_i T_i, 0)).$$

Proof 2 *We only consider the worst case mentioned above. In this situation, the time work $W_i(r_k^j, d_k^j) \leq N_i C_i + \varepsilon_i(r_k^j, d_k^j)$, where $\varepsilon_i(r_k^j, d_k^j)$ is the carry-in[4] of task τ_k in interval $[r_k^j, d_k^j)$. $N_i = (\lfloor (D_k - D_i)/T_i \rfloor + 1)$ is the maximum number of jobs of τ_i that can be completely contained in $[r_k^j, d_k^j)$ in this situation.*

Now we look for the upper bound of the carry-in. Obviously, We have $\varepsilon_i(r_k^j, d_k^j) \leq C_i$. When $T_i > D_i$ and $D_k - N_i T_i > 0$, we can see that the carry-in will never be larger than $D_k - N_i T_i$ in the worst-case situation mentioned above.

We can then prove Theorem 2 based on Lemma 3 and Lemma 4.

Theorem 2 ***(GN1)*** *A taskset Γ is schedulable using EDF-NF if, for each $\tau_k \in \Gamma$*

$$\sum_{i \neq k} A_i \min(\beta_i, 1 - C_k/D_k) < (A(H) - A_k)(1 - C_k/D_k)$$

where $\beta_i = (N_i C_i + \min(C_i, \max(D_k - N_i T_i, 0)))/D_i$.

5 Schedulability Bound Test for EDF-FkF

From Lemma 1, we know that EDF-FkF is global-α-work-conserving, where $\alpha = 1 - (A_{max} - 1))/A(H)$. As a result, the lower bound of system resource utilization in the analysis interval $[t - \delta, t)$ for task τ_k is unrelated to the area size A_k of τ_k. This fact gives us an opportunity to taking advantage of Baker's idea of problem window[1] extension [4]. Baker [4] showed that if one can find a lower bound on the interference load in the problem window that is necessary for a job J_k^j to miss its deadline, and it can be guaranteed that a given set of tasks cannot possibly generate so much load in the problem window, then J_k^j will meet its deadline. Baker showed that one can obtain a tighter lower bound on the interference by considering an extension of the problem window, and in turn obtain a tighter schedulability bound. We will follow the same idea here. Next, we derive lower bound of the interference of τ_k that causes it to miss its deadline.

Lemma 5 *If t is the time of τ_k's first deadline miss of τ_k and $[t - \delta, t)$ is the corresponding maximal τ_k-busy interval then:*

$$I_k(t - \delta, t) > \delta - (\delta + T_k - D_k) C_k / T_k.$$

[1]Let τ_k be the task of a job that misses its deadline for the first time, then τ_k's problem window is $[t, t + d_k)$.

The proof is similar to [8] although it is in the context of FPGA scheduling, since the interference is only related to task execution times. We will not repeat the proof here, and refer the interested reader to the technical report [12] for details.

The time work $W_i^T(t-\delta, t)$ done by τ_i in an interval $[t-\delta, t)$ has an upper-bound of the interval length δ, and may include the following components:

1. A portion of the execution times of the jobs that are released before $t - \delta$ but unable to complete by that time, called the *carry-in*, as defined in [4].
2. The full execution time C_i of the jobs released on or after $t - \delta$ and completed by time t.
3. A portion of the execution time of at most one job released at some time $t - \varepsilon, 0 < \varepsilon \leq \delta$ but is uncomplete by time t.

To bound the carry-in time contribution by τ_i, the maximal τ_k-busy interval is extended downward, i.e., keeping the right endpoint fixed and moving the left endpoint earlier, as far as possible while still maintaining a lower bound on block busy time as in Lemma 5.

Definition 5 *An interval $[t - \delta, t)$ is τ_k^λ-busy for a given constant $\lambda \geq C_k/T_k$ if $B(t - \delta, t) > \delta - \lambda(\delta + T_k - D_k)$. It is a maximal τ_k^λ-busy interval if it is τ_k^λ-busy and there is no $\delta' > \delta$ such that $[t - \delta', t)$ is also τ_k^λ-busy.*

Lemma 6 *If t is the time of the first deadline miss of τ_k^λ and $\lambda \geq C_k/T_k$, then there is a unique maximal τ_k^λ-busy interval $[t - \overline{\delta}, t)$, and $\overline{\delta} \geq D_k$.*

We call the unique interval $[t - \overline{\delta}, t)$ guaranteed by Lemma 6 the maximal τ_k^λ-busy interval, denoted by $[t - \overline{\delta}, t)$. The next step is to find an upper bound on the time work $W_i^T(t - \overline{\delta}, t)$ done by each task τ_i in a τ_k^λ-busy interval $[t - \overline{\delta}, t)$.

Lemma 7 *If t is the time of the first deadline miss of task τ_k, and $\lambda \leq C_k/T_k$ and $[t - \overline{\delta}, t)$ is the corresponding τ_k^λ-busy interval, then for any task τ_i such that $i \neq k$*

$$\frac{W_i^T(t - \overline{\delta}, t)}{\overline{\delta}} \leq \beta_k^i(i)$$

where

$$\beta_k^\lambda(i) = \begin{cases} \max(\frac{C_i}{T_i}, \frac{C_i}{T_i}(1 - \frac{D_i}{D_k}) + \frac{C_i}{D_k}) & \text{if } \frac{C_i}{T_i} \leq \lambda \\ \frac{C_k}{T_k} & \text{if } \frac{C_i}{T_i} > \lambda \wedge \lambda \geq \frac{C_i}{D_i} \\ \frac{C_i}{T_i} + \frac{C_i - \lambda D_i}{D_k} & \text{if } \frac{C_i}{T_i} > \lambda \wedge \lambda < \frac{C_i}{D_i} \end{cases}$$

Lemma 8 *If the interval $[t - \delta, t)$ is a block busy interval, then*

$$\forall i (A(H) - A_{max} + 1) B(t - \delta, t) \leq \sum_{i=1}^{N} A_i B_i(t - \delta, t).$$

Lemma 9 *If the interval $[t-\delta, t)$ is block busy interval and $B(t-\delta, t) > x$, then*

$$\sum_{i=1}^{N} \min(B_i(t-\delta, t), x) > (A(H) - A_{max} + 1)x.$$

Lemma 10 *If the interval $[t-\overline{\delta}, t)$ is τ_k-busy, then we have:*

$$W^S(t-\overline{\delta}, t) > A_{bnd} B(t-\overline{\delta}, t) + A_{min}(\overline{\delta} - B(t-\overline{\delta}, t)),$$

where $A_{bnd} = A(H) - A_{max} + 1$

We omit the proofs of Lemmas 6∼10 and refer the interested reader to the technical report [12].

Now we present Theorem 3 and its proof:

Theorem 3 ***(GN2)*** *Let $\beta_k^\lambda(i)$ be as defined as in Lemma 7 and let $\lambda_k = \lambda \max(1, T_k/D_k)$. A taskset Γ is schedulable using EDF-FkF if, for every task τ_k, there exists $\lambda \geq C_k/T_k$, such that one or more of the following conditions are satisfied:*

$$1) \sum_{i=1}^{N} A_i \min(\beta_k^\lambda(i), 1-\lambda_k) < (A_{bnd}(1-\lambda_k)$$

$$2) \sum_{i=1}^{N} A_i \min(\beta_k^\lambda(i), 1) \leq (A_{bnd} - A_{min})(1-\lambda_k) + A_{min}$$

where $A_{bnd} = A(H) - A_{max} + 1$.

Proof 3 *The proof is by contradiction. Suppose the taskset Γ with a release time assignment r is not schedulable, then there must be some task τ_k that first misses its deadline at time t. Let $[t-\overline{\delta}, t)$ be the maximal τ_k^λ-busy interval guaranteed by Lemma 6. By the definition of τ_k^λ-busy,*

$$\frac{B(t-\overline{\delta}, t)}{\overline{\delta}} > 1 - \lambda + \lambda \frac{T_k - D_k}{\overline{\delta}}. \quad (1)$$

There are two cases: If $T_k \leq D_k$, obviously, $\frac{B(t-\overline{\delta},t)}{\overline{\delta}} = 1 - \lambda\frac{T_k}{D_k}$. If $T_k > D_k$, then the value of the expression on the righthand side of inequality 1 is decreasing with respect to $\overline{\delta}$, and so $\frac{B(t-\overline{\delta},t)}{\overline{\delta}} \geq 1-\lambda$. Since $\lambda_k = \lambda \max(1, \frac{T_k}{D_k})$, so we have

$$\frac{B(t-\overline{\delta}, t)}{\overline{\delta}} \geq 1 - \lambda_k \quad (2)$$

Since $[t-\overline{\delta}, t)$ is τ_k-busy, from Lemma 10,

$$W^S(t-\overline{\delta}, t) > (A_{bnd} - A_{min}) B(t-\overline{\delta}, t) + A_{min}\overline{\delta} \quad (3)$$

Since $W_i^T(t-\overline{\delta}, t) \leq \overline{\delta}$, by (2), (3) and Lemma 7, we have

$$\sum_{i=1}^{N} A_i \min(\beta_k^\lambda, 1) > (A_{bnd} - A_{min})(1-\lambda_k) + A_{min} \quad (4)$$

Table 1. A taskset accepted by DP but rejected by GN1 and GN2.

task	C	D	T	A
τ_1	1.26	7	7	9
τ_2	0.95	5	5	6

Table 2. A taskset accepted by GN1 but rejected by DP and GN2.

task	C	D	T	A
τ_1	4.50	8	8	3
τ_2	8.00	9	9	5

Therefore condition 2) of the theorem must be false. It remains to be shown that condition 1) must also be false.

By Lemma 9 with $x = (1-\lambda_k)\overline{\delta}$ and (2), it must hold that:

$$\sum_{i=1}^{N} A_i \min(\frac{B_i(t-\overline{\delta}, t)}{\overline{\delta}}, 1-\lambda_k) > A_{bnd}(1-\lambda_k) \quad (5)$$

Combining Lemma 7 and (5), and since $B_i(t-\overline{\delta}, t) \leq W_i(t-\overline{\delta}, t)$, we have

$$\sum_{i=1}^{N} A_i \min(\beta_k^\lambda(i), 1-\lambda_k) > A_{bnd}(1-\lambda_k) \quad (6)$$

This contradicts condition 1) of the theorem.

So the assumption cannot hold, hence the taskset Γ must be schedulable.

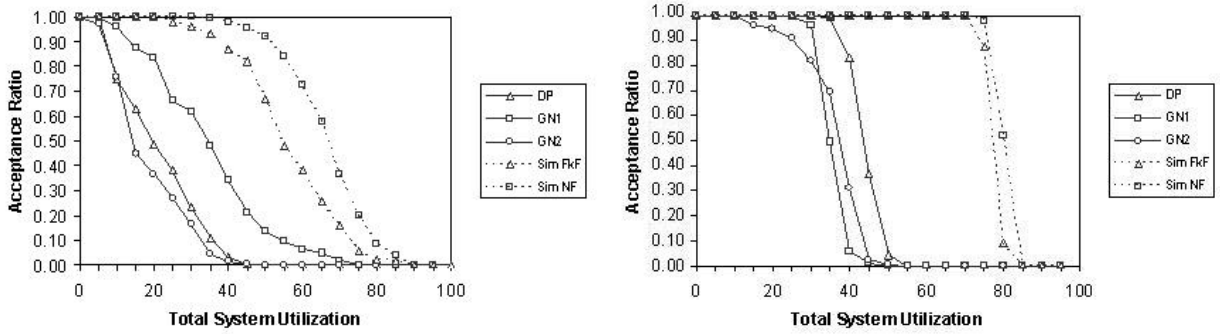
The test in Theorem 3 has running time complexity of $O(N^3)$, since the only values of λ need be considered are the minimum points, and the points where β_k^λ is discontinues, i.e., $\lambda = C_i/T_i, i = 1, ..., N$ and $\lambda = C_i/D_i, if D_i > T_i$.

6 Performance Evaluation

For EDF scheduling on FPGAs, the only known related work is **DP** presented in Danne et al [9], so we compare our tests **GN1** (Theorem 2 (**GN1**) in Section 4) and **GN2** (Theorem 3 (**GN2**) in Section 5) to **DP** (Theorem 1 in Section 3). Note that if the scheduling algorithm is EDF-NF, then we can compare all three tests **DP**, **GN1** and **GN2**; but if the scheduling algorithm is EDF-FkF, then we can only compare **DP** with **GN2**, because **GN1** is not applicable to EDF-FkF.

Tables 1, 2 and 3 show three tasksets that are accepted by one test but rejected by another on a FPGA with 10 columns ($A(H) = 10$). Take the taskset in Table 3 for example:

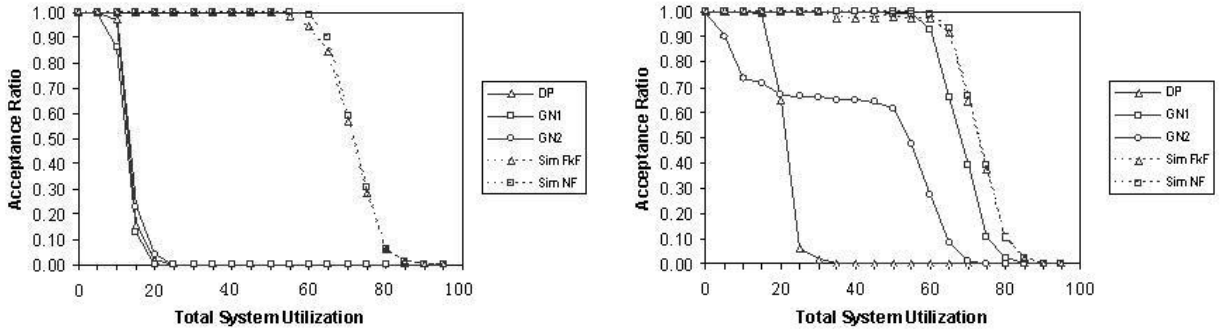
- **DP**:
  $U^S(\Gamma) = 4.94$. When $k = 2$, $(A(H) - A_{max} + 1)(1 - U^T(\tau_2)) + U^S(\tau_2) = 4.85 < 4.94$, so the schedulability test fails and the taskset is rejected.



(a) 4 tasks

(b) 10 tasks

Figure 3. Tasksets with unconstrained execution time and area size distributions.



(a) 10 spatially heavy and temporally light tasks.

(b) 10 spatially light and temporally heavy tasks.

Figure 4. Tasksets with constrained execution time and area size distributions.

Table 3. A taskset accepted by GN2 but rejected by DP and GN1

task	C	D	T	A
τ_1	2.10	5	5	7
τ_2	2.00	7	7	7

- **GN1**:
  When $k = 2$, $(A(H) - A_2 + 1)(1 - C_2/D_2) = \frac{20}{7}$
  $N_1 = 1$, $\beta_1 = \frac{4.1}{5}$, so
  $\sum_{i \neq k} A_i \min(\beta_i, 1 - C_k/D_k) = 5 > \frac{20}{7}$
  The test fails when $k = 2$, and the taskset is rejected.

- **GN2**:
  When $k = 1$ and $\lambda = \frac{C1}{T1}$, $\beta_1^\lambda(1) = 0.42$, $\beta_1^\lambda(2) = 0.29$.
  $(A(H) - A_{max} + 1 - A_{min})(1 - \lambda_k) + A_{min} = 5.26$
  $\sum_{i=1}^{N} A_i \min(\beta_k^\lambda(i), 1 - \lambda_k) = 4.97 < 5.26$. So condition 2) holds for $k = 1$.

  When $k = 2$ and $\lambda = \frac{C1}{T1}$, $\beta_2^\lambda(1) = 0.42$, $\beta_2^\lambda(2) = 0.29$.
  $(A(H) - A_{max} + 1 - A_{min})(1 - \lambda_k) + A_{min} = 5.26$
  $\sum_{i=1}^{N} A_i \min(\beta_k^\lambda(i), 1 - \lambda_k) = 4.97 < 5.26$. Condition 2) also holds for $k = 2$. So the taskset is accepted.

Unlike SW tasks on CPU, HW tasks on FPGA are characterized not only by their timing attributes, but also by their area sizes. Each of the three tests (**DP**, **GN1** and **GN2**) emphasizes different aspects of the taskset characteristics. For example, **GN1** has a more accurate bound of the task area sizes since it uses the tighter lower bound on area utilization of $(A(H) - A_k + 1)$ rather than the $(A(H) - A_{max} + 1)$ used by **DP** and **GN2**, but the bound on the time work contributed by the carry-in jobs may be too pessimistic. On the other hand, **GN2** have a more accurate bound of the carry-in job than **GN1**, but it uses $(A(H) - A_{max} + 1)$ as the lower bound on area utilization, because its analysis window is extended and not interval-α-work-conversing anymore, where $\alpha = (1-(A_k-1)/A(H))$.

We use randomly-generated synthetic tasksets with different timing and area size distributions to evaluate the performance of schedulability bound tests. Total area size of the FPGA is 100, and task area sizes are randomly distributed between 1 and 100. Task periods are randomly distributed in $(5, 20)$. Each task's deadline is equal to its period, and its execution time is the product of its period and a random factor. Each group of experiments contains at least 10000 tasksets in order to guarantee a large enough sample space. The performance metric

is *acceptance ratio*, percentage of tasksets that are accepted by each schedulability bound test, as well as by running simulation with all tasks initially released at time 0. As noted by Baker [8], it is not possible to determine exact schedulability without exhaustively simulating all possible task release offsets, so we use simulation to provide a coarse upper bound on the fraction of the task sets that are schedulable.

In Figs. 3 and 4, we plot the acceptance ratios against the total system utilization as defined in Section 2. We can make the following observations:

- All three tests are indeed pessimistic compared to simulation results.
- For the same total system utilization, **DP** performs best for large number of tasks, while **GN1** performs best for small number of tasks.
- For spatially-heavy tasksets, i.e., tasksets with high spatial utilization, all three tests exhibit poor performance.
- For temporally-heavy tasks, i.e., tasksets with high time utilization, **GN1** performs best while **DP** performs worst.

We can see that no single utilization bound test consistently out-performs others, and each may have better performance for tasksets with different characteristics. In practice, different schedulability bounds should be applied together, i.e., determine that a taskset is unschedulable only if all tests fail.

7 Conclusions and Future Work

We have presented two schedulability bound tests for EDF scheduling of HW tasks on FPGAs, one for EDF-Next-Fit and one for EDF-First-k-Fit, and used synthetic tasksets to evaluate their performance compared to previous work and simulation results. For future work, we plan to relax some of the assumptions mentioned in Section 1 to handle 2D reconfigurable FPGAs, pre-configured cells, FPGA reconfiguration overheads, partitioned scheduling, restricted task migration and FPGA area fragmentation. Especially for 2D reconfiguration, task placement strategy has a large effect on FPGA fragmentation, and we cannot assume that a task can fit on the FPGA as long as there is enough free area, even with free task migrations. We plan to study the impact of different free area management and task placement strategies on the schedulability bound tests. We also plan to investigate the applicability of hybrid scheduling algorithms such as EDF-US$[m^2/(2m-1)]$ [13] (m is the number of processors), in which a few high-utilization tasks are assigned top priority and other tasks are assigned priorities in EDF order. We may need to extend the concept of "high-utilization" to refer to system utilization instead of time utilization in order to take into account impact of a task's area size.

References

[1] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S. Baruah, *A Categorization of Real-Time Multiprocessor Scheduling Problems and Algorithms*. Chapman and Hall/CRC, 2004, pp. 30–1 – 30–19.

[2] M. Harbour, M. H. Klein, and J. Lehoczky, "Timing Analysis for Fixed-Priority Scheduling of Hard Real-Time Systems," *IEEE Trans. Software Eng.*, vol. 20, no. 2, pp. 13–28, 1994.

[3] J. Goossens, S. Funk, and S. K. Baruah, "Priority-Driven Scheduling of Periodic Task Systems on Multiprocessors." *Real-Time Systems*, vol. 25, no. 2-3, pp. 187–205, 2003.

[4] T. P. Baker, "Multiprocessor EDF and Deadline Monotonic Schedulability Analysis." in *IEEE Real-Time Systems Symposium (RTSS)*, 2003, pp. 120–129.

[5] ——, "An Analysis of EDF Schedulability on a Multiprocessor." *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 8, pp. 760–768, 2005.

[6] M. Bertogna, M. Cirinei, and G. Lipari, "Improved schedulability analysis of edf on multiprocessor platforms." in *Euromicro Conference on Real-Time Systems (ECRTS)*, 2005, pp. 209–218.

[7] T. P. Baker, "A comparison of global and partitioned edf schedulability tests for multiprocessors," in *International Conference on Real-Time and Network Systems (RTSN)*, 2006, pp. 119–130. [Online]. Available: http://rtns07.irisa.fr/

[8] ——, "Further improved schedulability analysis of EDF on multiprocessor platforms," Florida State University, Tech. Rep. TR-051001, Oct 2005.

[9] K. Danne and M. Platzner, "An EDF Schedulability Test for Periodic Tasks on Reconfigurable Hardware Devices." in *ACM Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, 2006, pp. 93–102.

[10] ——, "Partitioned Scheduling of Periodic Real-Time Tasks onto Reconfigurable Hardware," in *International Parallel and Distributed Processing Symposium (IPDPS'06), Reconfigurable Architecture Workshop (RAW'06)*, 2006.

[11] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, "Proportionate Progress: A Notion of Fairness in Resource Allocation." *Algorithmica*, vol. 15, no. 6, pp. 600–625, 1996.

[12] N. Guan, Z. Gu, Q. Deng, W. Liu, and G. Yu, "Improved Schedulability Analysis of EDF Scheduling on Reconfigurable Hardware Devices," in *Technical report, available online at http://www.cse.ust.hk/∼zgu/FPGA/.*

[13] A. Srinivasan and S. K. Baruah, "Deadline-based scheduling of periodic task systems on multiprocessors." *Inf. Process. Lett.*, vol. 84, no. 2, pp. 93–98, 2002.