EquiMax. A New Formulation of Acyclic Scheduling Problem for ILP Processors

Sid Touati

EquiMax: A New Formulation of Acyclic Scheduling Problem for ILP Processors

Sid-Ahmed-Ali Touati

November 9, 2000

Abstract

In this paper, we give a new formulation of acyclic scheduling problem under registers and resources constraints in multiple instructions issuing processors (VLIW or superscalar). Given a direct acyclic data dependence graph $G = (V, E)$, the complexity of our integer linear programming model is bounded by $\mathcal{O}(|V|^2)$ variables and $\mathcal{O}(|E| + |V|^2)$ constraints according to a target architecture description. This complexity is better than the complexity of the existing techniques which includes a worst total schedule time factor.

1 Introduction

To sustain increases in processor performance, current compilers try to take benefit from instruction level parallelism (ILP) present in nowadays processors. Multiple operations are issued in the same clock cycle to increase the throughput of executed operations per cycle. Completing a computation as soon as possible is a scheduling problem constrained by many factors. The most important ones are the data dependencies, the availability of hardware features and the availability of registers. The data dependencies define the code semantic and the intrinsic available ILP in the code. The resources constraints limit the the number of instructions issued during the same clock cycle because of the lack of free functional units (FU). Also, architectural characteristics of current processors reveal heterogeneous and complex pipelined FUs where an operation can use a group of FUs in different clock cycles during its presence in the pipeline. Finally, since accessing a register has a null latency, we need to keep as many values in the registers as possible.

Unfortunately, theoretical studies on scheduling reveal that integrating resources constraints [3] or registers constraints [13] are two NP-complete problems. Combining scheduling under both registers and resources constraints is also NP-complete [4]. General compilers use many heuristics to get an optimized schedule in polynomial time complexity. However, embedded applications or real time systems may need optimal (best) schedule. For this purpose, we need

a "good" formulation for the problem. Many work has been done using integer linear programming (intLP) models. In our work, we present a new formulation of acyclic scheduling in basic blocs (BB) such that the complexity of the model generated is lower than the current ones, like we will explain in the end of this paper. Our formulation should reduce the resolution time since we considerably reduce the number of variables and constraints in the generated intLP model.

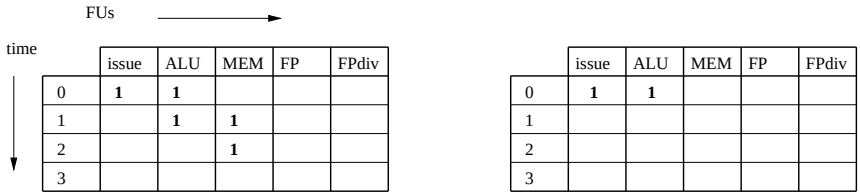
This paper is organized as following. We first present the model of the targeted processors in Sec. 2 and the acyclic data dependence graph (DDG) to be scheduled in Sect. 3 : in our study, we assume heterogeneous FUs, more than one register type, and delayed latencies of writing into and reading from registers. The problem of acyclic scheduling is briefly recalled in Sect. 4. After, we define some intLP modeling techniques in Sect. 5 to show how to linearize some logical operators (disjunction and equivalence) and how to compute the maximum of a set of integers. We then use these techniques to write our EquiMax (Equivalence-Maximum) intLP formulation in Sect. 6. We present some achieved work in this field in Sect. 7 and conclude by our remarks and perspectives in Sect. 8.

2 Machine Description

An ILP processor [14] takes benefit from inherent parallelism in the instruction flow and issues multiple operations per clock cycle thanks to the pipelined execution and the presence of multiple functional units (FUs). An operation can be executed on one or more functional units (FU). We model the complex behavior of the execution of the operations on FUs by the reservation tables [15]. We attach to each instruction a reservation table (RT) to describe at which clock cycle a FU is busy due to the execution of that instruction on it. A RT consists in a two-dimensional table, where the number of lines is the latency of the operation, and the columns consists in the set of FUs. Given a RT of an instruction u, $\mathcal{RT}_u(c,q) = 1$ means that u executes on the FU q during the clock cycle c after its issuing. The number of columns in $\mathcal{RT}$ is bounded by the set of FUs, and the number of lines bounded by the deep of the pipeline.

The target machine $\mathcal{M}$ is described by the set of its hardware resources, its registers types, and the set of operations which execute on these resources :

1. the set of registers types in the target architecture is $\mathcal{T}$. For instance, the target architecture of the code in Fig. 2 has $\mathcal{T} = \{int, float\}$;
2. the machine resources are represented by the couple $(Q, \vec{NQ})$:
   - $Q = \{q_1, \ldots, q_M\}$ is the set of the different FUs ;
   - $\vec{NQ} = [N_{q_1}, \ldots, N_{q_M}]$ where N_q is the number of copies of q.
3. the set of instructions is represented by a couple $(\mathcal{IS}, \vec{\mathcal{RT}})$:
   - $\mathcal{IS} = \{u\}$ is the instructions set which can be executed on $\mathcal{M}$;



(a) $\mathcal{RT}_{fload}$ (b) $\mathcal{RT}_{iadd}$

Figure 1: Reservation Tables

- $\vec{\mathcal{RT}} = [\mathcal{RT}_{u_1}, \mathcal{RT}_{u_2}, \ldots]$ such that $\mathcal{RT}_u$ designates the RT of the instruction u.

In some architectures (superscalar), the processor cannot issue more than m operations even if the number of FUs is greater than m. To handle this case, we consider a virtual "issue' FU' with m copies, such that all instructions use it once at the issue cycle. Figure. 1 gives two examples of RTs.

3 DDG Model

An acyclic DDG $G = (V, E, \delta, \delta_{w,t}, \delta_{r,t})$ consists in a set of operations V and a set of arcs E. Each operation u has a latency $lat(u)$. We assume one sink operation $\perp$ in G in order to catch the total schedule time: if there is more than one sink node, we add the virtual node $\perp$ with an arc e from each sink s to $\perp$ with$\delta(e) = lat(s)$. A valid schedule of G is a positive integer function σ that associates to each operation u an issue time $\sigma(u)$. Any acyclic schedule σ of G must ensure that:

$$\forall e = (u, v) \in E : \quad \sigma(v) - \sigma(u) \geq \delta(e)$$

The total schedule time of G is noted $\overline{\sigma} = \sigma(\perp)$.

In this paper, we consider that each operation $u \in V$ writes into at most one register of a type $t \in \mathcal{T}$. The operations that define multiple values with different types are accepted in our model iff they do not define more than one value of a certain type. For instance, operations that write into a floating point register and set a condition flag are taken into account in our model. We denotes by u^t the value of type t defined by the operation u.

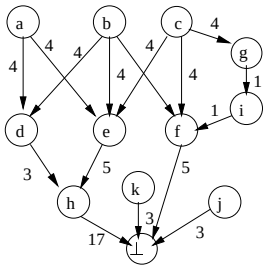
We also consider the following sets:

1. $V_{R,t}$ is the set of values of type $t \in \mathcal{T}$. In Fig. 2, $V_{R,float} = \{a, b, c, d, g, f, h, j, k\}$;
2. $E_{R,t}$ is the set of flow dependency arcs through a value of type $t \in \mathcal{T}$. For instance $E_{R,int} = \{(g, i), (i, f)\}$. If there is some values not read in the DDG, or are still alive after leaving this DDG[1], these values have to

[1] An inter BB data dependence analysis can reveal that a value is still used after the treated DDG.

(a) fload [i1], fR_a
(b) fload [i2], fR_b
(c) fload [i3], fR_c
(d) fmult fR_a, fR_b, fR_d
(e) imultadd fR_a, fR_b, fR_c, iR_e
(g) ftoint fR_c, iR_g
(i) iadd iR_g, 4, iR_i
(f) fmultadd fR_b, iR_i, fR_c, fR_f
(h) fdiv fR_d, iR_e, fR_h
(j) fadd fR_j, 1 , fR_j
(k) fsub fR_k, 1 , fR_k

(1) code before scheduling and register allocation



(2) the DDG G

Figure 2: DDG model

be kept in registers. We consider then that there is a flow arc from these values to $\perp$ (like the flow arc $(k, \perp) \in E_{R,float}$).

Finally, we consider that reading from and writing into a register may be delayed from the beginning of the schedule time (VLIW case). We define the two delay functions $\delta_{r,t}$ and $\delta_{w,t}$ such that :

$$\begin{array}{ll} \delta_{w,t}: & V_{R,t} \rightarrow \mathbb{N} \\ & u \mapsto \delta_{w,t}(u) / \ 0 \leq \delta_{w,t}(u) < lat(u) \\ & \text{the write cycle of } u^t \text{ into a register of type } t \text{ is } \sigma(u) + \delta_{w,t}(u) \\ \delta_{r,t}: & V \rightarrow \mathbb{N} \\ & u \mapsto \delta_{r,t}(u) / \ 0 \leq \delta_{r,t}(u) \leq \delta_{w,t}(u) < lat(u) \\ & \text{the read cycle of } u^t \text{ from a register of type } t \text{ is } \sigma(u) + \delta_{r,t}(u) \end{array}$$

4 Scheduling Problem

Like explained above, a valid schedule σ of G is first constrained by the inherent data dependency relations between operations or any other serial constraints. The target architecture add other constraints which are registers and resources constraints.

4.1 Registers Constraints

Given a DDG $G = (V, E, \delta, \delta_{w,t}, \delta_{r,t})$, a value $u^t \in V_{R,t}$ is alive at the first step after the writing of u^t until its last reading (consumption). The set of consumers of a value $u^t \in V_{R,t}$ is the set of operations that read it :

$$Cons(u^t) = \{v / \ \exists e = (u, v) \in E_{R,t}\}$$

For instance, $Cons(b^{float}) = \{d, e, f\}$ and $Cons(k^{float}) = \{\perp\}$ in Fig. 2. The last consumption of a value is called the killing date and noted:

$$\forall u^t \in V_{R,t} \quad kill(u^t) = \max_{v \in Cons(u^t)} \left(\sigma(v) + \delta_{r,t}(v)\right)$$

We assume that a value written at a clock cycle c in a register is available one step later. That is to say, if operation u reads from a register at a clock cycle c while operation v is writing in it at the same clock cycle, u does not get v's result but gets the value that was previously stored in that register. Then, the **lifetime interval** LT_{u^t} of the value u^t is $]\sigma(u) + \delta_{w,t}(u), kill(u^t)]$. Having all value's lifetime intervals, the number of registers of type t needed to store all defined values is the maximum number of values of type t that are simultaneously alive. We call this number the register need and we note it:

$$RN_t(G) = \max_{0 \leq c \leq \overline{\sigma}} |vsa_t(i)|$$

where

$vsa_t(c) = \{u^t \in V_{R,t} / c \in LT_{u^t}\}$ is the set of values of type t alive at clock cycle c

To compute the register need of type t, we build the indirected interference graph $H_t = (V_{R,t}, \mathcal{E})$, such that u^t and v^t are adjacent iff they are simultaneously alive. The register need $RN_t(G)$ is then the cardinality of the maximal clique (complete subgraph) of H_t.

Since the number $\mathcal{R}_t$ of available registers of type t is limited in the target machine, we need to find a schedule which doesn't need more than $\mathcal{R}_t$ registers for each register type t:

$$\forall t \in \mathcal{T} \qquad RN_t(G) \leq \mathcal{R}_t$$

If we cannot find such schedule, spill code has to be generated, i.e. we must store some values in memory rather than in registers. Spilling increases the total schedule time because it inserts new operations in the BB and the spilled data may cause cache misses.

4.2 Resources Constraints

Resources constraints are simply the fact that two operations must not execute simultaneously on the same FU, i.e. the total number of operations which execute on a FU q during a clock cycle c must not exceed the number of the FU copies N_q. By using the reservation table defined above, an operation u executes on a FU q during a clock cycle c iff $\mathcal{RT}_u[c - \sigma(u), q] = 1$. Formally, the resources constraints are written:

$$\forall 0 \leq c \leq \overline{\sigma},\ \forall q \in Q \qquad \sum_{u \in V} \mathcal{RT}_u[c - \sigma(u), q] \leq N_q$$

5 Integer Linear Programming Techniques

An integer linear programming problem (intLP) [7] is to solve:

$$\begin{cases} \text{maximize (or minimize) } cx \\ \text{subject to } Ax = b \end{cases}$$

with $c, x \in \mathbb{N}^n \ : \ x \geq 0$, and A is an $(m \times n)$ constraints matrix. This is the standard formulation. In fact, we can use any other linear constraints ($\leq, \geq, >$, $>, =$).

5.1 Writing Logical Operators with Linear Constraints

Intrinsically, an ILP problem defines the two boolean operators $\wedge$ and $\neg$:

- having two constraints matrix A and A', saying that x must be a solution for both $Ax \geq b$ and $A'x \geq b'$ is modeled by :

$$\begin{pmatrix} A \\ A' \end{pmatrix} x \geq \begin{pmatrix} b \\ b' \end{pmatrix}$$

- having a constraints matrix A with m lines (m linear constraints $f_1, f_2, \cdots, f_m$), forcing x to do not verify $Ax \geq b$ is modeled by:

$$f_1(x) < b_1 \ \vee \ f_2(x) < b_2 \ \vee \ \cdots \ \vee \ f_m(x) < b_m$$

In [7], the authors shown how to model the disjunctive operator $\vee$. Consider the problem:

$$\begin{cases} \text{maximize (or minimize) } f(x) \\ \text{subject to : } g(x) \geq 0 \ \vee \ h(x) \geq 0 \end{cases}$$

By introducing a binary variable $\alpha \in \{0, 1\}$, this disjunction is equivalent to:

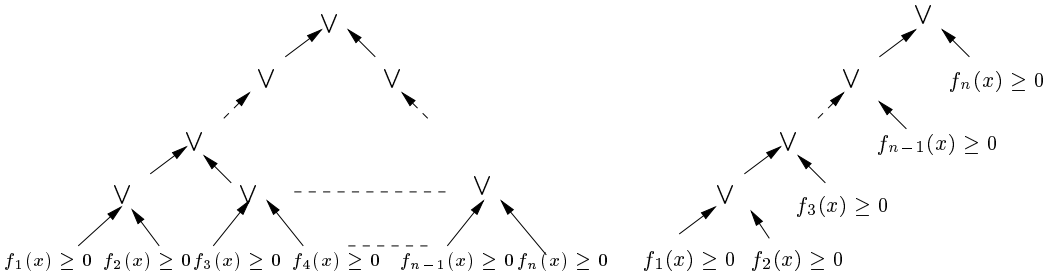
$$\begin{cases} g(x) \geq \alpha \underline{g} \\ h(x) \geq (1 - \alpha) \underline{h} \end{cases}$$

where $\underline{g}$ and $\underline{h}$ are two known non null finite lower bounds for g and h resp.

We can also generalize to arbitrary number of constraints in a disjunctive formula $\vee_n$:

$$\vee_n(f_1, \cdots, f_n) = (f_1(x) \geq 0 \ \vee \ f_2(x) \geq 0 \vee \cdots \vee \ f_n(x) \geq 0)$$

Since the dichotomy operator $\vee$ is associative, we group the constraints two by two using a binary tree. We can either express $\vee_n$ by grouping the constraints using a perfect binary tree as shown in Fig. 3.(a), or using a left associative binary tree as shown in Fig. 3.(b). With both techniques, there is $(n-1)$ internal



(a) Perfect Binary Tree

(b) Left Associative Binary Tree

Figure 3: Expressing an n-Disjunction with Linear Constraints

$\vee$ operators which need to define $(n-1)$ boolean variables $(h_1, \cdots, h_{n-1})$. The final constraints system to express $\vee_n$ has $\mathcal{O}(n)$ constraints $(f_1, \cdots, f_n)$ and $\mathcal{O}(n-1)$ boolean binary variables $(h_1, \cdots, h_{n-1})$. The non null lower bounds of the linear functions are always finite. They always can be computed statically and propagated up in the binary tree [16].

From above, we can deduce the linear constraints of any other logical operator:

1. $g(x) \geq 0 \Longrightarrow h(x) \geq 0$ can be written $g(x) < 0 \vee h(x) \geq 0$
2. $g(x) \geq 0 \Longleftrightarrow h(x) \geq 0$ can be written

$$\big(g(x) \geq 0 \wedge h(x) \geq 0\big) \vee \big(h(x) < 0 \wedge g(x) < 0\big)$$

5.2 Computing the Maximum with Linear Constraints

In our intLP formulation, we need to compute the function $z = max(x, y)$ which can formulated by considering the following constraints:

$$\begin{cases} z \geq x \\ z \geq y \\ z \leq (1-\alpha)x + \alpha\bar{y} \\ z \leq \alpha y + (1-\alpha)\bar{x} \\ \alpha \in \{0, 1\} \end{cases}$$

where $(\bar{x}, \bar{y})$ are two finite non null upper bounds for x, y resp. We can also express the max_n function with arbitrary number of parameters $z = max_n(x_1, x_2, \cdots, x_n)$. Since max is associative, we use a binary tree like explained for the n-disjunction operator in Fig. 3. The number of internal nodes including the root is equal to $n-1$, so we need to define $n-2$ intermediate variables (that hold intermediate maximums) and $(n-1)$ systems to compute"max" operators. It leads to a complexity of $\mathcal{O}(n-2) = \mathcal{O}(n)$ intermediate variables and $\mathcal{O}(4 \times (n-1)) = \mathcal{O}(n)$ linear constraints (each "max" operator needs 4 linear constraints to be defined)

and $\mathcal{O}(n-1) = \mathcal{O}(n)$ binary variables (each max operator needs 1 boolean). The non null upper bounds of the linear functions are always finite if the domain sets of the variables x_i is bounded [16].

6 EquiMax Integer Programming Formulation

In this section, we define a new formulation of scheduling problem using integer linear programming (intLP). We named it EquiMax because it uses the linear constraints which express the equivalence relation ($\Longleftrightarrow$) and the function max_n.

6.1 Scheduling Variables and Objective Function

For all operations $u \in V$, we define the integer variable σ_u that computes the schedule time. The objective function of our model is to minimize the total schedule time i.e. **minimize** $\sigma_\perp$.

The first linear constraints are those which describe the precedence relations, so we write in the model:

$$\forall e = (u, v) \in E \qquad \sigma_v - \sigma_u \geq \delta(e)$$

There is $\mathcal{O}(|V|)$ scheduling variables and $\mathcal{O}(|E|)$ linear constraints. To make the domains set of our variables bounded, we assume T as the worst possible schedule time. We chose T sufficiently large, where for instance $T = \sum_{u \in V} lat(u)$ is a suitable worst total schedule time[2]. Then, we write the following constraint:

$$\sigma_\perp \leq T$$

As consequence, we deduce for any $u \in V$:

- $\sigma_u \geq \underline{\sigma_u} = LonguestPathTo(u)$ is the "as soon as possible" schedule time;
- $\sigma_u \leq \overline{\sigma_u} = T - LonguestPathFrom(u)$ is the "as later as possible" schedule time according to the worst total schedule time T;

6.2 Registers Constraints

6.2.1 Interference Graph

The lifetime interval of a value u^t of type t is

$$LT_{u^t} =]\sigma_u + \delta_{w,t}(u), \max_{v \in Cons(u^t)} \left(\sigma_v + \delta_{r,t}(v)\right)]$$

We define for each value u^t the variable k_{u^t} that computes its killing date. The number of such defined variables is $\mathcal{O}(|\mathcal{T}| \times |V_{R,t}|)$. Since the domain of our variables is bounded, we know that k_{u^t} is bounded by the two following finite schedule times:

$$\forall t \in \mathcal{T}\ \forall u^t \in V_{R,t} \qquad \underline{k_{u^t}} < k_{u^t} \leq \overline{k_{u^t}}$$

where

[2] The case where no ILP is exploited.

- $\underline{k_{u^t}} = \underline{\sigma_u} + \delta_{w,t}(u)$ is the first possible definition date of u^t;
- $\overline{k_{u^t}} = \max_{v \in Cons(u^t)} \left(\overline{\sigma_v} + \delta_{r,t}(v)\right)$ is the latest possible killing date of u^t.

We use the max_n linear constraints to compute k_{u^t} like explained in Sect. 5.2: we need to define for each k_{u^t} $\mathcal{O}(|Cons(u^t)|)$ variables and $\mathcal{O}(4 \times |Cons(u^t)|)$ linear constraints to compute it. The total complexity to define all killing dates for all registers types is bounded by $\mathcal{O}(|V|^2)$ variables and $\mathcal{O}(|V|^2)$ constraints.

Now, we can consider H_t the indirected interference graph of G for the register type t. For any couple of values of the same type $u^t, v^t \in V_{R,t}$, we define a binary variable $s^t_{u,v} \in \{0,1\}$ such that it is set to 1 if the two values lifetimes intervals interfere: $\forall t \in \mathcal{T}$, $\forall$ couple $u^t, v^t \in V_{R,t}$

$$s^t_{u,v} = \begin{cases} 1 & \text{if } LT_{u^t} \cap LT_{v^t} \neq \phi \\ 0 & \text{otherwise} \end{cases}$$

For any registers type $t \in \mathcal{T}$, the number of variables $s^t_{u,v}$ is the number of combinations of 2 values among $|V_{R,t}|$ i.e. $\left(|V_{R,t}| \times (|V_{R,t}| - 1)\right)/2$.

$LT_{u^t} \cap LT_{v^t} = \phi$ means that one of the two lifetime intervals is "before" the other, i.e. $LT_{u^t} \prec LT_{v^t} \vee LT_{v^t} \prec LT_{u^t}$ where $\prec$ denotes is the precedence operator ("before") in interval algebra [12]. Then, we have to express:

$$s^t_{u,v} = 1 \iff \neg\left(LT_{u^t} \prec LT_{v^t} \vee LT_{v^t} \prec LT_{u^t}\right)$$

Since $s^t_{u,v} \in \{0,1\}$, these constraints are equivalent to:

$$s^t_{u,v} \geq 1 \iff \begin{cases} k_{u^t} - \sigma_v - \delta_{w,t}(v) - 1 \geq 0 \\ k_{v^t} - \sigma_u - \delta_{w,t}(u) - 1 \geq 0 \end{cases}$$

Given three logical expressions (P, Q, S), $(P \iff (Q \wedge S))$ is equivalent to $(P \wedge Q \wedge S) \vee (\neg P \wedge \neg Q) \vee (\neg P \wedge \neg S)$. We write these two disjunctions with linear constraints by introducing two binary variables $h, h' \in \{0,1\}$ (see Sect. 5) and computing the finite non null lower bounds of the linear functions. This leads to write in the model: $\forall t \in \mathcal{T}$, $\forall$ couple $u^t, v^t \in V_{R,t}$

$$\begin{cases} s^t_{u,v} + h + h' - 1 \geq 0 \\ k_{u^t} - \sigma_v - \delta_{w,t}(v) - min(-1, \underline{k_{u^t}} - \overline{\sigma_v} - \delta_{w,t}(v) - 1) \times (h + h') - 1 \geq 0 \\ k_{v^t} - \sigma_u - \delta_{w,t}(u) - min(-1, \underline{k_{v^t}} - \overline{\sigma_u} - \delta_{w,t}(u) - 1) \times (h + h') - 1 \geq 0 \\ \\ -s^t_{u,v} - h + h' + 1 \geq 0 \\ -k_u + \sigma_v + \delta_w(v) + min(-1, -\overline{k_{u^t}} + \underline{\sigma_v} + \delta_{w,t}(v)) \times (h - h' - 1) \geq 0 \\ \\ -s^t_{u,v} - h' + 1 \geq 0 \\ -k_{v^t} + \sigma_u + \delta_w(u) + min(-1, -\overline{k_{v^t}} + \underline{\sigma_u} + \delta_{w,t}(u)) \times (h' - 1) \geq 0 \\ h, h' \in \{0,1\} \end{cases}$$

The complexity of computing all the $s^t_{u,v}$ variables is $\mathcal{O}\big(|V_{R,t}| \times (|V_{R,t}|-1)\big)$ binary variables (two booleans for each couple of values (u^t, v^t)) and $\mathcal{O}\big(7/2|V_{R,t}| \times (|V_{R,t}|-1)|\big)$ linear constraints (7 linear constraints for each couple of values). The total complexity of considering the interference graph H_t is then bounded by $\mathcal{O}(|V_{R,t}|^2)$ variables and $\mathcal{O}(|V_{R,t}|^2)$ constraints.

6.2.2 Maximal Clique in the Interference Graph

The maximum number of values of type t simultaneously alive corresponds to a maximal clique in $H_t = (V_{R,t}, \mathcal{E}_t)$, where $(u^t, v^t) \in \mathcal{E}_t$ iff their lifetime intervals interfere ($s^t_{u,v} = 1$). For simplicity, rather to to handle the interference graph itself, we prefer considering its complementary graph $H'_t = (V_{R,t}, \mathcal{E}'_t)$ where $(u^t, v^t) \in \mathcal{E}'_t$ iff their lifetime intervals do *not* interfere ($s^t_{u,v} = 0$). Then, the maximum number of values of type t simultaneously alive corresponds to a maximal independent set[3] in H'_t.

To write the constraints which describe the independent sets (IS), we define a binary variable $x_{u^t} \in \{0,1\}$ for each value $x_{u^t} \in V_{R,t}$ such that $x_{u^t} = 1$ iff u^t belongs to an IS of H'_t. We must express in the model the following linear constraints:

$$\forall t \in \mathcal{T} \ \forall \text{ couple } x_{u^t}, x_{v^t} \in V_{R,t} \qquad x_{u^t} + x_{v^t} \leq 1 \iff s^t_{u,v} = 0$$

Since $s^t_{u,v} \in \{0,1\}$ and by using the linear expressions of the equivalence ($\iff$), we introduce a boolean $h \in \{0,1\}$ (see Sect. 5). The IS are defined in the intLP model by considering:

$$\begin{cases} -x_{u^t} - x_{v^t} + h + 1 \geq 0 \\ -s^t_{u,v} + h \geq 0 \\ x_{u^t} + x_{v^t} - 2h \geq 0 \\ s^t_{u,v} - h \geq 0 \\ h \in \{0,1\} \end{cases}$$

The number of the variables x_{u^t} is $\mathcal{O}(|V_{R,t}|)$. The number of introduced binary variables to express the equivalences is $\mathcal{O}(2 \times |V_{R,t}| \times (|V_{R,t}|-1))$. The number of linear constraints to define the IS is $\mathcal{O}(2 \times |V_{R,t}| \times (|V_{R,t}|-1))$.

The registers constraints are the fact that any set of values simultaneously alive of registers type t must not exceed the number of available registers $\mathcal{R}_t$. The maximal IS in H'_t is the maximal $\sum_{u^t \in V_{R,t}} x_{u^t}$. Thereby, we write in the model;

$$\forall t \in \mathcal{T} \qquad \sum_{u^t \in V_{R,t}} x_{u^t} \leq \mathcal{R}_t$$

There is $\mathcal{O}(|\mathcal{T}|) = \mathcal{O}(1)$ such constraints. The total complexity of computing the maximal independent sets in H'_t (maximal cliques in H_t) is then bounded by $\mathcal{O}(|V_{R,t}|^2)$ variables and $\mathcal{O}(|V_{R,t}|^2)$ constraints.

[3] It is a subgraph such that there is no two adjacent nodes.

6.3 Resources Constraints

6.3.1 Conflicting Graph

The resources constraints are handled by considering for each FU an indirected graph $F_q = (V, \mathcal{E}_q)$ which represents conflicts between the instructions on a FU $q \in Q$. For any couple of operations, $(u, v) \in \mathcal{E}_q$ iff u and v are in conflicts on q. Any clique in F_q represents the set of operations that use q at the same clock cycle. So, any clique must not exceed N_q the number of the FU copies.

We define a binary variable $f^q_{u,v} \in \{0,1\}$ such that $f^q_{u,v} = 1$ iff there is a conflict between u, v on the FU q. For each FU, there is $\mathcal{O}(1/2 \times |V| \times (|V|-1))$ f^q binary variables. To compute them, we use the reservation tables explained in Sect. 2. Having the RT of two operations types u and v, we can deduce when a structural hazards occurs on a FU q. For example, the operations a and i described in Fig. 2 have the RT of Fig. 1. These two operations are in conflict on the ALU iff $\sigma_a = \sigma_v \vee \sigma_a + 1 = \sigma_v$. The general formulation of conflicting variables is the disjunction of all cases where a conflict on the FU may occur.

Let $U_{u,q}$ be the set of clock cycles in the reservation table of u where the FU q is used by u:

$$\forall u \in V \ \forall q \in Q \qquad U_{u,q} = \{c \in \mathbb{N} / \mathcal{RT}_u[c,q] = 1\}$$

The set of all cases where two operations conflicts on a FU q are described by the cartesian product $U_{u,q} \times U_{v,q}$. The general formula of the binary conflicting variables is then: $\forall q \in Q$

$$\forall q \in Q \ \forall \text{ couple } u, v \in V \quad f^q_{u,v} = 1 \iff \bigvee_{(c1,c2) \in U_{u,q} \times U_{v,q}} \sigma_u + c1 = \sigma_v + c2$$

We use the linear constraints of equivalences and disjunctions defined in Sect. 5 to write the linear description of this formula in the model. The number of terms in this disjunction depends on $U_{u,q} \times U_{v,q}$ which is a function of the target architecture characteristics (reservation tables and instructions set), and thereby it is a constant for any input DDG. We can write the linear constraints of conflicting cases of all the couples of instructions in $\mathcal{IS}$ only once for the target architecture, and then instantiate them for any DDG. The total complexity of computing the conflicting variables f^q is bounded by $\mathcal{O}(|V|^2)$ variables and $\mathcal{O}(|V|^2)$ constraints.

6.3.2 Maximal Click in the Conflicting Graph

For simplicity, rather than considering the conflict graph F_q itself, we use its complementary $F'_q = (V, \mathcal{E}'_q)$ such that $(u, v) \in \mathcal{E}'_q$ iff u and v are *not* in conflicts on q ($f^q_{u,v} = 0$). Then, a clique in F_q becomes an independent set in F'_q.

We define a binary variable $y^q_u \in \{0,1\}$ for each operation u such that $y^q_u = 1$ iff u belongs to an IS of F'_q. We write in the intLP model the linear constraints of IS:

$$\forall q \in Q \ \forall \text{ couple } u, v \in V \qquad y^q_u + y^q_v \leq 1 \iff f^q_{u,v} = 0$$

Since $f^q_{u,v} \in \{0,1\}$ and by using the linear constraints of the equivalence (Sect. 5), we introduce a binary variable $h \in \{0,1\}$. These constraints become:

$$\begin{cases} -y^q_u - y^q_v + h + 1 \geq 0 \\ -f^q_{u,v} + h \geq 0 \\ y^q_u + y^q_v - 2h \geq 0 \\ f^q_{u,v} - h \geq 0 \\ h \in \{0,1\} \end{cases}$$

There is $\mathcal{O}(1/2 \times |V| \times (|V|-1))$ binary variables h for each FU (one for each couple of operations) and $\mathcal{O}(2 \times |V| \times (|V|-1))$ linear constraints to describe the IS. The resources constraints are the fact the cardinality of the any independent set in F'_q must not exceed N_q. We write in the model:

$$\forall q \in Q \qquad \sum_{u \in V} y^q_u \leq N_q$$

There is $\mathcal{O}(|Q|) = \mathcal{O}(1)$ such linear constraints.

6.4 Summary

Our integer LP model has a total complexity bounded by $\mathcal{O}(|V|^2)$ variables and $\mathcal{O}(|E| + |V|^2)$ constraints:

1. the objective function: **minimize** $\sigma_\perp$
2. the total number of integer variables is bounded by $\mathcal{O}(|V|^2)$:
   (a) $\mathcal{O}(|V|)$ scheduling variables: σ_u for each node $u \in V$;
   (b) $\mathcal{O}\big((|V_{R,t}| \times (|V_{R,t}|-1))/2\big)$ interference binary variables for each registers type t: $s^t_{u,v} \in \{0,1\}$ for all couples $u^t, v^t \in V_{R,t}$;
   (c) $\mathcal{O}(|V_{R,t}|)$ binary independent sets variables for the complementary interference graph H'_t of the register type t: $x_{u^t} \in \{0,1\}$ for each value $u^t \in V_{R,t}$;
   (d) $\mathcal{O}\big((|V| \times (|V|-1))/2\big)$ conflict binary variables for each FU q: $f^q_{u,v} \in \{0,1\}$ for all couples $u, v \in V$;
   (e) $\mathcal{O}(|V|)$ binary independent sets variables for the complementary conflict graph F'_q of each FU q: $y^q_u \in \{0,1\}$ for each operation $u \in V$;
   (f) the total number of intermediate and binary variables to write max_n, n-disjunctions and equivalence with linear constraints is bounded by $\mathcal{O}(|V|^2)$;
3. the total number of linear constraints is bounded by $\mathcal{O}(|E| + |V|^2)$:
   (a) $\mathcal{O}(|E|)$ scheduling constraints:

$$\forall e = (u,v) \in E \qquad \sigma_v - \sigma_u \geq \delta(e)$$

(b) the total number of interval lifetimes interference constraints is bounded $\mathcal{O}(|V_{R,t}|^2)$ for each register type t:

$$\forall t \in \mathcal{T} \qquad s^t_{u,v} = 1 \Longleftrightarrow \neg\big(LT_{u^t} \prec L_{v^t} \vee L_{v^t} \prec L_{u^t}\big)$$

(c) the total number of independent sets constraints for the complementary interference graph H'_t is bounded by $\mathcal{O}(|V_{R,t}|^2)$ for each register type t:

$$\forall t \in \mathcal{T} \qquad x_{u^t} + x_{v^t} \leq 1 \Longleftrightarrow s^t_{u,v} = 0$$

(d) the number of registers constraints is $\mathcal{O}(|\mathcal{T}|) = \mathcal{O}(1)$:

$$\forall t \in \mathcal{T} \qquad \sum_{u^t \in V_{R,t}} x_{u^t} \leq \mathcal{R}_t$$

(e) the total number of conflicting constraints is bounded by $\mathcal{O}(|V|^2)$ for each FU q:

$$\forall q \in Q \qquad f^q_{u,v} = 1 \Longleftrightarrow \bigvee_{(c1,c2)\in U_{u,q}\times U_{v,q}} \sigma_u + c1 = \sigma_v + c2$$

(f) the total number of independent sets constraints for the complementary conflict graph F'_q is bounded by $\mathcal{O}(|V|^2)$:

$$\forall q \in Q \qquad y_u + y_v \leq 1 \Longleftrightarrow f_{u,v} = 0$$

(g) the number of resources constraints is $\mathcal{O}(|Q|) = \mathcal{O}(1)$:

$$\forall q \in Q \qquad \sum_{u \in V} y_u \leq N_q$$

(h) the total number of linear constraints to express max_n, n-disjunctions and equivalence is bounded by $\mathcal{O}(|V|^2)$;

We can optimize the length of our model by considering;

- a precedence constraints $e = (u, v)$ is redundant and can be evicted from the model iff $lp(u, v) > \delta(e)$, where $lp(u, v)$ denotes the longest path from u to v;
- two values $(u^t, v^t) \in V_{R,t}$ can never be simultaneously alive iff for all possible schedules one value is always defined after the killing date of the other. This is the case if any of the two following conditions is verified:

$$\begin{array}{ll} \forall v' \in Cons(v^t) & lp(v', u) \geq \delta_r(v') - \delta_w(u) \\ \forall u' \in Cons(u^t) & lp(u', v) \geq \delta_r(u') - \delta_w(v) \end{array}$$

such that if no path exists between two nodes, we consider it as $-\infty$;

- two operations $u, v \in V$ can never conflict on a FU q iff they can never use q at the same clock cycle. This is the case if any of the two following conditions is verified :

$$\begin{array}{ll} \forall c \in U_{u,q}\ \forall c' \in U_{v,q} & lp(u,v) > c - c' \\ \forall c' \in U_{v,q}\ \forall c \in U_{u,q} & lp(v,u) > c' - c \end{array}$$

  such that if no path exists between two nodes, we consider it as $-\infty$.

7 Related Work

Acyclic scheduling under registers and resources constraints is a classical problem where lot of work has been done. An intLP formulation (SILP) was defined in [17] to compute an optimal schedule with register allocation under resources constraints. The complexity of this model is bounded by $\mathcal{O}(|V|^2)$ variables and $\mathcal{O}(|V|^2)$ constraints. However, this formulation does not introduce registers constraints, i.e. it does not limit the number of values simultaneously alive. Moreover, the resources usage patterns which they use was simple and do not formalize structural hazards that are present in most current ILP processors. A formulation, called OASIC, introduced registers constraints and was given in [8, 9]. The number of variables was $\mathcal{O}(|V|^2)$ but the number of linear constraints grown exponentially due to registers constraints. An extension of OASIC formulation was written in [11] to take into account non regular registers sets (some registers must not be used by some operations) and some other special constraints on ILP which are specific to their target processor characteristics. The registers constraints was formulated but not integrated in that model because of the exponential number of constraints to be generated.

Lot of work has also been done for cyclic scheduling problem (software pipelining) under registers and resources constraints. It is easy to rewrite these intLP models to solve acyclic scheduling problems. Hanen has written an original formulation to linearize disjunctive resources constraints in [10]. The drawback of her formulation is to treat only simple resources, i.e. an operation can execute only on a single FU. Feautrier in [6] has extended this latter to take into account multiple copies of one FU. However, his formulation has the same drawback as in [17] and does not treat complex and heterogeneous FUs. Cyclic scheduling under both registers and resources constraints has been formulated in [1, 4, 5]. All these formulations have a complexity which depends on a worst total schedule time T. Indeed, they define a binary variable $\sigma_{u,c}$ for each operation u and for each execution step c during the whole execution interval $[0, T]$. $\sigma_{u,c}$ is set to 1 iff the operation u is scheduled at the clock cycle c. The complexity of their models is clearly bounded by $\mathcal{O}(T \times |V|)$ variables and $\mathcal{O}(|E| + T \times |V|)$ constraints. The factor T can be very large since it depends on the input data itself (critical paths and specified operations latencies), and not depend on the *amount* of input data. For instance, if we are sure statically that the access to the memory performed by the operation a in Fig. 2 is a cache miss, then we

would specify that its latency is a memory access (~ 100) rather than a cache access in order to better exploit free slots during scheduling. In this case, the number of variables and constraints in the intLP model is multiplied by a factor of hundred.

The coefficients introduced by our formulation in the final constraints matrix are all bounded by T and $-T$, which is the case of the coefficients in the models defined in [1, 4, 5]. If T is very huge, resolving an EquiMax model or any of the previous formulations may cause computational overflows: in fact, searching for an exact solution of an intLP model needs to compute some determinants of the constraints matrix which can be very huge if the coefficients are sufficiently large [2]. Since EquiMax reduces the size of the constraints matrix, computing these determinants must be less critical with our formulation than with the previous techniques.

8 Conclusion

In this work, we give an intLP formulation of scheduling under resources and registers constraints. The FUs can have a complex usage pattern and are modeled by reservation tables. We handle multiple registers types and delayed read from and write into the registers. The complexity of our model depends only on the number of operations to be scheduled and on the number of serial constraints. Theoretically, our formulation should reduce considerably the time of finding the exact solution. In the future, we will extend our formulation to cyclic scheduling (software pipelining), where the values lifetime intervals and the resources usage patterns become cyclic.

References

[1] Eric Altman. *Optimal Software Pipelining with Functional Units and Registers.* PhD thesis, McGill University, Montreal, October 1995.

[2] William Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial optimization.* J. Wiley and sons, 1998.

[3] Alain Darte, Yves Robert, and Frédéric Vivien. *Scheduling and Automatic Parallelization.* Birkhauser Boston , 2000.

[4] Christine Eisenbeis, Franco Gasperoni, and Uwe Schwiegelshohn. Allocating Registers in Multiple Instruction-Issuing Processors. In Lubomir Bic and Wim Böhm and Paraskevas Evripidou and Jean-Luc Gaudiot, editor, *Proceedings of the IFIP WG 10.3 Working Conference on Parallel Architectures and Compilation Techniques, PACT'95*, pages 290–293, Limassol, Cyprus, June 27–29, 1995. ACM Press.

[5] Christine Eisenbeis and Antoine Sawaya. Optimal Loop Parallelization under Register Constraints. In *Sixth Workshop on Compilers for Parallel Computers CPC'96.* , pages 245–259, Aachen - Germany, December 1996.

[6] Paul Feautrier". Fine-Grain Scheduling under Resource Constraints. In *Proceedings of the 7th International Workshop on Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science, pages 1–15. Springer-Verlag, August 1994.

[7] Robert S. Garfinkel and George L. Nemhauser. *Integer Programming.* John Wiley & Sons, New York, 1972. Series in Decision and Control.

[8] C. H. Gebotys. Optimal Scheduling and Allocation of Embedded VLSI Chips. In *Proceedings of the 29th Conference on Design Automation*, pages 116–119, Los Alamitos, CA, USA, June 1992. IEEE Computer Society Press.

[9] C. H. Gebotys and M. I. Elmasry. A Global Optimization Approach for Architectural Synthesis. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, pages 258–261, Santa Clara, CA, November 1990. IEEE Computer Society Press.

[10] Claire Hanen. Study of NP-hard Cyclic Scheduling problem: The periodic recurrent job-shop. In *International Workshop on Compiler for Parallel Computers.* Ecole des Mines de Paris, December 1990.

[11] D. Kaestner and M. Langenbach. Code Optimization by Integer Linear Programming. *Lecture Notes in Computer Science*, 1575:122–136, 1999.

[12] Martin Charles Golumbic and Ron Shamir. Interval Graphs, Interval Orders and the Consistency of Temporal Events. In *Proceedings of Theory of Computing and Systems (ISTCS'92)*, volume 601 of *LNCS*, pages 32–42, Berlin, Germany, May 1992. Springer.

[13] R. Sethi. Complete register allocation problems. *SIAM Journal on Computing*, 4(3):226–248, 1975.

[14] Jurij Silc, Borut Bobic, and Theo Ungerer. *Processor Architecture: from Dataflow to Superscalar and Beyond.* Springer, première edition, 1999.

[15] M. Tokoro, E. Tamura, and T. Takizuka. Optimization of Microprograms. *IEEE Trans. on Computers*, C-30(7):491–504, 1981.

[16] Sid-Ahmed-Ali Touati. Optimal Register Saturation in Superscalar and VLIW Codes. Research Report, INRIA, October 2000. ftp.inria.fr/INRIA/Projects/a3/touati/optiRS.ps.gz.

[17] L. Zhang. *SILP: Scheduling and Register Allocation with Integer Linear Programming.* PhD thesis, University of Saarlands, 1996.