Decomposed software pipelining for cyclic unitary RCPSP with precedence delays

Abir Benabid, Claire Hanen

MISTA 2009

Decomposed Software Pipelining for cyclic unitary RCPSP with precedence delays *

Abir Benabid · Claire Hanen

Abstract In this paper we adress a new cyclic problem: finding periodic schedules for unitary resource constrained cyclic scheduling problem. Such resource constraints are characterized by k, the number of types of functional units employed and m_x the maximal number of processors of the same type. The main problem is to cope with both precedence delays and resources which make the problem $\mathcal{NP}$-complete in general.

A guaranteed approach, called decomposed software pipelining, has been proposed by Gasperoni and Schwiegelshohn, followed by the retiming method by Calland, Darte and Robert to solve the problem for parallel processors and ordinary precedences. We present, in this paper, an extension of this approach to resource-constrained cyclic scheduling problems with precedence delays and we provide an approximation algorithm. Let λ and λ_{opt} be respectively the period given by the algorithm and the optimal period. We establish the bound:

$$\lambda \leq \left(k + 1 - \frac{1}{m_x(\rho+1)}\right)\lambda^{opt} + \left(1 - \frac{1}{m_x(\rho+1)}\right)(\delta - 1).$$

1 Introduction

Cyclic scheduling problems have numerous practical application in production systems [1] as well as in embeded systems [2]. Our research in this field is partically motivated by the advances in hardware technology, but our results still available for mass production systems.

* *ORCYAE project supported by awards from DIGITEO, a research park in Ile-de-France dedicated to information science and technology.*

A. Benabid
LIP6, Pierre and Marie Curie University
E-mail: Abir.Benabid@lip6.fr

C. Hanen
Paris Ouest University and LIP6
E-mail: Claire.Hanen@lip6.fr

Embedded architectures used for devices such as mobile, automotive and consumer electronics need high performance, low silicon implementation costs, low power consumption and rapid development to ensure minimum time-to-market. Most of todays high performance applications uses instruction level parallel processors such as VLIW processors [3].

VLIW architectures are mainly used for media processing in embedded devices, and instruction schedules produced by the compiler are a performance critical optimization that has a direct impact on the overall system cost and energy consumption. High-quality instruction schedules enable to reduce the operating frequency given real-time processing requirements. Most of the parallelism present in these systems is expressed in the form of loops.

In this paper, we consider a loop composed of many tasks that are to be executed a large number of times. Instruction scheduling for inner loops is also known as software pipelining [4] and can be modelised by a cyclic scheduling problem. Among the different cyclic scheduling frameworks, modulo scheduling [5] is the most successful in production compilers. The modulo scheduling focuses on finding a periodic schedule with the minimal period λ.

The classical results of modulo scheduling apply to problems that are too limited to be of practical use in instruction scheduling in modern processors as well as in mass production problems. For example, these results assume simple precedence constraints on tasks in a schedule, instead of precedences with delays like those in pipelined processors, and focus on machine models where each operation uses one of m identical processors for its execution.

In order to model the software pipelining problem, [6] proposes an extension of the classic modulo scheduling problem to resource-constrained modulo scheduling problems with precedence delays where the resources are adapted from the renewable resource of the resource-constrained scheduling problem [7].

We define, in this paper, a special case of this problem where the resource demands are unitary, and we present a guaranteed algorithm for these problems.

1.1 Problem formulation

An instance of a unitary resource-constrained cyclic scheduling problem can be defined by:

- An architecture model defined by $\mathcal{P} = \{P_{(i,j)} \backslash 1 \leq i \leq k, 1 \leq j \leq m_i\}$, where k denotes the number of different types of processors and m_i denotes the number of type i processors. Let $m_x = \max_{1 \leq r \leq k} m_r$.
- A set of n tasks $V = \{T_i\}_{1 \leq i \leq n}$ with integer processing time $\{p_i\}_{1 \leq i \leq n}$. To each task T_i is associated a binary vector $\{b_r^i\}_{1 \leq r \leq k}$ over the resource types, such that T_i uses b_r^i units of resource of type r during its execution. Notice that a task might use several processors but of different types.
- Each task T_i must be performed an infinite number of times. We call T_i at iteration q the q-th execution of T_i.
- Precedence graph $G(V, E)$ where:
  - E is a set of edges defining uniform dependence relations denoted by $T_i \xrightarrow{l_{ij}, h_{ij}} T_j$, where the delay l_{ij} and the height h_{ij} are nonnegative integers. l_{ij} and h_{ij}

model the fact that the task T_j at iteration q has to be issued at least l_{ij} time units after the end of task i in iteration $q - h_{ij}$.

– We denote $\rho = \dfrac{l_{max}}{p_{min}}$ where $l_{max} = \max\limits_{(T_i,T_j)\in E} l_{ij}$ and $p_{min} = \min\limits_{1\leq i\leq n} p_i$.

Notice that this model generalize the classical parallel processors statements (in which $k = 1$ -i.e. there is a unique type of processors) as well as typed tasks systems where each binary vector $\{b_r^i\}_{1\leq r\leq k}$ has only one positive component. Let us illustrate the notions of tasks, iterations, delays and heights with the following example. We will work on this example throughout the paper.

$$
\begin{array}{l}
For\ 1 \leq i \leq N \\
(T_1) : t_1(i) = t_3(i) \\
(T_2) : t_2(i) = t_3(i) \\
(T_3) : t_3(i) = t_2(i-3) \\
(T_4) : t_4(i) = t_2(i-1) \\
(T_5) : t_5(i) = t_4(i-2) \\
\qquad +t_2(i-1) + t_7(i) \\
(T_6) : t_6(i) = t_5(i-1) \\
(T_7) : t_7(i) = t_6(i-1)
\end{array}
$$

Tasks	T_1	T_2	T_3	T_4	T_5	T_6	T_7
p_i	2	2	3	3	1	1	1
$\{b_1^i, b_2^i\}$	1,1	1,1	1,0	1,0	1,1	0,1	1,1

The loop has $n = 7$ tasks, each one is executed N times. N is a given parameter representing the number of iterations and can be very large, and thus is assumed infinite. The associated precedence graph $G(V, E)$ is given in Figure 1. Values of l_{ij} and h_{ij} are displayed next to the corresponding arc. The resource request $\{b_1^i, b_2^i\}$ of each task T_i is highlighted next to the corresponding node.

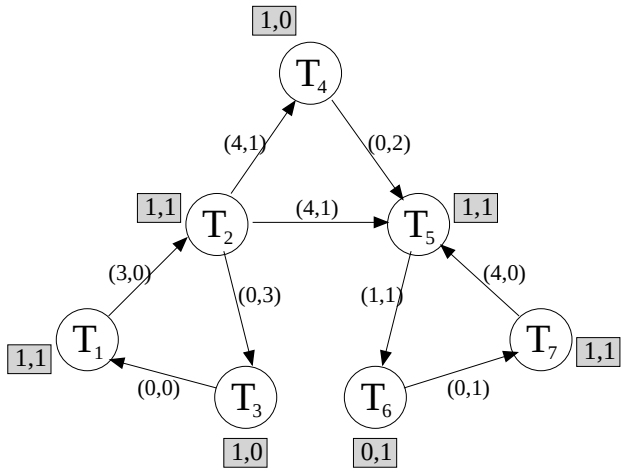

Fig. 1 An example of precedence graph $G(V, E)$.

A resource-constrained cyclic scheduling problem is to find a schedule σ that assigns a starting time $\sigma(T_i, q)$ for each task occurence (T_i, q) such that for all $r \in \{1, \cdots, k\}$

and for each time s, the number of tasks using processors of type r at time s is at most equal to m_r, and

$$\left(T_i \overset{l_{ij},h_{ij}}{\longrightarrow} T_j\right) \Rightarrow \sigma(T_i,q) + p_i + l_{ij} \leq \sigma(T_j, q + h_{ij}) \quad \forall q \in \mathbb{N}$$

The modulo scheduling focuses on finding a periodic schedule with the minimal period λ such that:

$$\forall i \in \{1, \cdots, n\}, \forall q \in \mathbb{N}: \quad \sigma(T_i, q) = \sigma(T_i, 0) + q \cdot \lambda$$

Periodic schedules are of high interest from a practical point of view, because their representation is compact so that they can be easily implemented in real systems.

1.2 Decomposed software pipelining

Generating an optimal resource constrained cyclic scheduling with minimal period is known to be $\mathcal{NP}$-hard. To overcome this $\mathcal{NP}$-hardness, we used the decomposed software pipelining approach introduced simultaneously by Gasperoni and Schwiegelsohn [8], and by Wang, Eisenbeis, Jourdan and Su [9]. The main idea is to decouple the problem into dependence constraints and resource constraints so as to decompose the problem into two subproblems: a cyclic scheduling problem ignoring resource constraints, and a standard acyclic graph for which efficient techniques are known.

Gasperoni and Schwiegelshohn give an efficiency bound to the period λ for the problem with m identical processors and precedences without delays. Let λ_{opt} be the optimal (smallest) period, this bound is given by the following inequality:

$$\lambda \leq \left(2 - \frac{1}{m}\right)\lambda_{opt} + \left(1 - \frac{1}{m}\right)\left(\max_{1 \leq i \leq n} p_i - 1\right)$$

Darte *et al.* [10] presents a heuristic based on circuit retiming algorithms to generalize the efficiency bound given for Gasperoni-Schwiegelshohn algorithm. The main idea is to use a retiming $\mathcal{R}$ to decide which edges to cut in $G(V, E)$ so as to make it acyclic. Then, the acyclic makespan minimization problem is solved using any approximated algorithm, that provides a pattern (schedule of a period) for a feasible periodic schedule.

In this paper, we generalize this approach for our problem. Several new elements have to be taken into account: extended resource constraints and arbitrary precedence delays.

We first use the definition of [10] for a legal retiming:

$$\mathcal{R}: V \to \mathbb{Z}, \quad \forall (T_i, T_j) \in E, \ \mathcal{R}(T_j) + h_{ij} - \mathcal{R}(T_i) \geq 0$$

A legal retiming for G of Figure 1 is given in Table 1.

Table 1 A retiming $\mathcal{R}$ of G.

Tasks	T_1	T_2	T_3	T_4	T_5	T_6	T_7
$\mathcal{R}(T_i)$	0	1	0	2	2	1	1

Then, we define the acyclic graph $G^{\mathcal{R}}$ by keeping only the arcs of G for which $\mathcal{R}(T_j)+h_{ij}-\mathcal{R}(T_i)=0$. We add two dummy tasks $Start$ and $Stop$ with null processing times and no resource use. For each task T_i, we add arcs $(Start, T_i)$ and $(T_i, Stop)$ and we define valuations of these new arcs such that:

$$\forall (T_i, T_j) \in G \backslash G^{\mathcal{R}},\ l_{i,Stop} + l_{Start,j} \geq l_{ij}. \tag{1}$$

Notice that such valuations can be defined as follows for each task T_i:

- $l_{Start,i} = 0$
- $l_{i,Stop} = \max_{(T_i,T_j)\in G\backslash G^{\mathcal{R}}} l_{ij}$.

Let $\pi^{\mathcal{R}}$ be any (non cyclic) schedule of $G^{\mathcal{R}}$ that fulfills the resource constraints as well as the precedences induced by $G^{\mathcal{R}}$. We note $\pi_i^{\mathcal{R}}$ the start time of task T_i in this schedule. Then, setting $\lambda^{\mathcal{R}} = \pi^{\mathcal{R}}_{Stop}$ and for any task T_i,

$$\sigma^{\mathcal{R}}(T_i, q) = \pi_i^{\mathcal{R}} + (q + \mathcal{R}(T_i))\,\lambda^{\mathcal{R}}$$

we get the following result:

Lemma 1 *$\sigma^{\mathcal{R}}$ is a feasible periodic schedule of G with period $\lambda^{\mathcal{R}}$.*

Proof. First, we prove that, at any time slot t, the tasks scheduled at t in $\sigma^{\mathcal{R}}$ meet the resource constraints. We note F_t the set of tasks for which one of occurence is scheduled at t. Let T_i and T_j be two tasks in F_t. We note q and q' their corresponding occurences such that $\sigma^{\mathcal{R}}(T_i,q) \leq t < \sigma^{\mathcal{R}}(T_i,q)+p_i, \sigma^{\mathcal{R}}(T_j,q') \leq t < \sigma^{\mathcal{R}}(T_j,q')+p_j$. Hence,

$$\begin{array}{rcl} t = \sigma^{\mathcal{R}}(T_i,q) + s & = & \sigma^{\mathcal{R}}(T_j,q') + s' \\ \pi_i^{\mathcal{R}} + (q+\mathcal{R}(T_i))\,\lambda^{\mathcal{R}} + s & = & \pi_j^{\mathcal{R}} + \big(q' + \mathcal{R}(T_j)\big)\,\lambda^{\mathcal{R}} + s' \end{array}$$

$$\pi_i^{\mathcal{R}} - \pi_j^{\mathcal{R}} + s - s' - \big(\mathcal{R}(T_j) - \mathcal{R}(T_i) + q' - q\big)\,\lambda^{\mathcal{R}} = 0. \tag{2}$$

Since $\pi_i^{\mathcal{R}} + s < \pi_i^{\mathcal{R}} + p_i \leq \pi^{\mathcal{R}}_{Stop} = \lambda^{\mathcal{R}}$ and similarly $\pi_i^{\mathcal{R}} + s' < \lambda^{\mathcal{R}}$, $-\lambda^{\mathcal{R}} < \pi_i^{\mathcal{R}} + s - \pi_i^{\mathcal{R}} - s' < \lambda^{\mathcal{R}}$. Hence, the equality (2) gives:

$$\pi_i^{\mathcal{R}} + s = \pi_j^{\mathcal{R}} + s' \text{ and } \mathcal{R}(T_i) + q = \mathcal{R}(T_j) + q'$$

Hence, T_i and T_j are performed on the same time slot $\pi_i^{\mathcal{R}} + s$ in the acyclic schedule $\pi^{\mathcal{R}}$. Thus, $T_j \in F_{\pi_i^{\mathcal{R}}+s}$ and then $F_t \subseteq F_{s+\pi_i^{\mathcal{R}}}$

Since $\pi^{\mathcal{R}}$ fullfills the resource constraints induced by $G^{\mathcal{R}}$, $F_{\pi_i^{\mathcal{R}}}$ (and then F_t) meets the resource constraints.

For precedence constraints, we need to prove that, $\forall (T_i,T_j) \in E,\ \forall q \in \mathbb{N}$:

$$\begin{array}{rcl} \sigma^{\mathcal{R}}(T_i,q) + p_i + l_{ij} & \leq & \sigma^{\mathcal{R}}(T_j, q + h_{ij}) \\ \Leftrightarrow \pi_i^{\mathcal{R}} + (q+\mathcal{R}(T_i))\,\lambda^{\mathcal{R}} + p_i + l_{ij} & \leq & \pi_j^{\mathcal{R}} + \big(q + \mathcal{R}(T_j) + h_{ij}\big)\,\lambda^{\mathcal{R}} \end{array}$$

Hence, we have to verify that inequality (3) is satisfied for each (T_i,T_j) in E and for any $q \in \mathbb{N}$

$$\pi_i^{\mathcal{R}} - \pi_j^{\mathcal{R}} + p_i + l_{ij} \leq \big(\mathcal{R}(T_j) - \mathcal{R}(T_i) + h_{ij}\big)\,\lambda^{\mathcal{R}}. \tag{3}$$

Case 1 : $(T_i, T_j) \in G^{\mathcal{R}}$

Then, $\mathcal{R}(T_j) - \mathcal{R}(T_i) + h_{ij} = 0$. Since $\pi^{\mathcal{R}}$ fullfills the precedence constraints induced by $G^{\mathcal{R}}$,

$$\pi_i^{\mathcal{R}} + p_i + l_{ij} \leq \pi_j^{\mathcal{R}}.$$

Hence, inequality 3 is satisfied.

Case 2 : $(T_i, T_j) \notin G^{\mathcal{R}}$

Thus, $\mathcal{R}(T_j) - \mathcal{R}(T_i) + h_{ij} > 0$. And then, using 1

$\pi_i^{\mathcal{R}} - \pi_j^{\mathcal{R}} + p_i + l_{ij} \leq \pi_i^{\mathcal{R}} + p_i + l_{i,Stop} + l_{j,Start} - \pi_j^{\mathcal{R}}$

As $\pi_j^{\mathcal{R}} \geq l_{j,Start}$ and $\pi_i^{\mathcal{R}} + p_i + l_{i,Stop} \leq \pi_{Stop}^{\mathcal{R}}$, we get:

$$\begin{aligned} \pi_i^{\mathcal{R}} - \pi_j^{\mathcal{R}} + p_i + l_{ij} &\leq \pi_{Stop}^{\mathcal{R}} \\ &\leq \lambda^{\mathcal{R}} \\ &\leq \lambda^{\mathcal{R}}(\mathcal{R}(T_j) - \mathcal{R}(T_i) + h_{ij}) \end{aligned}$$

which achieves the proof. ■

Now, the idea, previously used by [8] and [10] is to choose a particular retiming and use a guaranteed algorithm to get a schedule $\pi^{\mathcal{R}}$ of $G^{\mathcal{R}}$, and then to extend the guarantee to the induced periodic schedule.

List scheduling algorithms are the most used heuristics for scheduling with precedence and resource constraints. Alix *et al.* [11] prove that such algorithms have the following worst case performance in the case of m identical processors in presence of precedence delays:

$$C_{max} \leq \left(2 - \frac{1}{m(\rho+1)}\right) C_{max}^{opt}$$

where $\rho = \dfrac{l_{max}}{p_{min}}$. For systems with unit execution time tasks and typed processors, Chou and Chung [12] give the following bound:

$$C_{max} \leq \left(k + 1 - \frac{1}{m_x(l_{max}+1)}\right) C_{max}^{opt}$$

We thus propose the following generic algorithm 1 to solve our problem, by using a list algorithm to produce $\pi^{\mathcal{R}}$.

Algorithm 1: Extended DSP

1. Find a legal retiming $\mathcal{R}$ for G;
2. **for** $(T_i, T_j) \in E$ **do**
   if $\mathcal{R}(T_j) - \mathcal{R}(T_i) + h_{ij} = 0$ **then**
   keep (T_i, T_j) in $G^{\mathcal{R}}$; add nodes Start and Stop;
3. Perform a list scheduling on $G^{\mathcal{R}}$ coping with both precedence and resource constraints. Compute $\pi_i^{\mathcal{R}}$ the start time of task T_i in this schedule and $\lambda^{\mathcal{R}} = C_{max}(G^{\mathcal{R}}) = \pi_{Stop}^{\mathcal{R}}$;
4. Define the cyclic schedule $\sigma^{\mathcal{R}}$ by:
   for $1 \leq q \leq N$ **do**
   for $T_i \in V$ **do**
   $\sigma^{\mathcal{R}}(T_i, q) = \pi_i^{\mathcal{R}} + \lambda^{\mathcal{R}}(q + \mathcal{R}(T_i))$;

The acyclic graph provided by the retiming $\mathcal{R}$ is given by Figure 2 and its corresponding list scheduling allocation is presented in Figure 3. The makespan of this

pattern is $\lambda^{\mathcal{R}} = 9$ and it gives the period of the modulo scheduling of G. The different steps of this heuristic are illustrated by algorithm 1.

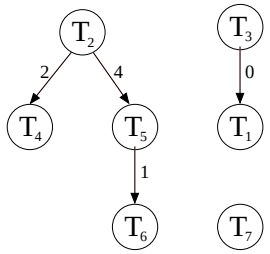

Fig. 2 The acyclic graph given by the retiming $\mathcal{R}$.

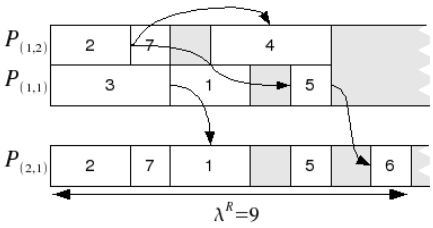

Fig. 3 A pattern generated by a list scheduling of $G^{\mathcal{R}}$: $\lambda^{\mathcal{R}} = 9$.

2 Worst case analysis

In this section we analyze the worst case performance of algorithm 1 in general, making use of the proof of Chou and Chung [12] for list scheduling. Then, we show that using some particular retiming, that can be computed in polynomial time, we can get an overall guarantee for the Extended DSP algorithm.

2.1 Minimal length of pattern

Consider a dependence graph G. An acyclic graph $G^{\mathcal{R}}$ is obtained by a retiming $\mathcal{R}$. Then, we schedule $G^{\mathcal{R}}$ by a list algorithm and generate a pattern $\pi^{\mathcal{R}}$. We note $\phi^{\mathcal{R}}$ the length (sum of the delays and processing times) of the longest path in $G^{\mathcal{R}}$. Let λ^{opt} be the optimal period of G.

We consider two types of bounds obtained from resource and precedence constraints.

2.1.1 Resource bound

Lemma 2 *For each type r, let m_r be the number of machines of type r. Then,*

$$\lambda^{opt} \geq \max_{1 \leq r \leq k} \frac{\sum_{i=1}^{n} b_r^i \cdot p_i}{m_r}.$$

Proof. The shortest time required to complete the tasks using resources of type r on a single machine is $\sum_{i=1}^{n} b_r^i \cdot p_i$. Hence, on m_r parrallel processors, the shortest time required is $\frac{\sum_{i=1}^{n} b_r^i \cdot p_i}{m_r}$. Furthermore, the length of the optimal period λ^{opt} is not shorter than the time required to schedule these tasks once. ■

2.1.2 Precedence bounds

Let $\pi^{\mathcal{R}}$ be a schedule induced by a list algorithm on $G^{\mathcal{R}}$. In order to reveal the dependencies among tasks, we classify the time slots into three kinds:

1. A full slot t_f is a time slot in which at least all the processors of a certain type are executing tasks.
2. A partial slot t_p is a time slot in which at least one processor of each type is idle and this slot contains at least one non-idle processor.
3. A delay slot t_d is a time slot in which all processors are idle.

We note:

- p: the number of partial slots in $\pi^{\mathcal{R}}$.
- d: the number of delay slots in $\pi^{\mathcal{R}}$.

Lemma 3 *The partial-slots lemma:*
If $\pi^{\mathcal{R}}$ contains p partial slots and d delay slots, then $\phi^{\mathcal{R}} \geq p + d$.

Proof. We prove this lemma by finding a chain $h =< T_{j_1}, \cdots, T_{j_c} >$ in $\pi^{\mathcal{R}}$ such that the length of h is at least equal to $p + d$.

Let $T_{j_c} = Stop$ and assume that we already have a chain $< T_{j_{i+1}}, \cdots, T_{j_c} >$. Consider, if it exists, the predecessor T_{j_i} of $T_{j_{i+1}}$ such that $\pi^{\mathcal{R}}_{j_i} + p_{j_i} + l_{j_i,j_{i+1}}$ is maximum.

The construction of h leads to the following observation: All the slots before $\pi^{\mathcal{R}}_{j_1}$ or between $\pi^{\mathcal{R}}_{j_i} + p_{j_i} + l_{j_i,j_{i+1}}$ and $\pi^{\mathcal{R}}_{j_{i+1}}$ (if they exist) are full slots and in which there is no available processor in a type of resource used by $T_{j_{i+1}}$. Otherwise, $T_{j_{i+1}}$ would have been scheduled earlier by the list algorithm.

Therefore, all the p partial slots and d delay slots are covered by the intervals $[\pi^{\mathcal{R}}_{j_i}, \pi^{\mathcal{R}}_{j_i} + p_{j_i} + l_{j_i,j_{i+1}})$, so that the length of h is not less than $p+d$. Thus, $\phi^{\mathcal{R}} \geq p+d$. ■

Lemma 4 *The delay-slots lemma:*
If $\pi^{\mathcal{R}}$ contains d delay slots, then $\phi^{\mathcal{R}} \geq d + \left\lceil \frac{d}{l_{max}} \right\rceil p_{min}$.

Proof. The schedule is computed by a list algorithm, then the number of any consecutive delay slots is not greater than l_{max}. Consider the chain h defined in the previous lemma. All the delay slots are included in $\cup_{1 \leq i \leq c-1} [\pi^{\mathcal{R}}_{j_i} + p_{j_i}, \pi^{\mathcal{R}}_{j_i} + p_{j_i} + l_{j_i,j_{i+1}})$, since during interval $[\pi^{\mathcal{R}}_{j_i}, \pi^{\mathcal{R}}_{j_i} + p_{j_i})$, T_{j_i} is performed. Now the length of each interval $[\pi^{\mathcal{R}}_{j_i} + p_{j_i}, \pi^{\mathcal{R}}_{j_i} + p_{j_i} + l_{j_i,j_{i+1}})$ is less than l_{max}. So it holds that $c \cdot l_{max} \geq d$. The length of h is thus not less than d plus the length of the chained tasks, which is greater than $c \cdot p_{min} \geq \left\lceil \frac{d}{l_{max}} \right\rceil p_{min}$. Thus, $\phi^{\mathcal{R}} \geq d + \left\lceil \frac{d}{l_{max}} \right\rceil p_{min}$. ■

2.2 Performance bound

Here we define the notations to be used below:
$M = \sum_{1 \leq r \leq k} m_r$.
u_r: the number of non-idle cycles on processors of type r.

v_r: the number of non-idle cycles on processors of type r in partial slots in $\pi^{\mathcal{R}}$.
$V = \sum_{1 \le r \le k} v_r$: the number of non-idle cycles in partial slots in $\pi^{\mathcal{R}}$.

We now prove a first bound ont the algorithm performance based on the length of the longest path in $G^{\mathcal{R}}$.

Theorem 1 *Consider a dependence graph G. Let $\mathcal{R}$ be a legal retiming $\mathcal{R}$ on G and $\phi^{\mathcal{R}}$ the length of the longest path in $G^{\mathcal{R}}$. Then,*

$$\frac{\lambda^{\mathcal{R}}}{\lambda^{opt}} \le k + \left(1 - \frac{1}{m_x(\rho+1)}\right)\frac{\phi^{\mathcal{R}}}{\lambda^{opt}}.$$

Proof. Consider the pattern $\pi^{\mathcal{R}}$:

$$M\lambda^{\mathcal{R}} = \text{number of non-idle cycles} + \text{number of idle cycles.}$$

where the second term on the right hand side can be decomposed as the number of idle cycles occurring during delay, partial and full slots.

1. The number of idle cycles per processor occurring during delay slots is equal to Md.
2. The number of idle cycles per processor occurring during partial slots is at most equal to $Mp - V$.
3. Using Lemma 2, we now give a bound on the number of idle cycles occurring during full slots.
   Notice that for a resource type r, there are at most $\frac{u_r - v_r}{m_r}$ full slots in which resource r is full. Thus the number of idle cycles in these slots is at most $(M - m_r)\frac{u_r - v_r}{m_r}$. Hence we get:
   $$\begin{aligned}
   &\le \sum_{1 \le r \le k} (M - m_r)\frac{u_r - v_r}{m_r} \\
   &\le \sum_{1 \le r \le k} (M - m_r)\frac{u_r}{m_r} - \sum_{1 \le r \le k} (M - m_r)\frac{v_r}{m_r} \\
   &\le \sum_{1 \le r \le k} (M - m_r) \max_{1 \le r \le k} \frac{\sum_{i=1}^{n} b_r^i \cdot p_i}{m_r} - (M - m_x)\frac{\sum_{1 \le r \le k} v_i}{m_x} \\
   &\le (kM - M)\lambda^{opt} - (M - m_x)\frac{V}{m_x}
   \end{aligned}$$

   Then,
   $$\begin{aligned}
   M\lambda^{\mathcal{R}} &\le M\lambda^{opt} + (kM - M)\lambda^{opt} - (M - m_x)\frac{V}{m_x} + Mp - V + Md \\
   &\le kM\lambda^{opt} - (M - m_x)\frac{V}{m_x} + Mp - V + Md \\
   &\le kM\lambda^{opt} - M\frac{V}{m_x} + Mp + Md
   \end{aligned}$$

   V is the number of non-idle cycles in partial slots, since a partial slot contains at least one non-idle cycle, $V \ge p$. Thus,

$$\begin{aligned}
\lambda^{\sigma} &\leq k\lambda^{opt} - \frac{p}{m_x} + p + d \\
&\leq k\lambda^{opt} + (1 - \frac{1}{m_x})(p+d) + \frac{1}{m_x}d \\
&\leq k\lambda^{opt} + (1 - \frac{1}{m_x})(p+d) + \frac{1}{m_x}d\left(1 + \frac{p_{min}}{l_{max}}\right)\left(\frac{1}{1 + \frac{p_{min}}{l_{max}}}\right) \\
&\leq k\lambda^{opt} + (1 - \frac{1}{m_x})(p+d) + \frac{1}{m_x}\left(d + \left\lceil \frac{d}{l_{max}} \right\rceil p_{min}\right)\left(\frac{\rho}{\rho+1}\right)
\end{aligned}$$

From Lemmas 3 and 4, we have $\phi^{\mathcal{R}} \geq p + d$ and $\phi^{\mathcal{R}} \geq d + \left\lceil \frac{d}{l_{max}} \right\rceil p_{min}$. Then,

$$\begin{aligned}
\lambda^{\mathcal{R}} &\leq k\lambda^{opt} + (1 - \frac{1}{m_x})\phi^{\mathcal{R}} + \frac{1}{m_x}\frac{\rho}{\rho+1}\phi^{\mathcal{R}} \\
&\leq k\lambda^{opt} + (1 - \frac{1}{m_x(\rho+1)})\phi^{\mathcal{R}}
\end{aligned}$$

■

2.3 Choosing a good retiming

In order to improve the performance bound, it seems important to minimize the ratio between $\phi^{\mathcal{R}}$ and λ^{opt}. So if we have a good lower bound LB of λ^{opt}, using theorem 7 in [13], we can verify the existence of a legal retiming $\mathcal{R}'$ such that $\phi^{\mathcal{R}'} \leq LB \leq \lambda^{opt}$. If such retiming exists, we have a performance guarantee of:

$$\frac{\lambda^{\mathcal{R}'}}{\lambda^{opt}} \leq k + 1 - \frac{1}{m_x(\rho+1)}.$$

An another approach consists on minimizing the length of the longest path in the pattern. There are well-known retiming algorithms [13] to minimize $\phi^{\mathcal{R}}$. Let $\mathcal{R}_{opt}$ be a retiming for which the length of the longest path in the acyclic graph $G^{\mathcal{R}_{opt}}$ is minimal. We note it ϕ^{opt}. We also denote by σ_∞ an optimal periodic schedule for unlimited resources (with period λ_∞).

Lemma 5 *Let $\delta = \max_{(T_i,T_j)\in E}(p_i + l_{ij})$ be the maximum scope in G. Then,*

$$\lambda_\infty + \delta - 1 \geq \phi^{opt}.$$

Proof. Consider an optimal cyclic schedule σ_∞ for unlimited resources. Let us define $r : V \to [0, \lambda_\infty - 1]$ and $\mathcal{R} : V \to \mathbb{Z}$ such that:

$$\sigma_\infty(T_i, q) = r(T_i) + \lambda_\infty\left(q + \mathcal{R}(T_i)\right),\ \forall T_i \in V, \forall q \in \mathbb{N}$$

Then, the precedence constraint for each edge $(T_i, T_j) \in E$ is:

$$\begin{aligned}
\sigma_\infty(T_i, q) + p_i + l_{ij} &\leq \sigma_\infty(T_j, q + h_{ij}) \\
r(T_i) + \lambda_\infty\left(q + \mathcal{R}(T_i)\right) + p_i + l_{ij} &\leq r(T_j) + \lambda_\infty\left(q + h_{ij} + \mathcal{R}(T_j)\right) \\
r(T_i) + p_i + l_{ij} &\leq r(T_j) + \lambda_\infty\left(h_{ij} + \mathcal{R}(T_j) - \mathcal{R}(T_i)\right)
\end{aligned}$$

[10] proved that $\mathcal{R}$ defines a valid retiming for G. Furthermore, $G^{\mathcal{R}}$ is obtained by keeping the edges of G for which $\mathcal{R}(T_j) + h_{ij} - \mathcal{R}(T_i) = 0$. Thus ,

$$r(T_i) + p_i + l_{ij} \leq r(T_j),\ \forall (T_i, T_j) \in E$$

Let $h =< T_{j_1}, \cdots, T_{j_c}, Stop >$ be a chain in $G^{\mathcal{R}}$.

$$r(T_{j_i}) + p_{j_i} + l_{j_i,j_{i+1}} \leq r(t_{j_{i+1}}),\ \forall i \in \{1, \cdots, c-1\},$$

By summing up these $c-1$ inequalities, we have

$$r(T_{j_1}) + \sum_{i=1}^{c-1}(p_{j_i} + l_{j_i,j_{i+1}}) \leq r(T_{j_c})$$

Thus, $\sum_{i=1}^{c-1}(p_{j_i} + l_{j_i,j_{i+1}}) \leq r(T_{j_c})$. This inequality is true for any chain of $G^{\mathcal{R}}$ in particular for the longest path in $G^{\mathcal{R}}$. Hence,

$$\underbrace{\sum_{i=1}^{c-1}(p_{j_i} + l_{j_i,j_{i+1}}) + (p_{j_c} + l_{j_c,Stop})}_{\phi^{\mathcal{R}}} - \underbrace{(p_{j_c} + l_{j_c,Stop})}_{\leq \delta} \leq \underbrace{r(T_{j_c})}_{\leq \lambda_\infty - 1}$$

Hence, $\phi^{\mathcal{R}} - \delta \leq \lambda_\infty - 1$ and since $\phi^{opt} \leq \phi^{\mathcal{R}}$, we have the desired result. ■

Finally, from lemma5, we deduce our last performance ratio.

Theorem 2 *Consider a dependence graph G. Let $\mathcal{R}_{opt}$ be a retiming on G that minimize the length of the longest path in $G^{\mathcal{R}}$. Then,*

$$\lambda^{\mathcal{R}_{opt}} \leq \left(k + 1 - \frac{1}{m_x(\rho+1)}\right)\lambda^{opt} + \left(1 - \frac{1}{m_x(\rho+1)}\right)(\delta - 1).$$

Proof. Using theorem 1 applied to $\mathcal{R}_{opt}$, we get:

$$\begin{aligned}
\lambda^{\mathcal{R}_{opt}} &\leq k\lambda^{opt} + (1 - \frac{1}{m_x(\rho+1)})\phi^{opt} \\
&\leq k\lambda^{opt} + (1 - \frac{1}{m_x(\rho+1)})(\lambda^{\infty} + \delta - 1) \\
&\leq k\lambda^{opt} + (1 - \frac{1}{m_x(\rho+1)})(\lambda^{opt} + \delta - 1) \\
&\leq (k + 1 - \frac{1}{m_x(\rho+1)})\lambda^{opt} + (1 - \frac{1}{m_x(\rho+1)})(\delta - 1)
\end{aligned}$$

■

3 Conclusion

Instruction scheduling, which takes place when compiling applications for modern processors, affects critically the performance of the the overall system cost and energy consumption.

In this paper, we presented a generalized model of instruction scheduling but our results are still available for applications in production systems problems.

We have built upon results of [8,10,12] and extended them to propose a guaranteed heuristic for unitary resource-constrained modulo scheduling problems. A worst case analysis of this heuristic is explored and a performance bound is established. It is the first guarantee derived for cyclic scheduling problems in the case of many different resources.

We point out that it would be interesting to derive algorithms more sophisticated than list scheduling to improve this performance bound.

Finally, it would be worth to study the importance of choosing a good retiming and its impact on the performance guarantee. Good lower bounds for the optimal period would give a better guarantee.

References

1. M., P.J., X., X.: Modélisation, analyse et optimisation des systèmes à fonctionnement cyclique. Masson (1995)
2. Hanen, C., Munier, A.: Cyclic scheduling on parallel processors: An overview. In: P. Chrétienne, E.G. Coffman, J.K. Lenstra, Z. Liu (eds.) Scheduling theory and its applications. J. Wiley and sons (1994)
3. Rau, B.R.: Dynamically scheduled vliw processors. In: MICRO 26: Proceedings of the 26th annual international symposium on Microarchitecture, pp. 80–92. IEEE Computer Society Press, Los Alamitos, CA, USA (1993)
4. Allan, V.H., Jones, R.B., Lee, R.M., Allan, S.J.: Software pipelining. ACM Comput. Surv. **27**(3), 367–432 (1995)
5. Rau, B.R.: Iterative modulo scheduling: an algorithm for software pipelining loops. In: MICRO 27: Proceedings of the 27th annual international symposium on Microarchitecture, pp. 63–74. ACM, New York, NY, USA (1994)
6. Artigues, C., Dinechin, B.D.: Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications. C. Artigues, S. Demassey, E. Nron (2008)
7. Brucker, P., Drexl, A., Mohring, R., Neumann, K., Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research **112**(1), 3–41 (1999)
8. Gasperoni, F., Schwiegelshohn, U.: Generating close to optimum loop schedules on parallel processors. Parallel Processing Letters **4**, 391–403 (1994)
9. Wang, J., Eisenbeis, C., Jourdan, M., Su, B.: Decomposed software pipelining: a new perspective and a new approach. Int. J. Parallel Program. **22**(3), 351–373 (1994). DOI http://dx.doi.org/10.1007/BF02577737
10. Calland, P.Y., Darte, A., Robert, Y.: Circuit retiming applied to decomposed software pipelining. IEEE Trans. Parallel Distrib. Syst. **9**(1), 24–35 (1998). DOI http://dx.doi.org/10.1109/71.655240
11. Munier, A., Queyranne, M., Schulz, A.S.: Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In: IPCO, pp. 367–382 (1998)
12. Chou, H.C., Chung, C.P.: Upper bound analysis of scheduling arbitrary delay instruction on typed pipelined processors. Int. Journal of High Speed Computing **4**(4), 301–312 (1992)
13. Leiserson, C.E., Saxe, J.B.: Retiming synchronous circuitry. NASA STI/Recon Technical Report N **89**, 17,797–+ (1988)