Note

Minimization of circuit registers: Retiming revisited

Bruno Gaujal\textsuperscript{a,}\textsuperscript{*}, Jean Mairesse\textsuperscript{b}

\textsuperscript{a} INRIA, LIG (CNRS, UJF, INPG), 51 Av. J. Kuntzmann, 38330 Montbonnot, France
\textsuperscript{b} CNRS, LIAFA (Université Paris 7), Case 7014, 2 place Jussieu, 75251 Paris Cedex 05, France

A R T I C L E   I N F O

Article history:
Received 8 December 2005
Received in revised form 2 February 2008
Accepted 9 March 2008
Available online 28 April 2008

Keywords:
Retiming
Digital circuits
Register minimization
Periodic graphs
Cuts and flows

A B S T R A C T

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? We show that the minimal number of registers is the size of the minimal cut on a bi-infinite graph, namely the unfolding of the dependencies in the digital circuit. Furthermore, the construction of such a cut and the corresponding circuit can be done in polynomial time, using a max-flow min-cut result of Orlin for one-periodic bi-infinite graphs. Finally, we show the relation between this construction and the retiming technique introduced by Leiserson and Saxe.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Digital circuits can be seen as a rather accurate model of computer hardware. Their design and optimization is a long lasting challenge from several viewpoints. For instance, the problems of layout compaction [6], verification [7], placement and partitioning [6] have been intensively studied in the VLSI literature, see for instance [10].

From 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 [9] 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 [9], where they show how to ret ime (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 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.

More precisely, the contributions of the present paper are the following:

(i) We provide a characterization of the minimal number of registers as the size of a minimal cut in a graph. This implies that retiming is not always optimal.

(ii) We provide a polynomial algorithm to construct a circuit with the minimal number of registers.

(iii) We characterize the circuits in which retiming is indeed optimal.

A preliminary paper on that subject [5] considered a particular class of circuits, namely recycled ones. Here, general circuits are considered and the proof is different and based on a result of Orlin [11] on one-periodic bi-infinite graphs, which

* Corresponding author.

E-mail addresses: bruno.gaujal@imag.fr (B. Gaujal), mairesse@liafa.jussieu.fr (J. Mairesse).

0166-218X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2008.03.030
Theorem 2.1 discusses how the result can be used in the design of digital circuits to contain basic definitions and notations and formulates the above-mentioned “max-flow min-cut” theorem of Orlin. Section 3 discusses how the result can be used in the design of digital circuits to minimize the number of registers.

2. One-periodic bi-infinite graphs and Orlin’s theorem

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

By a bi-infinite graph we mean an infinite directed graph $\mathcal{D} = (V, A)$ with node set $V = (V \times \mathbb{Z})$ and arc set $A \subset V \times V$, 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 $h = 0, \ldots, k$.

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 $\{0, \ldots, n\}$ 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 $\cdots \rightarrow v_i \rightarrow v_{i+1} \rightarrow \cdots$ 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 $u \rightarrow v$ 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 $\mathcal{D}$, $(v, v') \in A$ if and only if $(v + 1, v' + 1) \in A$, and
- $(C)$ 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 $F$ is a set of pairwise (node-)disjoint bi-infinite paths. It is one-periodic if for each path $P$ in $F$, the path $P + 1$ belongs to $F$ as well. A cut is 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 $k := \max\{|i - j| \mid ((u, i), (w, j)) \in A\} (k$ is finite since $\mathcal{D}$ is locally finite and one-periodic). Therefore, the maximum cardinality of a flow is finite.

**Theorem 2.1** (Orlin [11]). 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 [11]. 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 [11] 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 Fig. 1.

A representation of $\mathcal{D}$ in a compact form is obtained by “folding” $\mathcal{D}$ into a finite arc-weighted digraph $\mathcal{R} = (V, E, \Delta)$ 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 $\mathcal{R}$ with tail $t(e) := u$, head $h(e) := w$, and weight $\Delta(e) := j - i$. We call $\mathcal{R}$ the folded graph associated with $\mathcal{D}$ (it is called a dynamic graph by Orlin). In Fig. 2, we have represented the folded graph associated with the one-periodic bi-infinite graph of Fig. 1.

![Fig. 1. An illustration to Theorem 2.1.](image-url)
In Section 3, we use the reverse construction. Given a finite directed (multi)graph \( R = (V, E, \Delta) \) with \( \Delta \in \mathbb{Z}^e \), its unfolding is the one-periodic bi-infinite graph \( D = (V \times \mathbb{Z}, A) \) in which \(((u, i), (v, j)) \in A\) if and only if there is an arc \( e \in E \) with \( t(e) = u, h(e) = v \) and \( j - i = \Delta(e) \).

We say that \( R \) is causal if \( D \) is causal. Clearly, \( R \) is causal if and only if \( \Delta(C) > 0 \) for each cycle \( C \) of \( R \).

Splinters. Let \( D \) be causal and one-periodic. For a subset \( S \subseteq V \), define

\[
\text{succ}(S) = \{v \in V \setminus S \mid \exists u \in S, (u, v) \in A\},
\]

\[
\text{pred}(S) = \{v \in V \setminus S \mid \exists u \in S, (v, u) \in A\},
\]

\[
\text{succ}^*(S) = \{v \in V \setminus S \mid \exists u \in S, u \rightarrow v\},
\]

\[
\text{pred}^*(S) = \{v \in V \setminus S \mid \exists u \in S, v \rightarrow u\}.
\]

We assume that each node of \( D \) 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 \( \text{pred}^*(C) \), \( C \) and \( \text{succ}^*(C) \) are pairwise disjoint because of causality but they need not cover the whole graph. We extend \( \text{pred}^*(C) \) and \( \text{succ}^*(C) \) into the desired sets \( P(C) \) and \( S(C) \), respectively, by examining \( D \) 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 \( \text{succ}^*(C) \cap u \) is empty. On \( u \), we assign the nodes of \( \text{pred}^*(C) \) 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 \( \text{succ}^*(C) \) to \( S(C) \), and the other nodes to \( P(C) \).

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

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

The sets \( P(C) \) and \( S(C) \) are called, respectively, the negative and positive splinters associated with \( C \) (see [3]). An example is provided in Fig. 3. This partition is used in the next section.

3. 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 \( R = (V, E, \Delta) \), whose nodes, arcs, and delays correspond respectively to the gates, wires, and number of registers.
3.1. Computations in digital circuits

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

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

$$\sigma : V \rightarrow F$$

$$u \mapsto F_u.$$ 

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 $Q^0$ to $Q$).

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:

$$i : M \rightarrow Q$$

$$m \mapsto i(m).$$

The computation of $(\mathcal{R}, \sigma, i)$ is the sequence $(x(u, n))_{u \in V, n \in \mathbb{Z}_+}$, where $x(u, n) \in Q$ 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 $\sigma$. More formally, we have

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

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

The computational power of a digital circuit $\mathcal{R}$ is defined as follows. For an arbitrary finite set $Q$, the sequence $(x(u, n))_{u \in V, n \in \mathbb{Z}_+}$, of elements of $Q$ is computable by $\mathcal{R}$ if there exists $\sigma$ 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 $Q$ and for each specialization $\sigma_1$ of $\mathcal{R}_1$, there exists a specialization $\sigma_2$ of $\mathcal{R}_2$ and an injective mapping

$$\theta : V_1 \times \mathbb{Z}_+ \rightarrow V_2 \times \mathbb{Z}_+,$$

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 $\forall (u, n) \in V_1 \times \mathbb{Z}_+, x(u, n) = y \circ \theta(u, n)$. 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 $\theta$ 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 $\Delta$. It is independent of $Q, \sigma,$ or $i$.

3.2. Duplication and forward splitting

As mentioned before, the graphs corresponding to digital circuits only have non-negatives delays. However, it is essential for the following sections to work on causal graphs that may have some negative delays.

Consider 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:

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

Another quantity of interest is the following one:

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

In words, $\Delta_B(\mathcal{R})$ only sums the maximum delays on the output arcs of all nodes.

The definition of $\Delta_A(\mathcal{R})$ and $\Delta_B(\mathcal{R})$ is illustrated in the upcoming Fig. 5. 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 $\Delta_A(\mathcal{R}) = \sum_{u \in V} \sum_{e \in E, \theta(e) = u} \Delta(e)$, we have $\Delta_A(\mathcal{R}) \leq \Delta_B(\mathcal{R})$.

Introducing the quantity $\Delta_B$ is relevant because of the following result.
Proposition 3.1 ([5]). There exists a causal graph denoted $\varphi(\mathcal{R})$ (set of nodes $\varphi(V)$), with non-negative 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 $\theta$ in (2) has the following form:

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

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

Sketch of the proof. In [5, Propositions 6.5 and 7.3], this result is proved for recycled graphs. It is straightforward to see that the proof extends to the general case. The transformation $\varphi$ can be decomposed into two elementary operations on $\mathcal{R}$.

The first one is a duplication of some nodes and the second one a forward splitting of arcs.

The duplication can be described by the following algorithm.

- While the graph contains an arc $(v, w)$ with a negative delay $-d$;
  1. duplicate the input node $v$ into two nodes $v$ and $v'$;
  2. remove arc $(v, w)$ and create a new arc $(v', w)$ with delay $0$.
  3. for each input arc $(u, v)$ of $v$ (with delay $h$), create a new arc $(u, v')$ with delay $h - d$.

An example of the local duplication transformation of one node is given in Fig. 4.

Note that after one duplication, new cycles may be created in the graph. However, the total delay on each new cycle is the same as on the corresponding original cycle. Since the original graph is causal, all its cycles have positive delays and this is also the case in the new graph. By a precise analysis of the circulation of negative delays, one can show that the algorithm must stop after a finite number of duplications. Finally, note that the algorithm removes all negative delays and does not change the value of $\Delta_B$.

As for forward splitting, it was introduced in [9] under the name of “register sharing”. Forward splitting operates on graphs with non-negative delays.

The forward splitting can be described by the following algorithm.

- For each node $v$ in the graph
  1. replace all output arcs $(v, w_1), \ldots, (v, w_k)$ with sorted delays $d_1 \geq \cdots \geq d_k \geq 0$ with a path of length $d_1$ over new nodes $v_0 (=v) \rightarrow v_1 \cdots \rightarrow v_{d_1}$ where all arcs have delay $1$;
  2. for all nodes $w_1, \ldots, w_k$ create an arc $(v_{d_k}, w_k)$ with weight $0$.

An example of the local forward splitting transformation on one node is given in Fig. 5. Note that forward splitting transforms $\mathcal{R}$ into a new graph $\mathcal{R}'$ such that $\Delta_A(\mathcal{R}') = \Delta_B(\mathcal{R})$. □
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 and let us recall the classical notion of retiming [9]. A retiming is a function $r : V \rightarrow \mathbb{Z}$. 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 $e \in E$, $\Delta_r(e) = \Delta(e) + r(i(e)) - r(t(e))$;
- $\mathcal{D}_r = (V \times \mathbb{Z}, A_r)$ is the unfolding of $\mathcal{R}_r$; that is $((i, n), (j, m)) \in A_r \iff ((i, n + r(i)), (j, m + r(j))) \in A$.

In the example of Fig. 6, the new graphs $\mathcal{R}_r$ and $\mathcal{D}_r$ correspond to the retiming $r$ defined by $r(1) = 1, r(2) = 1$ and $r(3) = 0$.

Using Proposition 3.1, for any retiming $r$, the graph $\varphi(\mathcal{R}_r)$ has non-negative delays and a larger computational power than $\mathcal{R}$. In particular it implies that:

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

where the minimum is taken over all possible ret timings. 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 $\text{Min-Reg}(\mathcal{R})$ 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 $f_i((u, n)) = (u, n - r(u))$. Obviously, $f_i(S)$ is a positive splinter of $\mathcal{D}_r$. Furthermore, by definition of $r$, we have $f_i(S) = \{(u, n), u \in V, n \in \mathbb{Z}_+, \}$ Now, the size of the cut $C = \text{pred}(S)$ in $\mathcal{D}$ is the same as the size of the cut $\text{pred}(f_i(S))$ in $\mathcal{D}_r$. Let us consider a node $u \in V$. Let $m = \max_{e \in E, t(e) = u} \Delta_r(e)$ and let $v$ be such that there exists $e \in E$ with $t(e) = u$, $t(e) = v$, $\Delta_r(e) = m$. 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, $\text{pred}(f_i(S))$ contains exactly the nodes $(u, -m), \ldots, (u, -1)$ on line $u$. The same argument repeated on each line shows that $\Delta_r(\mathcal{R}_r) = |\text{pred}(S)| = |C| = \chi(\mathcal{D})$. As recalled above, the graph $\varphi(\mathcal{R}_r)$ has non-negative delays, a larger computational power than $\mathcal{R}$ and satisfies $\Delta_r(\varphi(\mathcal{R}_r)) = \Delta_r(\mathcal{R}_r) = \chi(\mathcal{D})$.

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 $\mathcal{C}$ to the ones of $\mathcal{C} + L$, for any non-negative 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 $e_i$ be the only arc which corresponds to a set of arcs in $\mathcal{D}$ belonging to the flow. Then we define $F_u : \mathbb{Q}^k \rightarrow \mathbb{Q}$ by $F_u(x_1, \ldots, x_k) = x_k$. By composing the functions $F_u$, we get an application from the nodes of $\mathcal{C}$ to the ones of $\mathcal{C} + L$ of the form $F : \mathbb{Q}^{|C|} \rightarrow \mathbb{Q}^{|C|}$ which is a permutation of the coordinates. In particular $F$ is bijective.
For instance, consider Fig. 1. Rank the nodes of the cut C in the order: (2, k) < (3, k) < (4, k) < (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 \( Q^{\text{cut}(R)} \) then the values of the computation of \( (R, \sigma, I) \) in the cut C, namely \( (x(u, n))_{u,n \in C} \), cover all the values in \( Q^C \).

Since \( R' \) has a larger computational power than \( R \), there exists a specialization \( \sigma' \) of \( R' \) and an injective mapping \( \theta : V \times Z \rightarrow V' \times Z \) such that each sequence \( (x(u, n))_{u \in V, n \in Z} \) computed by \( (R, \sigma, I) \) for some initial condition I, is related to a sequence \( (y(u, n))_{u \in V', n \in Z} \) computed by \( (R', \sigma', I') \) for an adequate initial condition I', by \( x(u, n) = y \circ \theta(u, n) \).

Let \( C' \) be a minimum consecutive cut of \( D' \). Let \( S' \) and \( P' \) be the positive and negative splinters associated with \( C' \).

Let \( \theta(C) \) be the image by \( \theta \) of the cut C in \( D' \). Since \( \theta \) 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 \( \sigma' \) of \( R' \), we have

\[ \forall u \in U', \exists c_u \in Q, \ni \forall n \in Z_+, y(u, n) = c_u. \]

where \( (y(u, n))_{u \in U', n \in Z} \) are the values computed by \( (R', \sigma', I') \). We consider all the paths of \( D' \) 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 \( y(w, n) = G[(c_u)_{u \in U'} ; y(v, k)_{v \in \theta(C + L)}] \) for some function G depending only on \( \sigma' \). We conclude that for a fixed specialization of \( R' \), the variables \( (y(u, n))_{(u, n) \in \theta(C + L)} \) can take at most \( |Q^C| \) different values when the initial condition I varies. Since \( F \) is bijective from \( Q^C \) into itself, it follows that \( |C| \leq |C'| \).

**Proposition 3.3.** Let \( R \) be a digital circuit with \( n \) functional elements and \( m \) arcs. The circuit \( \varphi(R_r) \) 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 \( O(m(n(m + n \log n))) \); 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 \( R \) with unfolding \( D \), the quantity \( \chi(D) \) can be seen as the intrinsic quantity of memory needed to carry all the computations which could be wired by \( R \).
2. In the degenerated case where \( R \) is acyclic, the result Min-Reg \( (R) = \chi(D) = 0 \), clearly holds. Computing the relevant retiming is easy in this case.
3. The digital circuit \( R \) (nodes \( V \)) can be replaced by the digital circuit \( \varphi(R_r) \) (nodes \( \varphi(V) \)) without loss of computational power. It is however necessary to ask if the mapping \( \theta : V \times Z_+ \rightarrow \varphi(V) \times Z_+ \), defined in (2), is simple enough. Actually, it follows from [5] that the mapping \( \theta \) is elementary.
4. To each minimal cut of the unfolding of \( R \) corresponds an associated retiming \( r \) of \( R \). If there exists such a retiming \( r \) for which the retimed graph \( R_r \), contains only non-negative delays, then duplication is useless and only forward splitting is necessary to get a circuit with a minimal number of registers. In that case, the optimal retiming technique given in [9] coincides with our construction. However if all retimed graph \( R_r \), contain negative delays, duplication is always necessary in our construction and the optimal retiming technique given in [9] provides a circuit with strictly more registers than \( \varphi(R_r) \).

3.4. An illustrating example

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

Let us consider the digital circuit displayed in Fig. 7(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 in reducing the number of registers.

Following our algorithm, the construction of the associated unfolded graph is given in Fig. 7(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 Fig. 7(c). The new circuit is equivalent to the original circuit, with the sequence of values computed in node 3 being shifted by one.

**Acknowledgements**

The authors wish to thank Alexander V. Karzanov who provided them with a simple proof of Theorem 2.1 and an anonymous referee who pointed out the work of Orlin on dynamic graphs.
Fig. 7. A circuit and its optimized version.

References