$\ell$ edges. Therefore the weight (with respect to $w(e)$) of this set of edges combined with the weight of the jump is at least $\ell + 1$. A similar argument holds for vertical jumps. Therefore,

$$w(e) \geq (\ell + 1) \cdot \| \{ e \in c : e \text{ is a jump} \} \|.$$  

The rest of the proof follows the proof of claim 3. □

Equation 1 holds for cycles in the two-dimensional case, and the rest of the proof of the theorem follows the proof of theorem 1. □
B Proof of theorem 2

In this appendix we prove theorem 2. The proof follows the proof of theorem 1, but is complicated due to the two-dimensional topology.

Theorem 2: Let \( \ell \) denote the number of duplicated rows and columns along the chip boundaries. Let \( m \) denote the number of registers required along interconnections between chips. Let \( d_{\text{max}} \) denote the maximum propagation delay of a combinational gate after partitioning a cell into combinational gates and registers. Let \( \varepsilon > 0 \). If \( \ell \geq m \cdot (1 + 1/\varepsilon) - 1 \), then the two dimensional array (after duplicating rows and columns) can be retimed so that (a) the array can operate with a clock period of \( 1 + \varepsilon + d_{\text{max}} \) time units, and (b) interconnections which enter a chip from a neighboring chip have at least \( m \) registers along them.

As in the proof of theorem 1, we consider the communication graph derived from the array after duplicating rows and columns and the weight function associated with it. The communication graph is derived from the circuit depicted in figure 5. The weight of all the edges interconnecting cells equals 1. Each cell is partitioned into a network of combinational gates and registers, and the weight of an edge within a cell equals the number of registers along it.

A “lower-bound” weight function, \( w(\cdot) \), is defined on the edges of the communication graph as follows: (1) The weight, \( w(e) \), of edges which cross from one chip to another equals \( m \). (2) The weight of all other edges, \( e = u \rightarrow v \), is defined by \( w(e) = d(u)/(1 + \varepsilon) \), where \( d(u) \) denotes the propagation delay of vertex \( u \).

The proof of claim 4 is similar to the proof of claim 2.

Claim 4: If a retiming function, \( r(\cdot) \), of the two-dimensional array after duplicating rows and columns satisfies: \( w_r(p) \geq \lfloor w(p) \rfloor \), for every path \( p \), then:

1. For every edge, \( e \), which crosses from one chip to another, \( w_r(e) \geq m \). Therefore, after retiming, the interconnections which connect neighboring chips have at least \( m \) registers along them.
2. If \( w_r(p) = 0 \), then the propagation delay, \( d(p) \), of the path \( p \) is at most \( 1 + \varepsilon + d_{\text{max}} \) time units. Therefore, the retimed circuit can operate with a clock period of \( 1 + \varepsilon + d_{\text{max}} \) time units.

According to [EL91, lemma 2.1], a retiming which satisfies \( w_r(p) \geq \lfloor w(p) \rfloor \), for every path \( p \), exists if and only if \( w(e) \geq w_r(e) \), for every simple cycle \( e \). It is, therefore, sufficient to show, that if \( \ell \geq m \cdot (1 + 1/\varepsilon) - 1 \), then \( w(e) \geq w_r(e) \), for every simple cycle \( e \).

Claim 5: Let \( c \) denote a simple cycle in the two-dimensional array after duplicating rows and columns. If \( \ell \geq m \cdot (1 + 1/\varepsilon) - 1 \), then:

\[
w(e) \geq \frac{w(e)}{1 + \varepsilon} + m \cdot \| \{ e \in c : e \text{ is a jump } \} \|.
\]

Proof: We differentiate between vertical jumps and horizontal jumps. Similarly, we differentiate between horizontal and vertical edges which connect cells on the same chip.

Consider a horizontal jump in cycle \( c \) and the set of horizontal edges connecting cells which follow this jump until the next horizontal jump in \( c \). This set of horizontal edges contains at least
Claim 2: If \( r() \) is a retiming function of the linear array after duplicating segments which satisfies: \( w_r(p) \geq \lfloor w(p) \rfloor \), for every path \( p \), then:

1. Let \( e \) denote an edge which crosses between chips. \( w_r(e) \geq m \), and therefore, after retiming, the interconnection which enters the segment from the neighboring chip has at least \( m \) registers along it.

2. If \( w_r(p) = 0 \), then the propagation delay, \( d(p) \), of the path \( p \) is at most \( 1 + \varepsilon + d_{\max} \) time units. Therefore, the retimed circuit can operate with a clock period of \( 1 + \varepsilon + d_{\max} \) time units.

Proof: Part (1) follows immediately from the definition of \( w(e) \).

Let \( p \) be a path for which \( w_r(p) = 0 \) which ends in vertex \( v \). Since \( w_r(p) \geq \lfloor w(p) \rfloor \), it follows that \( \overline{w(p)} < 1 \). By the definition of \( w(p) \), it follows that \( d(p) - d(v) = \overline{w(p)} - (1 + \varepsilon) \). Since \( \overline{w(p)} < 1 \), and since \( d(v) \leq d_{\max} \), it follows that \( d(p) \leq 1 + \varepsilon + d_{\max} \), and the claim follows. \( \square \)

According to [EL91, lemma 2.1], a retiming which satisfies \( w_r(p) \geq \lfloor w(p) \rfloor \), for every path \( p \), exists if and only if \( w(e) \geq w(e) \), for every simple cycle \( e \). Therefore, it is sufficient to prove, that if \( \ell \geq m \cdot (1 + 1/\varepsilon) - 1 \), then \( w(e) \geq w(e) \), for every cycle \( e \).

Claim 3: Let \( e \) denote a simple cycle in the communication graph depicted in figure 2. Call an edge which interconnects cells on different chips a jump. If \( \ell \geq m \cdot (1 + 1/\varepsilon) - 1 \), then:

\[
w(e) \geq \frac{w(e)}{1 + \varepsilon} + m \cdot \| \{ e \in e : e \text{ is a jump} \} \|
\]

Proof: Every path from a jump to a jump passes through at least \( \ell + 1 \) cells. Therefore, \( \| \{ e \in e : e \text{ is a jump} \} \| \leq w(e)/(\ell + 1) \). It is sufficient to show, that if \( \ell \geq m \cdot (1 + 1/\varepsilon) - 1 \), then:

\[
w(e) \geq \frac{w(e)}{1 + \varepsilon} + m \cdot \frac{w(e)}{(\ell + 1)}
\]

This can be easily verified, and the claim follows. \( \square \)

Given a cycle, \( c \), partition cycle \( c \) into jump edges and non-jump edges. The sum of the weights, \( w(e) \), of jump edges equals \( m \cdot \| \{ e \in e : e \text{ is a jump} \} \| \). The sum of the weights, \( w(e) \), of non-jump edges is bounded by \( d(e)/(1 + \varepsilon) \). The delay between every two registers along \( e \) is bounded by 1 time unit. (Recall, that the circuit originally operated with a clock period of 1 time unit). Therefore, \( d(e) \leq w(e) \). Combining these observations, we obtain:

\[
w(e) \leq \frac{w(e)}{1 + \varepsilon} + m \cdot \| \{ e \in e : e \text{ is a jump} \} \|,
\]

Using claim 3 and equation 1, it follows that, if \( \ell \geq m \cdot (1 + 1/\varepsilon) - 1 \), then \( w(e) \geq w(e) \), for every simple cycle \( e \), and the theorem follows. \( \square \)

The retiming which satisfies \( w(p) \geq \lfloor w(p) \rfloor \), for every path \( p \), can be found by solving a single-source shortest-paths problem on the communication graph with the weight function defined by \( \hat{w}(e) \triangleq w(e) - w(e) \).
A Proof of theorem 1

In this appendix we prove theorem 1. The proof is based on an algorithm presented in work of Even and Litman [EL91]. Definitions of terms used in the proof (communication graphs, weight functions, etc.) are given in [LS91, EL91]. The theorem can also be proven using the technique and algorithm of Papaefthymiou [P93].

**Theorem 1:** Let \( \ell \) denote the number of cells in the duplicated segment of the linear array. Let \( m \) denote the number of registers required along interconnections between chips. Let \( d_{\text{max}} \) denote the maximum propagation delay of a combinational gate after partitioning a cell into combinational gates and registers. Let \( \varepsilon > 0 \). If \( \ell \geq m \cdot (1 + 1/\varepsilon) - 1 \), then the duplicated segments can be retimed so that (a) they can operate with a clock period of \( 1 + \varepsilon + d_{\text{max}} \) time units, and (b) interconnections which enter the segments from the neighboring chip have at least \( m \) registers along them.

Figure 2 depicts the communication graph of the linear array in the vicinity of the chip boundary. Note, that each cell is now considered as communication graph by itself, since the cells are partitioned into a network of combinational gates and registers. Define a weight function, \( w() \), on the edges, so that the weight of edge, \( e \), equals the number of registers along it.

A "lower-bound" weight function, \( \underline{w}() \), is defined on the edges of the communication graph as follows: (1) The weight of edges, \( e \), crossing between chips is \( \underline{w}(e) = m \). (2) The weight of all other edges, \( e = u \rightarrow v \), is defined by \( \underline{w}(e) = d(u)/(1 + \varepsilon) \), where \( d(u) \) denotes the propagation delay of vertex \( u \).

Let \( p \) denote a path in the communication graph, and let \( w(p) \) denote the sum of the weights of the edges along \( p \). Similarly, \( \underline{w}(p) \) denotes the sum of the lower-bound weights of the edges along \( p \). Let \( w_r(e) \) denote the number of registers along edge \( e \) after retiming the circuit by a retiming function \( r() \). Similarly, \( w_r(p) \) denotes the number of registers along path \( p \) after retiming the circuit by a retiming function \( r() \). Let \( d(p) \) denote the sum of the propagation delays of the vertices along the path \( p \).

First, we show that our goal is to find a retiming function, \( r() \), so that \( w_r(p) \geq \lfloor \underline{w}(p) \rfloor \), for every path \( p \). When this goal is achieved, the clock period is at most \( 1 + \varepsilon + d_{\text{max}} \) time units, and all interconnections which cross the chip boundary have at least \( m \) registers along them. The following claim summarizes this property.
Figure 6: Retiming the duplicated rows and columns \((m = 4)\) to obtain a clock period equal to 2.

Although the number of pins is usually a scarce resource, it is definitely not the case with systolic linear arrays. If longer clock periods are good enough, then the number of pins can be decreased by increasing the clock period. The reason for this is that the delay and clock skew are measured in clock periods.

6 Acknowledgements

The main question addressed in this paper was raised by Mike Paterson during a discussion with him concerning the applicability of a systolic priority queue even when it is too large to fit into a single chip. The discussion with Mike Paterson spawned the research, and we thank him for it. We would also like to thank Simon Even for helpful discussions.

References


and columns) can be retimed so that (a) the array can operate with a clock period of $1 + \varepsilon + d_{\max}$ time units, and (b) interconnections which enter a chip from a neighboring chip have at least $m$ registers along them.

5 Hardware overheads of the proposed solution

In this section we discuss the hardware overhead required by the solution proposed in this paper. The hardware overhead consists of three parts:

1. A segment of $O(d + s)$ cells is duplicated for every interconnection between cells, where $d$ denotes the delay of the interconnection and $s$ denotes the clock skew (these parameters are measured in clock periods).

2. An addition of an $n$-demultiplexer and $n$-multiplexer is required for every interconnection between chips, where $n = d + 2 \cdot s + 1$. These components can be realized with $O(k \cdot n)$ area, where $k$ denotes the breadth of the interconnection between chips.

3. An increase in the number of pins used for interconnections between chips is required. Our solution increases the number of pins needed for every interconnection between chips by a factor of $n + 1$, where $n = d + 2 \cdot s + 1$. 

Figure 5: Duplicating rows and columns and some interconnections.
neighboring chip. For example, the right outputs of cells in the 3rd column are connected to the left inputs of the cells in the 4th column both in the same chip and in the chip to the right. At this stage, all interconnections have exactly one register along them. It is important to note, that the functionality of the array is kept intact after this stage.

In the second stage, the duplicated rows and columns are retimed. After the retiming, all interconnections between chips have $m$ registers along them. The retiming is a “two-dimensional” version of the retiming described in subsection 3.2. Figure 6 depicts a retiming for the case $m = 4$ which enables a clock cycle of 2 time units (where the feasible clock period of the array when mapped onto a single chip equals 1 time unit). The duplication stage involved duplicating 6 rows and columns along the chip boundaries. In figure 6, each chip is divided into areas with identical lags, and the lags are shown in these areas. This retiming has two important properties: (a) There are no paths of length 2 which start at a cell with lag $i + 2$ and end in a cell of lag $i$. Therefore, the longest combinational (registerless) path consists of two cells, and each chip can operate with a clock cycle of 2 time units. (b) If an interconnection between chips emanates from a cell with lag $i$, then it enters a cell with lag $i + 3$. Therefore, 3 registers are added to all interconnections between chips, and they have 4 registers along them after the retiming.

Shorter clock periods can be obtained by considering the gate-level description of the cells, i.e., replace each cell with a network of combinational gates and registers. Theorem 2 states the tradeoff between the length of the duplicated segment and the clock period which can be obtained by retiming. The retiming, the existence of which is guaranteed by the theorem, can be found by solving a single-source shortest-paths problem.

Theorem 2: Let $\ell$ denote the number of duplicated rows and columns along the chip boundaries. Let $m$ denote the number of registers required along interconnections between chips. Let $d_{\text{max}}$ denote the maximum propagation delay of a combinational gate after partitioning a cell into combinational gates and registers. Let $\varepsilon > 0$. If $\ell \geq m \cdot (1 + 1/\varepsilon) - 1$, then the two dimensional array (after duplicating rows
maximum propagation delay of a combinational gate after partitioning a cell into combinational gates and registers. Let \( \varepsilon > 0 \). If \( \ell \geq m \cdot (1 + 1/\varepsilon) - 1 \), then the duplicated segments can be retimed so that (a) they can operate with a clock period of \( 1 + \varepsilon + d_{\text{max}} \) time units, and (b) interconnections which enter the segments from the neighboring chip have at least \( m \) registers along them.

### 4 Two dimensional arrays

In this section we consider a systolic two dimensional array which does not fit into one chip. Figure 4 depicts a segment of a systolic mesh-connected two dimensional array. The segment shows the cells in the vicinity of a boundary between four chips. As in section 3, a reduction is shown, which reduces the problem caused by interconnections between chips to the situations solved in section 2. Namely, the array is transformed into a functionally equivalent array, in which all interconnections between chips have several registers along them. Assume that the feasible clock period of the two-dimensional array when mapped onto a single chip equals 1 time unit. First, we show a solution which enables a clock period of 2 time units. Then, we show how one can obtain clock periods which are close to 1 time units, when the gate level description of the cells is considered.

The reduction consists of two stages (as in section 3): duplicating hardware and retiming. Let \( m = d + s \), where \( d \) denotes the delay of wires interconnecting chips and \( s \) denotes the clock skew between connected chips. In the first stage, \( O(m) \) rows and columns are duplicated along the chip boundaries. The interconnections between chips are set so that cells along the chip boundaries only receive inputs from other chips. The outputs of the cells along the chip boundaries which originally emanate from the chip are not connected. Figure 5 provides a diagrammatic description of the duplication stage: Indices of rows and columns are shown at the left and at the top. Six rows and six columns are duplicated along the chip borders. Therefore, rows 4–9 appear both in the top two chips and in the bottom two chips. Similarly, columns 4–9 appear twice. The interconnections within each chip are not modified, and are not shown in the figure. Interconnections between chips are modified. Figure 5 depicts only a small portion of interconnections between chips. However, all other interconnections between chips follow the same principle. Namely, an input to a cell on a chip boundary comes from the neighbor of the cell (taking into account the duplication) in the
3.1 Duplicating Segments

Duplicate a segment of $O(m)$ cells, so that a copy of the segment appears on each side of the boundary. The exact length of the duplicated segment is discussed in subsection 3.2. Connect the segments as shown in figure 2. Note that: (a) The rightmost output of the left duplicated segment and the leftmost output of the right duplicated segment are not connected. (b) The output of the right most cell to the left of the duplicated segments is split and sent to two cells. The same holds for the output of the left most cell to the right of the duplicated segments.

Clearly, the cells which are connected to the duplicated segments do not “notice” any change in functionality, and therefore, this step keeps functionality intact.

![Diagram of duplicating segments](image)

Figure 2: Duplicating segments.

3.2 Retiming The Linear Array

Our goal is to retime the linear array after duplicating segments so that the interconnecting between different chips have at least $m$ registers along them. When this goal is achieved, the implementation suggested in section 2 is applicable. Functionality of the circuit is kept intact due to the retiming lemma [LS81, EL91, E93].

First, we present a retiming which enables a clock period of 2 time units (where the feasible clock period of the array when mapped onto a single chip equals 1 time unit). The advantage of this retiming is that it does not require partitioning of the cells into “smaller” sub-components. A solution which enables a clock period equal to 2 time units is depicted in figure 3. Each segment consists of $2m - 3$ cells. The lags of the cells are written in the cells, and the number of registers along the wires are depicted in the figure.

Shorter clock periods can be obtained by considering the gate-level description of the cells, i.e., replace each cell with a network of combinational gates and registers. Theorem 1 states the tradeoff between the length of the duplicated segment and the clock period which can be obtained by retiming. The retiming, the existence of which is guaranteed by the theorem, can be found by solving a single-source shortest-paths problem.

**Theorem 1:** Let $\ell$ denote the number of cells in the duplicated segment of the linear array. Let $m$ denote the number of registers required along interconnections between chips. Let $d_{max}$ denote the
valid for \(2 \cdot s + 1\) clock cycles, it follows that the value read from the wire equals the value written to it, and the implementation behaves as required.

**Proof of part 3:** Consider a value written to wire \(e_{i \mod (d+2 \cdot s+1)}\) in the \(i\)th clock cycle of the demultiplexer. The multiplexer reads a value from this wire in its \(i+d+s\) clock cycle. The written value equals the read value provided that the elapsed time between writing it and reading it is at least \(d\) clock cycles and less than \(d+2 \cdot s+1\) clock cycles. Since the skew is at most \(s\) clock cycles, it follows that the implementation behaves as required.

Note, that if the breadth of an interconnection is \(k\) (i.e. \(k\) bits are sent in parallel every clock cycle), then the proposed implementation requires \(k \cdot n\) “real” wires. The area of an \(n\)-multiplexer and an \(n\)-demultiplexer for breadth \(k\) is \(O(k \cdot n)\).

The propagation delay of the \(n\)-demultiplexer and the \(n\)-multiplexer can be accounted for in the delay of the wires. It is reasonable to assume that the propagation delays of the \(n\)-multiplexer and the \(n\)-multiplexer are \(O(\log n)\). Therefore, it is sufficient to use a value of \(n\) which satisfies: \(n \geq d+2 \cdot s+1+O(\log n)\). This does not increase \(n\) significantly, and we ignore this increase, for the sake of simplicity, in the sequel.

### 3 Linear Arrays

In this section we consider a systolic linear array which consists of cells which are Mealy finite state machines. Every edge connecting two cells has a register along it. For the sake of simplicity, we assume that all the cells are identical, however, the proposed implementation holds even when the cells are not identical. Suppose that one wishes to implement such a systolic linear array which is so long, that it does not fit into one chip. When one maps this linear array onto more than one chip, the interconnections between chips incur a large delay or there might be a large clock skew between the clocks of connected chips. The proposed solution reduces the problem caused by chip-to-chip interconnections to the situations solved in section 2, which can be implemented so that they can operate with short clock periods. In particular, this reduction transforms the linear array into an equivalent circuit, in which all chip-to-chip interconnections have several registers along them. The functionality of the linear array is kept intact. Assume that the feasible clock period of the linear array when mapped onto a single chip equals 1 time unit. First, we show a solution which enables a clock period of 2 time units. Then, we show how one can obtain clock periods which are close to 1 time units, when the gate level description of the cells is considered.

Let \(m\) denote the number of registers along the interconnection required for the implementation suggested in section 2 for overcoming the delay of wires interconnecting chips and the clock skew between connected chips. Namely, \(m = d + s\), where \(d\) denotes the delay and \(s\) denotes the clock skew. The reduction consists of two stages: In the first stage, a segment of \(O(m)\) cells is duplicated, so that a copy of the duplicated segment appears on both sides of the interconnection. The duplicated segments perform, at this stage, the same computations at the same time, and therefore, functionality is kept intact. In the second stage, the duplicated segments are retimed. This retiming causes all troublesome interconnections (which cross from one chip to another) to have \(m\) registers along them. Retiming preserves functionality, and therefore, the original linear array and the obtained circuit are functionally equivalent. The interconnections between chips are implemented as suggested in section 2.
Claim 1: Consider an implementation of an interconnection depicted in figure 1. Assume that the delay and clock skew are measured in clock periods. The following statements hold:

1. Assume that the clock skew is zero. Let $n = d + 1$ and let the shift be $d$. The implementation realizes an interconnection with a latency equal to $d$, even if the delay of each wire is upto $d$.

2. Assume that the wire delay is zero. Let $n = 2 \cdot s + 1$ and let the shift be $s$. The implementation realizes an interconnection with a latency equal to $s$, even if the the clock skew is upto $s$.

3. Let $n = d + 2 \cdot s + 1$ and let the shift be $(d + s)$. The implementation realizes an interconnection with a latency equal to $(d + s)$, even if the delay of each wire is upto $d$ and the clock skew is upto $s$.

Proof:

proof of part 1: A value written to a wire is kept valid for $d + 1$ clock cycles, and arrives at the input of the multiplexer after at most $d$ clock cycles. The multiplexer reads a value from a wire exactly $d$ clock cycles after it is written to it. Therefore, the value read from a wire equals the value written to it, and the signal is delivered through the interconnection with a latency equal to $d$ clock cycles as required.

proof of part 2: We differentiate between the clock cycles of the multiplexer and the demultiplexer. In the $i$'th clock cycle of the demultiplexer, a value is written on wire $e_{i \mod (2 \cdot s + 1)}$. The multiplexer reads a value on wire $e_{i \mod (2 \cdot s + 1)}$ in cycle $i + s$ of its clock. Since the clock skew is at most $s$ clock cycles, the time elapsed between writing and reading the value to the wire is non-negative and it is not greater than $2 \cdot s$ clock cycles. Since the value written on the wire remains
The physical setting is characterized by the following parameters:

1. The delay of the wires. The delay is the time that elapses from the moment a signal is input to a wire until the signal arrives at the other end of the wire. The delay is denoted by $d$. Moreover, the signal is supposed to remain valid for at least $d$ time units at the input to insure correct arrival at the other end of the wire. Therefore, a delay of a wire also determines the bandwidth of the wire.

2. The clock skew between the chips. We consider clock skews which can be much larger than the clock period, and thus constitute a “loosely” synchronous implementation. Let $s$ bound the clock skew, i.e. the absolute value of the difference between the $i$'th pulses of both clocks is at most $s$.

Henceforth, we normalize the values of the delay and the clock skew by dividing these values by the clock period. In other words, $d$ and $s$ are measured in units of clock periods, where the clock period is part of the specification of the behavior of the interconnection. This normalization frees us from referring to the value of the clock period.

We propose implementations of the following combinations of behavioral specification of the interconnection and physical settings:

1. An interconnection with a latency of $d$ clock cycles, in a physical setting which has a delay $d > 0$ and a clock skew $s = 0$.

2. An interconnection with a latency of $s$ clock cycles, in a physical setting which has a delay $d = 0$ and a clock skew $s > 0$.

3. An interconnection with a latency of $(d + s)$ clock cycles, in a physical setting which has a delay $d > 0$ and a clock skew $s > 0$.

The proposed implementation uses two components, an $n$-demultiplexer and an $n$-multiplexer, which are defined as follows: An $n$-demultiplexer is clock controlled and has a single input and $n$ outputs. It is a device which has memory and it routes the input to the outputs, one by one, in cyclic order. In each clock cycle, the input is routed to one output, and the values of the other outputs are kept intact. The $n$-multiplexer is also clock controlled and has $n$ inputs and a single output. The $n$-multiplexer routes its inputs to its output, one by one, in cyclic order.

Figure 1 depicts an implementation of an interconnection constructed with an $n$-demultiplexer and an $n$-multiplexer. The implementation uses $n$ wires in order to implement a single “required” interconnection. We name the wires $e_0, e_1, ..., e_{n-1}$. Wire $e_i$ emanates from the $i$'th output of the demultiplexer, and enters the $i$'th input of the multiplexer.

Assume for the time being that the propagation delay of an $n$-demultiplexer and an $n$-multiplexer is zero. We consider these propagation delays at the end of this section.

Correct functionality of an implementation of an interconnection requires synchronizing between the $n$-multiplexer and the $n$-demultiplexer. This synchronization is quantified by a shift. When the shift is set to $\delta$, the components function as follows: The $n$-multiplexer, in its $i$'th clock cycle, sets the value of output $(i \mod n)$ to the value of the input, all other outputs are kept intact. The $n$-demultiplexer, in its $(i + \delta)$ clock cycle, routes its $(i \mod n)$ input to its output. Synchronizing the shift between the multiplexer and the demultiplexer is part of initializing the circuit, and is done during power up. The suggested implementation resembles circular buffers which are common in software.

The following claim summarizes the properties of such an interconnection:
the implementation of the interconnection by connecting the chips with several parallel wires. This implementation uses two special hardware components which we call an \( n \)-demultiplexer and an \( n \)-multiplexer. A fine tuning between the required behavior of the interconnection and the parameters of the physical setting enables correct functionality of the implementation with very fast clock rates.

2. We duplicate logic in the vicinity of the interconnections between chips. The same computations are performed by the duplicated logic at both ends of an interconnection between chips.

3. The systolic array (after duplicating logic) is retimed so that two objectives are met: (a) The chip-to-chip interconnections have enough registers along them so that the implementation proposed in the first stage is applicable. The number of registers required along the interconnection between chips is determined by the delay and clock skew. (b) The retimed array can operate with a clock period which almost equals the feasible clock period which is obtainable when a small version of the array is mapped onto a single chip.

Our solution is applicable also in situations in which a systolic array is partitioned among remote areas of the same chip. In general, this work presents a method of overcoming the naive lower bound on clock rates imposed by sparse slow interconnections in systolic arrays.

This paper is organized as follows: Section 2 presents an implementation, in a given physical setting, of an interconnection whose behavior is specified. Section 3 presents our solution for systolic linear arrays. Section 4 presents our solution for systolic two-dimensional arrays. Section 5 discusses the hardware overheads required by our solutions.

## 2 Trading delay with latency

In this section we describe a mechanism which we use in the proposed solutions in sections 3 and 4. This tool describes how to implement an interconnection whose behavior is given in a physical setting which is characterized by long delays and clock skews.

The required behavior of the interconnection between chips is specified as follows:

1. The interconnection is synchronous. A new value is input on every clock cycle, and it outputs a new value on every clock cycle. The clock period of the interconnection is specified.

2. Latency - the number of clock cycles which elapse from the time a signal is input to the interconnection until the signal is received at the interconnection’s output. We also refer to this parameter as “the number of registers along the interconnection”. The reason for this terminology comes from the fact, that one can regard the required behavior of an interconnection as if it is a \( k \)-stage shift register. Every clock cycle, a new value is input, and a value is output from the the shift register. It takes a value \( k \) clock cycles to get to the output of the shift register, and therefore, the latency is \( k \) clock cycles.

   We require that one may input new signals to the interconnection on every clock cycle, regardless of the number of registers along the interconnection. In other words, the number of registers along a wire effects the latency, but does not effect on the throughput.

3. The breadth of the interconnection. This parameter specifies the number of bits input simultaneously to the interconnection. To simplify the discussion, we first deal with interconnections with single-bit breadths.
1 Introduction

This paper proposes a solution which enables mapping large systolic linear arrays and systolic two-
dimensional arrays onto several chips while almost maintaining the clock rates which are obtainable
when these circuits are small enough to fit into a single chip. In general, mapping a circuit onto
several chips incurs a physical setting which differs from those within a chip. Specifically, the delay
of chip-to-chip interconnections is much longer than the delay of wires and gates in a chip. This
delay also affects the bandwidth of the wires. In addition, the clock skew between chips is larger
than the clock skew within a chip. One may mistakenly conclude that the feasible clock period of
a systolic array cannot be smaller than the maximal delay of an interconnection in a realization of
the circuit.

Systolic arrays are highly suited for VLSI circuits. Many systolic designs have appeared in the
literature (Kung [K82] and Leiserson [L83] provide a few designs). We focus on systolic linear arrays
and systolic two-dimensional arrays. These systolic arrays have a very simple physical layout, which
have short interconnections between functional units. The functional units of systolic arrays are
called “cells”. The cells are Mealy finite-state machines, and the interconnections between the cells
have registers along them. Systolic arrays consist of identical cells, or are constructed from a small
variety of cells. Therefore, systolic arrays can function with very fast clock rates. These properties
accommodate scalability, since the clock period does not depend on the size of the systolic arrays.
Namely, the minimum feasible clock period of a systolic array is determined by the propagation
delay of its cells, and not by the number of cells in the array. However, how does one maintain
such fast clock rates when the systolic array contains too many cells to fit into a single chip? This
paper presents a solution to this problem which enables a feasible clock period which is very close
to the clock period of a small systolic array which fits into one chip. Our solution does not rely on
special analogue techniques. It is described in a sequence of transformations (logic duplication and
retiming), reductions, and an implementation of interconnections which have a required behavior
in a given physical setting. It is shown that each step preserves functionality, and subsequently,
the correctness of the proposed solution is implied.

We formalize the parameters which characterize the physical properties of chip-to-chip inter-
connections as follows: The delay of an interconnection is the amount of time it takes for a signal to
arrive to the other end of the interconnection. The delay also affects the bandwidth, since the input
to the interconnection must be kept stable until the signal arrives to the other end. Delays of chip-
to-chip interconnections are much larger than the delays of wires and gates within a chip. Utilizing
fast clock rates in presence of long delay interconnections is an important issue, which we solve
for systolic linear arrays and systolic two dimensional arrays. Our implementation allows delays
of chip-to-chip interconnections which equal several clock periods. In addition, we consider clock
skews between the clocks of the chips. Our implementation is “loosely” synchronous in the sense
that it allows clock skews of several clock periods. We believe that this is the first work which deals
which such large clock skews, and which shows how to implement “purely” synchronous circuits in
“loosely” synchronous environments.

Our solution consists of three stages:

1. We consider an interconnection between chips whose behavior is given and which is supposed
to be realized in a given physical setting which is characterized by long delays and clock skews.
This implementation uses two ideas: First, we trade the delay of the wires connecting chips
with the latency of the interconnection (which is the number of clock cycles it should take
a signal to propagate through the interconnection). Second, we increase the bandwidth of
Overcoming chip-to-chip delays and clock skews

Guy Even* Ami Litman †

Abstract

In general, mapping a circuit onto several chips incurs a physical setting which differs from those within a chip. Specifically, the delay of chip-to-chip interconnections is much longer than on-chip delays of wires and gates. This delay affects the bandwidth as well. In addition, the clock skew between chips is larger than the clock skew within a chip. One may mistakenly conclude that the feasible clock period of a systolic array cannot be smaller than the maximal delay of an interconnection in a realization of the circuit.

This paper proposes a technique for mapping large systolic linear arrays and systolic two-dimensional arrays onto several chips while almost maintaining the clock rates which are obtainable when these circuits are small enough to fit into a single chip.

Our solution does not rely on special analogue techniques. It is described in a sequence of transformations (logic duplication and retiming), reductions, and an implementation of interconnections which have a required behavior in a given physical setting. It is shown that each step preserves functionality, and subsequently, the correctness of the proposed solution is implied.

*Computer Science Dept., Technion, Haifa 32000, Israel. guy@cs.technion.ac.il. Research supported by the Miriam and Aaron Gutwirth Memorial Fellowship.

†Computer Science Dept., Technion, Haifa 32000, Israel. litman@cs.technion.ac.il.