Minimization of circuit registers: retiming revisited

Bruno Gaujal, Jean Mairesse

Minimization of circuit registers: retiming revisited

Bruno Gaujal* Jean Mairesse†

December 2, 2005

Abstract

In this paper, we address the following problem: given a synchronous digital circuit, is it possible to construct a new circuit computing the same function as the original one but using a minimal number of registers? The construction of such a circuit can be done in polynomial time and is based on a result of Orlin for one periodic bi-infinite graphs showing that the cardinality maximum flow is equal to the size of a minimum cut. The idea is to view such a graph as the unfolding of the dependences in a digital circuit.

1 Introduction

Digital circuits can be seen as a rather accurate model of computer hardware. Their design and optimization is a long lasting challenge on several viewpoints. For instance the problems of layout compaction [4], verification [5], placement and partitioning [4] have been intensively studied in the VLSI literature, see for instance [8].

On an abstract point of view, synchronous digital circuits are often seen as finite graphs constituted by *functional gates*, *wires* and *registers*. At each clock tick, functional elements transform input data into output data which are transmitted on the wires to the next nodes. A register is a storage facility, or a memory cell, of finite size.

Several optimizations are possible at that level. One may want to accelerate the clock frequency ([7]) or reduce the number of nodes in the circuit. In this paper we show how to minimize the number of registers. This problem has already been considered by Leiserson and Saxe in a seminal paper [7], where they show how to retime (this will be defined later) the circuit in order to reduce the number of registers. They also provide an algorithm computing the best possible retiming. One question remains: is it possible to do better? In other words, can one modify a circuit so that the number of registers is smaller than what the optimal retiming does, while keeping the original functional behavior?

The answer is yes and no. For many circuits retiming is indeed optimal. In those where it is not, the gain in the number of registers comes at the expense of additional functional nodes.

*INRIA, Laboratoire ID-IMAG,CNRS,UJF,INPG, 51 Av. J. Kuntzmann, 38330 Montbonnot, France. Email: bruno.gaujal@imag.fr

†LIAFA, CNRS-Université Paris 7, Case 7014, 2 place Jussieu, 75251 Paris Cedex 05, France. Email: mairesse@liafa.jussieu.fr.

More precisely, the contribution of that paper is as follows: (*i*) we show that retiming is not always optimal; (*ii*) we provide an algorithm providing a circuit with the minimal number of registers; (*iii*) and we characterize those circuits where retiming is indeed optimal.

A preliminary paper on that subject [3] considered a particular class of circuits, namely recycled one. Here, general circuits are considered and the proof is different and based on a result of Orlin [9] on one-periodic bi-infinite graphs, which is a "max-flow min-cut" theorem. Such graphs have mainly been used in scheduling applications [10], Uniform Recurrence Equations or PRAM program loops [1, 6], where the weights on the arcs represent time and where the flow is a crucial quantity. Here instead, we view weights as memory resources and the cut becomes the central notion.

The paper is organized as follows. Section 2 contains basic definitions and notation and formulates the above-mentioned "max-flow min-cut" theorem. Section 3 discusses how the obtained results can be used in design of digital circuits to minimize the number of registers and Section 4 illustrates the complete construction over an example.

2 One-periodic bi-infinite graphs and Orlin's Theorem

The sets of nonnegative and non-positive integers are denoted by $\mathbb{Z}_+$ and $\mathbb{Z}_-$, respectively.

By a *bi-infinite graph* we mean an infinite directed graph $\mathcal{D} = (\mathcal{V}, \mathcal{A})$ with node set $\mathcal{V} = (V \times \mathbb{Z})$ and arc set $\mathcal{A} \subset \mathcal{V} \times \mathcal{V}$, assuming throughout that V is a finite set and that $\mathcal{D}$ is locally finite, i.e., each node is incident with a finite number of arcs.

For $u \in V$ and $i \in \mathbb{Z}$, the u-th *line* in $\mathcal{D}$ is formed by the nodes (u, j) for all $j \in \mathbb{Z}$, and the i-th *column* by the nodes (w, i) for all $w \in V$. For brevity, for a node $v = (u, i)$ and an integer k, the node $(u, i + k)$ is denoted by $v + k$, and similarly for a set of nodes. A set S of nodes is called *consecutive* (in each line) if $(u, i) \in S$ and $(u, i + k) \in S$ with $k > 0$ imply $(u, i + h) \in S$ for each $h = 0, \dots, k$.

A *path* in $\mathcal{D}$ is an alternating sequence P of (not necessarily distinct) nodes v_i $(i \in I)$ and arcs (v_i, v_{i+1}) (when $i + 1 \in I$). Here I is either $\{0, \dots, n\}$ for $n \in \mathbb{Z}_+$ (yielding a *finite path*) or $I = \mathbb{Z}_+$ or $I = \mathbb{Z}_-$ or $I = \mathbb{Z}$ (a *bi-infinite path*). We also use notation $\cdots \to v_i \to v_{i+1} \to \cdots$ for P and, depending on the context, may consider a path as a subgraph of $\mathcal{D}$. For two nodes u and v, we write $u \stackrel{*}{\to} v$ if there exists a (finite) path from u to v.

Two additional conditions on the bi-infinite graphs we deal with are imposed. A bi-infinite graph $\mathcal{D}$ is said to be *one-periodic* if
(OP) for any two nodes v and v' of $\mathcal{D}$, $(v, v') \in \mathcal{A}$ if and only if $(v + 1, v' + 1) \in \mathcal{A}$,
and *causal* if
(C) for any infinite path P indexed by $\mathbb{Z}_+$, the set $P \cap (V \times \mathbb{Z}_-)$ is finite.

Note that properties (OP) and (C) imply that $\mathcal{D}$ is acyclic.

The result presented below is analogous to the classical Menger theorem for usual finite graphs. A *flow* $\mathcal{F}$ is a set of pairwise (node-)disjoint bi-infinite paths. It is *one-periodic* for each path P in $\mathcal{F}$, the path $P + 1$ belongs to $\mathcal{F}$ as well. A *cut* a set of nodes that intersects every bi-infinite path. Clearly the size of a flow cannot exceed the size of a cut. Also if $\mathcal{D}$ is one-periodic and

causal, then the set $V \times \{0, \ldots, k\}$ forms a cut, where $k := \max\{|i-j| \mid ((u,i),(w,j)) \in \mathcal{A}\}$ (k is finite since $\mathcal{D}$ is locally finite and one-periodic). Therefore, the maximum cardinality of a flow is finite.

Theorem 2.1 (Orlin [9]). *Let $\mathcal{D}$ be a bi-infinite graph satisfying (OP) and (C). The maximum cardinality of a flow is equal to the minimum cardinality of a cut. Moreover, the maximum is attained by a one-periodic flow and the minimum is attained by a consecutive cut.*

Theorem 2.1 can be seen as a special case of Theorem 4 in [9]. Getting the above statement requires one transformation. Each node in $\mathcal{D}$ is replaced by a triple (node-arc-node) on the same column. Now, Theorem 4 in [9] with upper and lower capacities on all the arcs set to 1 and 0 respectively, is exactly Theorem 2.1.

An example illustrating a maximum one-periodic flow and a minimum consecutive cut is drawn in Figure 1.

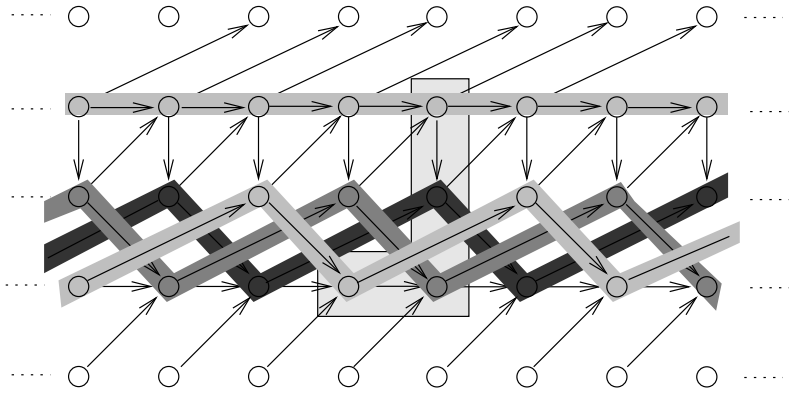


Figure 1: An illustration to Theorem 2.1.

A representation of $\mathcal{D}$ in a compact form called a is obtained by "folding" $\mathcal{D}$ into a finite arc-weighted digraph $G = (V, E, \Delta)$ with possible multiple arcs, as follows. Let us say that arcs $((u,i),(w,j))$ and $((u',i'),(w',j'))$ of $\mathcal{D}$ are *similar* if $u = u'$, $w = w'$, and $j - i = j' - i'$. Since $\mathcal{D}$ is locally finite and one-periodic, the number of classes under this similarity relation is finite, and each class, with a representative $((u,i),(w,j))$, generates one arc e in G with tail $t(e) := u$, head $h(e) := w$ and weight $\Delta(e) := j - i$. We call G the *folded graph* associated with $\mathcal{D}$ (it is called a dynamic graph by Orlin)

In Section 3 we use the reverse construction. Given a finite directed (multi)graph $\mathcal{R} = (V, E, \Delta)$ with $\Delta \in \mathbb{Z}^E$, its *unfolding* is the one-periodic bi-infinite graph $\mathcal{D} = (V \times \mathbb{Z}, \mathcal{A})$ in which $((u,i),(w,j)) \in \mathcal{A}$ if and only if there is an arc $e \in E$ with $t(e) = u$, $h(e) = w$ and $j - i = \Delta(e)$. We say that $\mathcal{R}$ is *causal* if $\mathcal{D}$ is such, i.e., if $\Delta(C) > 0$ for each cycle C of $\mathcal{R}$.

Splinters. Let $\mathcal{D}$ be causal and one-periodic. For a subset $S \subseteq \mathcal{V}$, define

$$\begin{aligned} \mathrm{succ}(S) &= \{v \in \mathcal{V}\backslash S \mid \exists u \in S, (u,v) \in \mathcal{A}\}, \quad \mathrm{pred}(S) = \{v \in \mathcal{V}\backslash S \mid \exists u \in S, (v,u) \in \mathcal{A}\} \\ \mathrm{succ}^*(S) &= \{v \in \mathcal{V}\backslash S \mid \exists u \in S, u \xrightarrow{*} v\}, \quad \mathrm{pred}^*(S) = \{v \in \mathcal{V}\backslash S \mid \exists u \in S, v \xrightarrow{*} u\}\,. \end{aligned}$$

We assume that each node of $\mathcal{D}$ is contained in an infinite path (indexed by $\mathbb{Z}_+$ or $\mathbb{Z}_-$). Then, given a consecutive cut C, one can partition the nodes into three sets in a natural way: the set $P(C)$ of nodes "before" C, the set $S(C)$ of nodes "after" C, and C itself. Note that the sets $\mathrm{pred}^*(C)$, C and $\mathrm{succ}^*(C)$ are pairwise disjoint because of causality but they need not cover the whole graph.

We extend $\mathrm{pred}^*(C)$ and $\mathrm{succ}^*(C)$ into the desired sets $P(C)$ and $S(C)$, respectively, by examining $\mathcal{D}$ line by line and acting as follows. Three cases are possible.

1. No bi-infinite path goes through line u but some infinite path indexed by $\mathbb{Z}_+$ does. Then there exists a path from u to C and $\mathrm{succ}^*(C) \cap u$ is empty. On u, we assign the nodes of $\mathrm{pred}^*(C)$ to $P(C)$, and the other nodes to $S(C)$.

2. No bi-infinite path goes through line u but some infinite path indexed by $\mathbb{Z}_-$ does. On u, we assign the nodes of $\mathrm{succ}^*(C)$ to $S(C)$, and the other nodes to $P(C)$.

3. There is a bi-infinite path intersecting u. Then $\mathrm{succ}^*(C), C$, and $\mathrm{pred}^*(C)$ do partition line u. We make $S(C)$ and $P(C)$ coinciding with $\mathrm{succ}^*(C)$ and with $\mathrm{pred}^*(C)$ on u, respectively.

It is easy to check that the sets $P(C)$ and $S(C)$ are consecutive, using that the graph $\mathcal{D}$ is one-periodic. Also one can see that $\mathrm{pred}(S(C)) = C$ and $\mathrm{succ}(P(C)) = C$.

The sets $P(C)$ and $S(C)$ are called, respectively, the *negative* and *positive splinters* associated with C. These sets are used in the next section.

3 Application to Register Minimization in Digital Circuits

In this section, the name *graph* stands for a finite directed multigraph with integer arc-weights, called *delays*.

A *digital circuit* is made of *gates* computing data according to boolean logical functions, *wires* connecting the gates and *memory registers* on the wires which are storing the data between two computation cycles. With a digital circuit, we associate the graph $\mathcal{R} = (V, E, \Delta)$, whose nodes, arcs, and delays correspond respectively to the gates, wires, and number of registers on the wires of the digital circuit. Since the number of registers has to be nonnegative, a specificity of the graph associated with a digital circuit is that it has only nonnegative delays. Also, for physical reasons, any cycle in the digital circuit should contain at least one register. The associated graph is therefore causal. So for our purpose, a digital circuit is simply a causal graph with nonnegative delays. The unfolding of the graph $\mathcal{R}$ can be viewed as the graph of the dependences between the computations performed by the digital circuit (the node (i, n) corresponds to the n-th computation at gate i). Our goal is to minimize the number of registers used in a digital circuit in a sense to be made precise below (problem **Min-Register**). A more thorough discussion of the relation between digital circuits and graphs is proposed in [7, 3].

3.1 Computations in digital circuits

Let $\mathcal{R} = (V, E, \Delta)$ be a digital circuit. Let $\mathcal{Q}$ be a finite set (corresponding to all the different values that one register can store) and let $\mathcal{F}$ be the set of functions from $\mathcal{Q}^k$ to $\mathcal{Q}$ for all $k \in \mathbb{Z}_+$.

A *specialization* σ of the digital circuit consists in mapping one function of $\mathcal{F}$ to each gate of $\mathcal{R}$:

$$\begin{array}{rccc} \sigma & : & V & \to & \mathcal{F} \\ & & u & \mapsto & F_u \end{array}$$

The function F_u attached to gate u must have as many arguments as u has input arcs. In particular, if u is a node with no predecessor, then F_u is a constant (since F_u is a function from $\mathcal{Q}^0$ to $\mathcal{Q}$).

On a wire e with $\Delta(e) > 0$, we denote the registers by $(e, 1), \ldots, (e, \Delta(e))$ where the ranking is performed according to the physical order of the registers on the wire. Let $M = \{(e, n), e \in E, 1 \leq n \leq \Delta(e)\}$ be the set of all the registers. An *initial condition* I assigns an initial value to each register of the digital circuit:

$$\begin{array}{rccc} I & : & M & \to & \mathcal{Q} \\ & & m & \mapsto & I(m) \end{array}$$

The *computation* of $(\mathcal{R}, \sigma, I)$ is the sequence $(x(u, n))_{u \in V, n \in \mathbb{Z}_+}$ where $x(u, n) \in \mathcal{Q}$ is the n-th value computed at gate u if the values stored initially in the registers have been set by I and if the functions computed at each gate are those given by σ. More formally, we have

$$x(u, n) = F_u[x(i(e_1), n - \Delta(e_1)), \ldots, x(i(e_k), n - \Delta(e_k))], \quad n \in \mathbb{Z}_+, u \in V, \tag{1}$$

where $e_1, \ldots, e_k$, are the arcs with terminal node u (listed according to some total order on E), and where, if $n - \Delta(e_j) < 0, \quad x(i(e_j), n - \Delta(e_j)) = I((e_j, \Delta(e_j) - n))$.

The *computational power* of a digital circuit $\mathcal{R}$ is defined as follows. For an arbitrary finite set $\mathcal{Q}$, the sequence $(x(u, n))_{u \in V, n \in \mathbb{Z}_+}$ of elements of $\mathcal{Q}$ is *computable* by $\mathcal{R}$ if there exists σ and I, such that the sequence is computed by $(\mathcal{R}, \sigma, I)$.

We say that a digital circuit $\mathcal{R}_2$ (nodes V_2) has a *larger computational power* than a digital circuit $\mathcal{R}_1$ (nodes V_1) if for an arbitrary finite set $\mathcal{Q}$ and for each specialization σ_1 of $\mathcal{R}_1$, there exists a specialization σ_2 of $\mathcal{R}_2$ and an injective mapping

$$\theta : V_1 \times \mathbb{Z}_+ \to V_2 \times \mathbb{Z}_+ , \tag{2}$$

such that for any initial condition I_1 for $\mathcal{R}_1$ there exists an initial condition I_2 for $\mathcal{R}_2$ such that the computation $(x(u, n))_{u \in V_1, n \in \mathbb{Z}_+}$ of $(\mathcal{R}_1, \sigma_1, I_1)$ and the computation $(y(u, n))_{u \in V_2, n \in \mathbb{Z}_+}$ of $(\mathcal{R}_2, \sigma_2, I_2)$ satisfy $\forall (u, n) \in V_1 \times \mathbb{Z}_+, \quad x(u, n) = y \circ \theta(u, n)$. Roughly speaking, this means that everything that can be computed by $\mathcal{R}_1$ can also be retrieved from a computation carried over $\mathcal{R}_2$. Obviously, this retrieval is efficient only if θ is simple enough.

Remark. In relation to the above point, observe that the computational power of a digital circuit only depends on its topology, namely V, E, and Δ. It is independent of $\mathcal{Q}$, σ, or I.

3.2 Forward splitting, duplication, and retiming

Given a causal graph $\mathcal{R} = (V, E, \Delta)$, possibly with some negative delays, we define the *total number of delays* of $\mathcal{R}$ as follows:

$$\Delta_A(\mathcal{R}) = \sum_{e \in E} \Delta(e) \,. \tag{3}$$

Another quantity of interest is the following one:

$$\Delta_B(\mathcal{R}) = \sum_{u \in V} \max_{e \in E, i(e)=u} \Delta(e) \,. \tag{4}$$

By causality, we have $\Delta_A(\mathcal{R}) \geq 0$ and $\Delta_B(\mathcal{R}) \geq 0$. Furthermore, as soon as $\mathcal{R}$ contains at least one cycle, we have $\Delta_A(\mathcal{R}) > 0$ and $\Delta_B(\mathcal{R}) > 0$. Since $\Delta_A(\mathcal{R}) = \sum_{u \in V} \sum_{e \in E, i(e)=u} \Delta(e)$, we have $\Delta_B(\mathcal{R}) \leq \Delta_A(\mathcal{R})$. Introducing the quantity Δ_B is relevant because of the following result.

Proposition 3.1 ([3]). *There exists a causal graph denoted $\varphi(\mathcal{R})$ (set of nodes $\varphi(V)$), with* nonnegative *delays, such that:*

(i) $\Delta_A(\varphi(\mathcal{R})) = \Delta_B(\mathcal{R})$;

(ii) $\varphi(\mathcal{R})$ has a larger computational power than $\mathcal{R}$. Furthermore, the map θ in (2) has the following form:

$$\theta : V \times \mathbb{Z} \to \varphi(V) \times \mathbb{Z},\ (v, n) \mapsto (\alpha(v), n + c_v) \,, \tag{5}$$

where $\alpha : V \to \varphi(V)$ is an injection and $c_v, v \in V$, are integer constants, which do not depend on the specialization and initial condition of $\mathcal{R}$.

In [3, Propositions 6.5 and 7.3], the result is proved for recycled graphs, but it is easy to see that the proof extends to the general case. The graph $\varphi(\mathcal{R})$ is obtained from $\mathcal{R}$ by *duplication* first and then *forward splitting*.

The duplication transformation is designed to get rid of negative delays without changing Δ_B. An example of the duplication transformation is given in Figure 2. By causality, all cycles have a positive delay; hence, by duplicating nodes successively, it is possible to remove all the negative delays.

Forward splitting was introduced in [7] under the name of "register sharing". Forward splitting operates on graphs with nonnegative delays, and transforms $\mathcal{R}$ into $\mathcal{R}'$ such that $\Delta_A(\mathcal{R}') = \Delta_B(\mathcal{R})$. An example of the forward splitting transformation is given in Figure 3.

Let us recall the classical notion of retiming [7]. A *retiming* is a function $r : V \to \mathbb{Z}$. Given a graph $\mathcal{R} = (V, E, \Delta)$, it specifies a new graph $\mathcal{R}_r$ and a new unfolded graph $\mathcal{D}_r$ as follows:

- $\mathcal{R}_r = (V, E, \Delta_r)$ with, for $e \in E$, $\Delta_r(e) = \Delta(e) + r(i(e)) - r(t(e))$;
- $\mathcal{D}_r = (V \times \mathbb{Z}, \mathcal{A}_r)$ is the unfolding of $\mathcal{R}_r$; that is $((i, n), (j, m)) \in \mathcal{A}_r \iff ((i, n + r(i)), (j, m + r(j))) \in \mathcal{A}$.

In the example of Figure 4, the new graphs $\mathcal{R}_r$ and $\mathcal{D}_r$ correspond to the retiming r defined by $r(1) = 1, r(2) = 1$ and $r(3) = 0$.

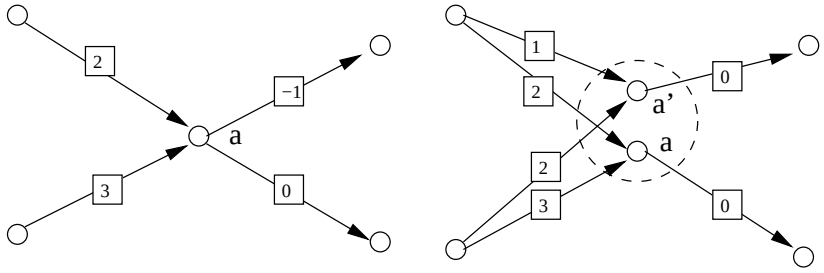
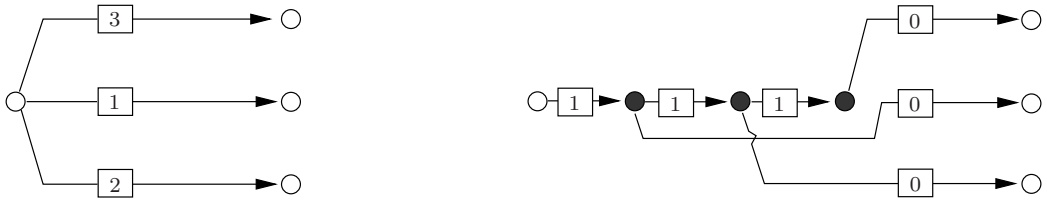


Figure 2: Duplication of node a



$\mathcal{R}:\ \Delta_A(\mathcal{R}) = 6,\ \Delta_B(\mathcal{R}) = 3$ $\qquad$ $\varphi(\mathcal{R}):\ \Delta_A(\varphi(\mathcal{R})) = \Delta_B(\varphi(\mathcal{R})) = 3$

Figure 3: The forward splitting transformation.

3.3 Solution to the Min-Register problem

We want to solve the **Min-Register** problem which is defined as follows:

Given a digital circuit $\mathcal{R}$, find another digital circuit with at least the same computational power and with as few registers as possible. This number will be denoted by Min-Reg($\mathcal{R}$).

Let $\mathcal{R} = (V, E, \Delta)$ be a digital circuit. Using Proposition 3.1, for any retiming r, the graph $\varphi(\mathcal{R}_r)$ has nonnegative delays and a larger computational power than $\mathcal{R}$. In particular it implies that:

$$\text{Min-Reg}(\mathcal{R}) \leq \min_r \Delta_B(\mathcal{R}_r) = \min_r \Delta_A(\varphi(\mathcal{R}_r))\ ,$$

where the minimum is taken over all possible retimings. The next theorem states that there is in fact equality.

Theorem 3.2. *Let $\mathcal{R}$ be a digital circuit and let $\mathcal{D}$ be its unfolding. Then the minimum number of registers* Min-Reg($\mathcal{R}$) *is equal to $\chi(\mathcal{D})$, the minimum cardinality of a cut of $\mathcal{D}$. Let C be a consecutive cut of $\mathcal{D}$ of minimum cardinality and let S be the corresponding positive splinter. For $i \in V$, let n_i be such that $(i, n_i - 1) \notin S, (i, n_i) \in S$. Let r be the retiming defined by $r(i) = n_i$. Then the digital circuit $\varphi(\mathcal{R}_r)$ is a solution to the* **Min-Register** *problem.*

Proof. We first prove that the number of registers of $\varphi(\mathcal{R}_r)$ is $\chi(\mathcal{D})$. Let f_r be the map from $V \times \mathbb{Z}$ into itself defined by $f_r((u, n)) = (u, n - r(u))$. Obviously, $f_r(S)$ is a positive splinter of $\mathcal{D}_r$. Furthermore, by definition of r, we have $f_r(S) = \{(u, n), u \in V, n \in \mathbb{Z}_+\}$. Now, the size of the cut $C = \text{pred}(S)$ in $\mathcal{D}$ is the same as the size of the cut $\text{pred}(f_r(S))$ in $\mathcal{D}_r$. Let us consider a node $u \in V$. Let $m = \max_{e \in E, i(e) = u} \Delta_r(e)$ and let v be such that there exists $e \in E$ with

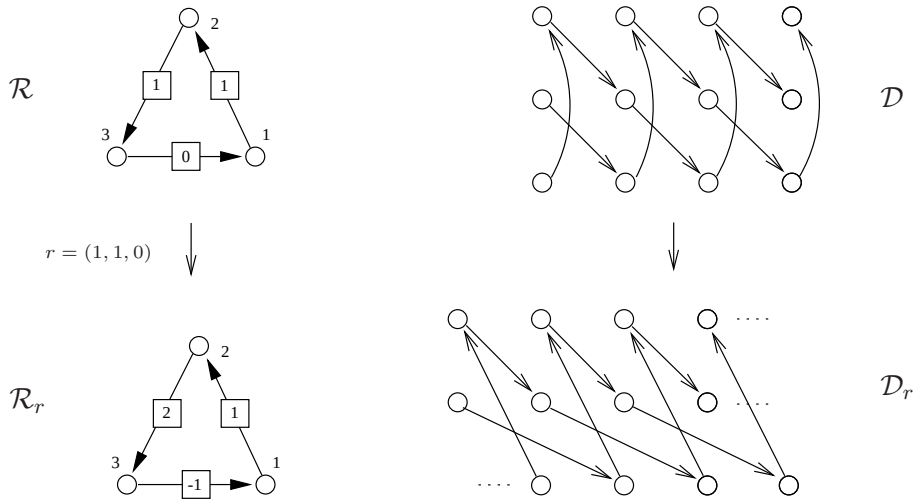


Figure 4: Retimed graph and its unfolding.

$i(e) = u, t(e) = v, \Delta(e) = m$. There is an arc in $\mathcal{D}_r$ from $(u, -m)$ to $(v, 0)$ and no arc from a node (u, k), $k < -m$ to a node $(w, \ell), \ell \geq 0$. Hence, by definition, $\text{pred}(f_r(S))$ contains exactly the nodes $(u, -m), \cdots, (u, -1)$ on line u. The same argument repeated on each line shows that $\Delta_B(\mathcal{R}_r) = |\text{pred}(S)| = |C| = \chi(\mathcal{D})$. As recalled above, the graph $\varphi(\mathcal{R}_r)$ has nonnegative delays, a larger computational power than $\mathcal{R}$ and satisfies $\Delta_A(\varphi(\mathcal{R}_r)) = \Delta_B(\mathcal{R}_r) = \chi(\mathcal{D})$.

In the second part of the proof we show that there exist no digital circuits with at least the same computational power as $\mathcal{R}$ and with strictly fewer registers than $\chi(\mathcal{D})$. Let $\mathcal{R}' = (V', E', \Delta')$ be a digital circuit with at least the same computational power as $\mathcal{R}$ and let $\mathcal{D}'$ be the unfolded graph associated with $\mathcal{R}'$.

According to Theorem 2.1, there exists in $\mathcal{D}$ a one-periodic flow of cardinality $|C|$ which defines a bijective mapping from the nodes of C to the ones of $C+L$, for any nonnegative integer L. Let us choose a specialization $\sigma : \ u \mapsto F_u$, in the following way. Consider $u \in V$ and let $e_1, \ldots, e_k$, be all the arcs in $\mathcal{R}$ with terminal node u (listed according to some total order on E). Let e_l be the only arc which corresponds to a set of arcs in $\mathcal{D}$ belonging to the flow. Then we define $F_u : \mathcal{Q}^k \to \mathcal{Q}$ by $F_u(x_1, \ldots, x_k) = x_l$. By composing the functions F_u, we get an application from the nodes of C to the ones of $C + L$ of the form $F : \mathcal{Q}^{|C|} \to \mathcal{Q}^{|C|}$ which is a permutation of the coordinates. In particular F is bijective.

For instance, consider Figure 1. Rank the nodes of the cut C in the order: $(2, k) < (3, k) < (4, k-1) < (4, k)$. For $L = 1$, the corresponding function is $F(x_1, x_2, x_3, x_4) = (x_1, x_3, x_4, x_2)$. For $L = 2$, the function is $F(x_1, x_2, x_3, x_4) = (x_1, x_4, x_2, x_3)$. For $L = 3$, the function F is the identity.

Observe that when we let the initial condition I vary over all the possible values in $\mathcal{Q}^{\Delta_A(\mathcal{R})}$ then the values of the computation of $(\mathcal{R}, \sigma, I)$ in the cut C, namely $(x(u, n))_{(u,n)\in C}$, cover all the values in $\mathcal{Q}^{|C|}$.

Since $\mathcal{R}'$ has a larger computational power than $\mathcal{R}$, there exists a specialization σ' of $\mathcal{R}'$ and an

injective mapping $\theta : V \times \mathbb{Z} \to V' \times \mathbb{Z}$ such that each sequence $(x(u,n))_{u\in V, n\in\mathbb{Z}_+}$ computed by $(\mathcal{R}, \sigma, I)$ for some initial condition I, is related to a sequence $(y(u,n))_{u\in V', n\in\mathbb{Z}_+}$ computed by $(\mathcal{R}', \sigma', I')$ for an adequate initial condition I', by $x(u,n) = y \circ \theta(u,n)$.

Let C' be a minimum consecutive cut of $\mathcal{D}'$. Let S' and P' be the positive and negative splinters associated with C'.

Let $\theta(C)$ be the image by θ of the cut C in $\mathcal{D}'$. Since θ is injective, we can assume by translating S' and by choosing L large enough, that $\theta(C) \subset P'$, and $\theta(C+L) \subset S'$. This means that $\theta(C)$ is on the "left" of C' and $\theta(C+L)$ is on the "right" of C'. Let U' be the subset of V' consisting of the nodes with no predecessor. By definition, given a specialization σ' of $\mathcal{R}'$, we have

$$\forall u \in U', \exists c_u \in \mathcal{Q}, \forall I', \forall n \in \mathbb{Z}_+,\ y(u,n) = c_u\ .$$

where $(y(u,n))_{u\in U', n\in\mathbb{Z}_+}$ are the values computed by $(\mathcal{R}', \sigma', I')$. We consider all the paths of $\mathcal{D}'$ ending in $\theta(C+L)$. Any such path intersects the cut C' or is finite and starts in a node of the type $(u,n), u \in U'$. Hence for any node $(w,n) \in \theta(C+L)$, we have $y(w,n) = G([c_u]_{u\in U'}; [y(v,k)]_{(v,k)\in C'})$ for some function G depending only on σ'. We conclude that for a fixed specialization of $\mathcal{R}'$, the variables $(y(u,n))_{(u,n)\in\theta(C+L)}$ can take at most $|\mathcal{Q}|^{|C'|}$ different values when the initial condition I' varies. Since F is bijective from $\mathcal{Q}^{|C|}$ into itself, it follows that $|C| \leq |C'|$. □

Proposition 3.3. *Let $\mathcal{R}$ be a digital circuit with n functional elements and m arcs. The circuit $\varphi(\mathcal{R}_r)$ can be constructed in $O(n(m + n\log n))$ units of time.*

Proof. The construction can be decomposed into several steps.
1. Computing the "max-flow min-cut" can be reduced to a maximum-weight perfect matching problem in a bipartite (undirected) graph and its dual one, which can be done with time complexity $O(n(m + n\log n))$; see, e.g., [2].
2. The retiming and forward splitting operations are local and take $O(m)$ units of time.
3. The duplication operation is also local. The number of duplications is bounded by $\sum_{e\in E} \Delta(e) = O(nm)$. □

Theorem 3.2 deserves several comments:

1. Given a digital circuit $\mathcal{R}$ with unfolding $\mathcal{D}$, the quantity $\chi(\mathcal{D})$ can be seen as the intrinsic quantity of memory needed to carry all the computations which could be wired by $\mathcal{R}$.

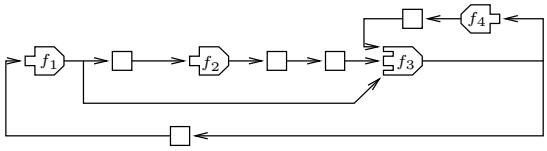
2. In the degenerated case where $\mathcal{R}$ is acyclic, the result *Min-Reg*$(\mathcal{R}) = \chi(\mathcal{D}) = 0$, clearly holds. Computing the relevant retiming is easy in this case.

3. The digital circuit $\mathcal{R}$ (nodes V) can be replaced by the digital circuit $\varphi(\mathcal{R}_r)$ (nodes $\varphi(V)$) without loss of computational power. It is however necessary to ask if the mapping $\theta : V \times \mathbb{Z}_+ \to \varphi(V) \times \mathbb{Z}_+$, defined in (2), is simple enough. Actually, it follows from (5) that the mapping θ is elementary.

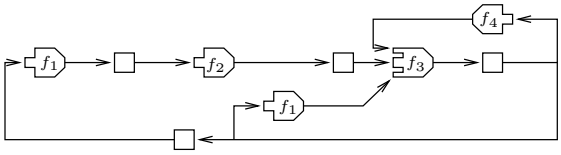
4 Solution for an example

In this section, we go through a small example to show how the construction works.

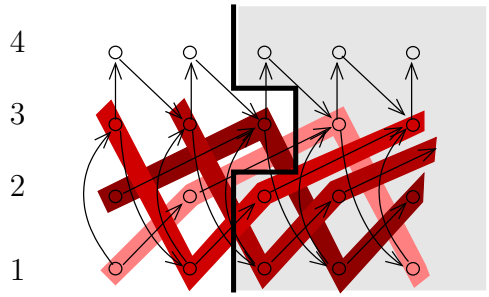
Let us consider the digital circuit displayed in Figure 4.a) Functional gates are represented by nodes with funny shapes and registers by boxes. The initial number of registers is 5. Here, retiming alone does not help to reduce the number of registers. Following our algorithm, the



a) A digital circuit with 5 registers



c) The new circuit has 4 registers and node 1 has been duplicated



b) The associated unfolding, its maximal flow and the corresponding positive splinter

construction of the associated unfolded graph is given in Figure 4.b)

The size of the maximal flow is 4. This means that one can find an equivalent circuit with 4 registers. The shape of the corresponding splinter gives the retiming to be applied. Duplicating node 1 finishes the construction of the circuit, displayed in Figure 4.c) which is equivalent to the initial circuit up to the bijection θ which maps the sequence of values computed in node 3 to the same sequence in the initial circuit, shifted by one.

Acknowledgement

The authors wish to thank Alexander V. Karzanov who provided us with a simple proof of Theorem 2.1 and an anonymous referee who pointed out the work of Orlin on dynamic graphs.

References

[1] V. Adlakha and V. Kulkarni. A classified bibliography of research on stochastic PERT networks: 1966-1987. *INFOR*, 27(3):272–296, 1989.

[2] R. Ahuja, T. Magnanti, and J. Orlin. *Network flows*. Prentice Hall Inc., 1993.

[3] B. Gaujal, A. Jean-Marie, and J. Mairesse. Computations of uniform recurrence equations using minimal memory size. *SIAM J. on Computing*, 30(5):1701–1738, 2000.

[4] Sabih H. Gerez. *Algorithms for VLSI Design Automation*. Wiley, 1999.

[5] Gary D. Hachtel and Fabio Somenzi. *Logic Synthesis and Verification Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[6] R. Karp, R. Miller, and S. Winograd. The organization of computations for uniform recurrence equations. *Journal of the ACM*, 14(3):563–590, 1967.

[7] C. Leiserson and J. Saxe. Retiming synchronous circuitry. *Algorithmica*, 1991.

[8] G. De Micheli. *Synthesis and Optimization of Digital Circuits*. Mc Graw Hill, 1994.

[9] J. B. Orlin. Maximum-throughput dynamic network flows. *Mathematical Programming*, 27:214–231, 1983.

[10] J. B. Orlin. Minimum convex cost dynamic network flows. *Mathematics of Operations Research*, 9(2):190–207, 1984.