Multiple-inputs systolic priority queue for fast sequential decoding of convolutional codes

H.-C. Kuo
C.-H. Wei

Indexing terms: Convolutional decoding, Metric searching, Sequential decoding, Systolic arrays, Trellis modulation

Abstract: The operating speed of a sequential decoder with stack algorithm is usually limited by the time to search the best node for further extension. This problem can be completely alleviated by using the systolic priority queue to replace the stack memory. However, the systolic priority queues developed previously are accessible only in the cases when the number of inputs processed is small. This is because the complexity of a queue grows up quickly as the volume of data flowing through it increases. Since the largest amount of data flowing through a systolic priority queue is equal to the number of inputs to this queue, the systolic priority queue is not suitable for a system with many inputs. A modified version of previously developed circuits is proposed. The number of transmission gates required in this circuit is proportional to $3N$ instead of N^2, where N is the number of inputs. And the total number of control signals is proportional to $3N^2$ instead of N^3. But the number of comparators required is proportional to C_2^{N+1}, as before. This modified circuit can be used in cases where the number of inputs is small ($N \leqslant 8$). A new algorithm for the multiple-inputs systolic priority queue (MISPQ) is proposed. By using this algorithm, a MISPQ may be implemented with several smaller queues, each is used to process a part of data in the MISPQ. Since the volume of data flowing through each queue is small, these queues will be simpler. However, some additional circuits should be used for the interactions between queues. A circuit for implementing this algorithm is presented and its complexity is analysed. The number of transmission gates for the MISPQ is proportional to $3N$, the number of control signals is proportional to $(3N^2/2)$, and the number of comparators is proportional to $4C_2^{N/2+1}$. Thus this new architecture is feasible for large N (e.g. $N \geqslant 8$).

1 Introduction

Systolic priority queue [1, 2] is a circuit developed for searching the best metric from a large set of metrics. The fast and constant searching speed makes the systolic priority queue useful in some applications. An important example is that the systolic priority queue can be used to replace the stacks in the stack algorithm for sequential decoding [3–8]. With such replacement, the sorting problem in the sequential decoding can be alleviated completely.

The linear systolic priority queue (Fig. 1) is used in a sequential decoder by Chang and Yao [4]. In every operating cycle, one input is sent into this queue and one

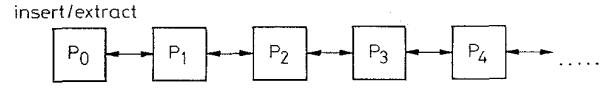

Fig. 1 *Architecture for linear systolic priority queue*

output is extracted from it. Recently, Lavoie, Haccoun and Savaria [6] have developed a parallel-entry systolic priority queue (Fig. 2), which can accept two input

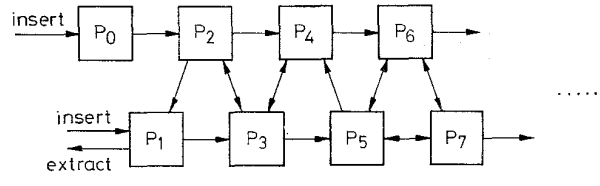

Fig. 2 *Architecture for PESPQ*

metrics at a time, and the best metric is extracted out in the same cycle. The parallel-entry systolic priority queue has been successfully applied to the sequential decoding of rate $1/n$ convolutional codes or high-rate puncture convolutional codes [6]. These algorithms for systolic priority queues are well defined and can be easily extended to the N-input case. However, the circuit complexity grows quickly as N increases, because the required number of transmission gates is proportional to N^2 and the number of control signal is proportional to N^3.

In this paper, a modified circuit for the systolic priority queue is proposed. By restricting the possible destinations for metric in each processor during the metrics transmission operations, the circuit can be simplified so that the number of transmission gates required is proportional to $3N$ and the number of control signals is proportional to $3N^2$. This modified circuit is suitable for cases when N are small ($N \leqslant 8$). To cope with cases with larger number of inputs, a new algorithm and the corresponding architecture are then proposed for a multiple inputs systolic priority queue (MISPQ). According to this algorithm, a MISPQ consists of several smaller queues with each queue processing a part of the data. Such an arrangement will make the new MISPQ (type II MISPQ)

Paper 2163G (E10), first received 12th September 1994 and in revised form 23rd June 1995

The authors are with the Institute of Electronics and Center for Telecommunications Research, National Chiao-Tung University, Hsinchu, Taiwan, Republic of China

simpler than other systolic priority queue when N is large. For example, the number of transmission gates is proportional to $3N$ and the number of control signals proportional to $(3N^2/2)$. This new algorithm is suitable for fast sequential decoding of high-rate convolutional codes or trellis modulation codes.

2 Operations of multiple inputs systolic priority queues

In [3, 4] Chang and Yao used an array of processors to replace the stack memory required in the stack sequential decoder. These processors, called a systolic priority queue, are properly arranged so as to deliver the best metric quickly within a constant time interval. Since the operations of a systolic priority queue are completed simultaneously when the metrics travel through the processors in the queue, the time to get the best metric is always constant, no matter how many processors are used. This method can efficiently improve the sorting speed for searching best metric; otherwise the speed may become quite slow if the size of memory used is large. The implementations of the systolic priority queue with RAM or registers have been discussed in [3]. However, the combination of control signals and the distribution of transmission paths into each processor is not as easy as those on VLSI, for example, the following PESPQ.

Lavoie, Haccoun and Savaria [5, 6] further developed an advanced architecture called the parallel-entry systolic priority queue. Based on the operating principles and some special circuit design, this new scheme of systolic priority queue can complete all operations in a single clock cycle. The parallel-entry systolic priority queue is shown in Fig. 2. Following these research results, a multiple-input systolic priority queue is represented in Fig. 3, which is referred to as type I MISPQ. From Figs.

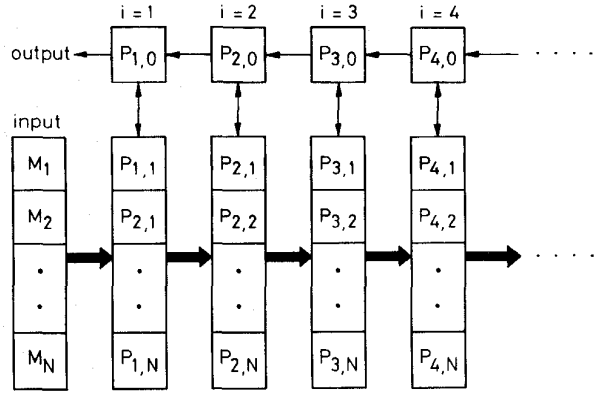

Fig. 3 *Architecture for type I MISPQ*

4 to 6, the operations of type I MISPQ are demonstrated step by step, where only the required connection lines are shown. These Figures are used only for illustration. When realising them with VLSI, the connection lines will be more complicated, as shown in the following Section.

In the VLSI implementation, each column in Fig. 3 may be named as a slice [5]. Processors in each slice belong to a group. The operations of systolic priority queue are usually considered slice by slice. In the following, the nomenclature slice is used frequently for ease of describing the operations. Operations of type I MISPQ are described in the following paragraph. The operations of the linear systolic priority queue and the parallel-entry systolic priority queue, and their direct extension to the N-input case can be found in [3, 6, 8]. Note that processors in Fig. 1 are labelled with one-dimensional notation, while the processors in Figs. 3–6 are labelled with two-dimensional notation; however, the algorithms for both cases are similar.

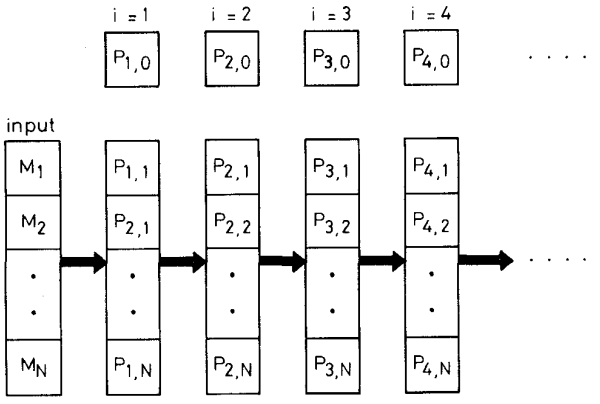

Fig. 4 *Insert input metrics and shift metrics to right*

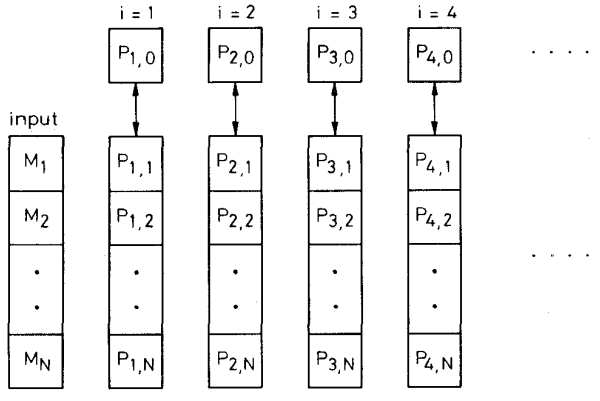

Fig. 5 *Rearrange metrics in each slice*

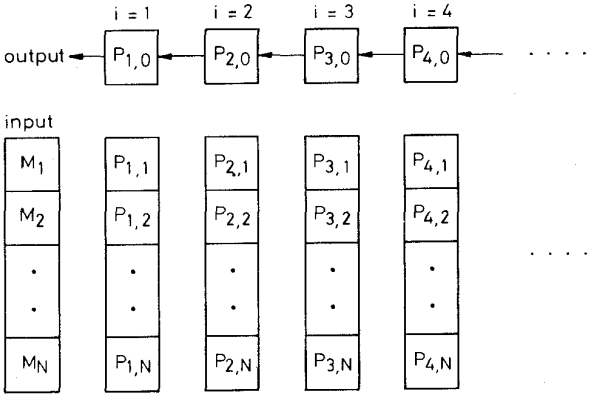

Fig. 6 *Extract best metric and shift top metric to left*

2.1 Operations for type I MISPQ

(*a*) Result of these operations: simultaneous insertion of N new input metrics $M_1, M_2, \ldots, M_N$ into the queue, and delivery of the best metric it contains.

(*b*) Mechanism of this algorithm:

(i) Insert N metrics simultaneously into the queue and shift each metric at position $P_{i,j}$ to its right, i.e. to $P_{i+1,j}$, where $i \geqslant 1$, $N \geqslant j \geqslant 1$. Note that the metric at $P_{i,0}$ in each ith slice is not shifted, as shown in Fig. 4, i.e.

$$\text{insertion: } M_1 \rightarrow P_{1,1}, M_2 \rightarrow P_{1,2}, \ldots, M_N \rightarrow P_{1,N}$$

and

shifting: $P_{i,1} \rightarrow P_{i+1,1}$, $P_{i,2} \rightarrow P_{i+1,2}$, ..., $P_{i,N} \rightarrow P_{i+1,N}$ for $i \geqslant 1$

(ii) Rearrange metric in each group of processors, i.e. in each slice. Move the best metric in each slice to its local top position. The positions of the other metrics are trivial, as shown in Fig. 5, i.e. for $i \geqslant 1$,

$P_{i,0} \leftarrow$ best metric among those metrics in position $(P_{i,0}, P_{i,1}, \ldots, P_{i,N})$

$P_{i,1}, P_{i,2}, \ldots, P_{i,N} \leftarrow$ the other N metrics

(The rearrangements of metrics in this algorithm and in that of type II MISPQ discussed later are both done by pairwise comparisons of processors in each slice.)

(iii) Extract the $P_{1,0}$ from the queue, which is always the best metric among all. At the same time shift the metric on the top of each slice to its left, as shown in Fig. 6, i.e. for $i \geqslant 1$,

$P_{i+1,0} \rightarrow P_{i,0}$ and $P_{1,0}$ is extracted

(iv) Rearrange metrics in each slice again, as shown in Fig. 5, i.e. for $i \geqslant 1$,

$P_{i,0} \leftarrow$ best metric among these metrics in position $(P_{i,0}, P_{i,1}, \ldots, P_{i,N})$

$P_{i,1}, P_{i,2}, \ldots, P_{i,N} \leftarrow$ the other N metrics

Although this mechanism consists of many steps, the steps can be merged into two steps. The first step consists of insertion, shifting metrics to the right, and sorting metrics in each slice. The second step consists of shifting the top metric in each slice to the left and sorting metrics in each slice. Thus with a two-phase clocking scheme these steps can also be completed in one single clock cycle. The merging of these steps is done by following the special circuit design presented in [6]. That is, the metrics in their original positions are precompared to determine their orders, which orders are then used as reference for the distribution of these metrics onto their future positions.

A theorem with proof is derived to confirm that the systolic priority queue described can work as desired.

Definition 1: As shown in Fig. 3, for a metric at processor $P_{i,j}$, one may express the position of this processor with

$P_{k,0} < P_{i,j} < P_{k',0}$

(where $k \leqslant i < k'; j \neq 0$ when $k = i$)

if $P_{i,j}$ is one processor in the set of processors S, where

$$S = \{P_{k,1}, P_{k,2}, \ldots, P_{k,N}\} \cup \{P_{k+1,0}, P_{k+1,1}, \ldots, P_{k+1,N}\} \cup \{P_{k+2,0}, P_{k+2,1}, \ldots, P_{k+2,N}\} \cup \cdots \cup \{P_{k'-1,0}, P_{k'-1,1}, \ldots, P_{k'-1,N}\}$$

and one may express the position of this processor with

$P_{k,0} \leqslant P_{i,j} \leqslant P_{k',0}$

(where $k \leqslant i \leqslant k'; j = 0$ when $i = k'$)

if $P_{i,j}$ is one processor in the set S', where

$S' = S \cup \{P_{k,0}, P_{k',0}\}$

Theorem 1: For a type I MISPQ, after any number of insertion or deletion (extraction) operations, the kth good metric is stored in some processor $P_{i,j}$, where $P_{1,0} \leqslant P_{i,j} \leqslant P_{k,0}$. For example, the best metric ($k = 1$) will always reside at $P_{1,0}$. For another example, the third good metric ($k = 3$) may reside at some place in slice 2 or slice 1. But if it resides at somewhere in slice 3, the only position must be $P_{3,0}$.

Proof of theorem 1: By induction as follows. The first several steps can be easily checked by observations, so that one may assume that after m steps of operation the kth good metric resides in processor $P_{i,j}$ which satisfies $P_{1,0} \leqslant P_{i,j} \leqslant P_{k,0}$. Then one is to prove that it is still true at the $(m + 1)$th step.

(a) If the $(m + 1)$th operation is an insertion, the best metric in the queue will still appear at $P_{1,0}$ for the kth good metric, where $k \geqslant 2$. First, if it resides in $P_{i,j}$ where $P_{1,0} \leqslant P_{i,j} \leqslant P_{k-1,0}$, then after the new N metrics being inserted this metric will still reside somewhere before $P_{k,0}$, and thus is a legal position (when $k = 2$ the case $P_{1,0} \leqslant P_{i,j} \leqslant P_{1,0}$ will not occur). Secondly, if it resides at $P_{i,j}$ with $P_{k-1,0} < P_{i,j} \leqslant P_{k,0}$, then after operations, this metric will reside at $P_{k,0}$. This is because the other metrics to be compared with this previous kth good metric all have ranks not lower than k, otherwise theorem 1 is violated at the mth step. Since the new rank of this metric will not be lower than k, this is a legal position again.

(*b*) If the $(m + 1)$th operation is a deletion (extraction). The new best metric will appear at $P_{1,0}$. This is because the previous second good metric is originally at $P_{i,j}$ with $P_{1,0} < P_{i,j} \leqslant P_{2,0}$, after shifting all top metrics to the left the new best metric will appear somewhere among the first $(N + 1)$ positions. Now consider the position for the kth good metric. First, if the previous kth ($k \geqslant 3$) good metric resides at some $P_{i,j}$ with $P_{1,0} < P_{i,j} \leqslant P_{k-1,0}$, then after operations it can still reside at a legal position. This is because $P_{1,0} < P_{i,j} < P_{k-1,0}$, thus is at a legal position for the $(k - 1)$th good metric. Secondly, if the previous kth ($k \geqslant 3$) good metric resides at some position $P_{i,j}$ with $P_{k-1,0} < P_{i,j} \leqslant P_{k,0}$, then after operations that metric will appear at $P_{k-1,0}$. Because the metrics to be compared with the previous kth good metric are located originally at some $P_{i,j}$ with $P_{k-1,0} < P_{i,j} \leqslant P_{k,0}$. Their ranks are all no less than k.

From (*a*) and (*b*), it can be seen that theorem 1 still holds at the $(m + 1)$th step. By induction, theorem 1 is always true after any number of steps of the algorithm.

In the following, a new algorithm for the systolic priority queue referred to as type II multiple inputs systolic priority queue is proposed. This new MISPQ is shown in Fig. 7. Figs. 8 to 10 illustrate the operations of this queue step by step. It can be seen that Fig. 3 is an abstract representation of Fig. 7.

Definition 2: In each slice, group of processors may be further divided into 'top group of processors' and 'bottom group of processors'. Thus one may use subslice to describe the region where the subgroup of processors reside. For example, in Fig. 7, each slice is divided into two subslices.

Definition 3: According to the operations of the queue described subsequently, the best metric in slice i must reside in the top position of one of the two subslices, i.e. in $T_{i,0}$, or $B_{i,0}$. No matter where it is, one may define a

pseudo top position $P_{i,0}$ for the best metric in slice i. That is

$$P_{i,0} = T_{i,0} \cup B_{i,0}$$

2.2 Operations for type II MISPQ

(*a*) Result of these operations: simultaneous insertion of N new input metrics $M_1, M_2, \ldots, M_N$ into the queue, and delivery of the best metric it contains.

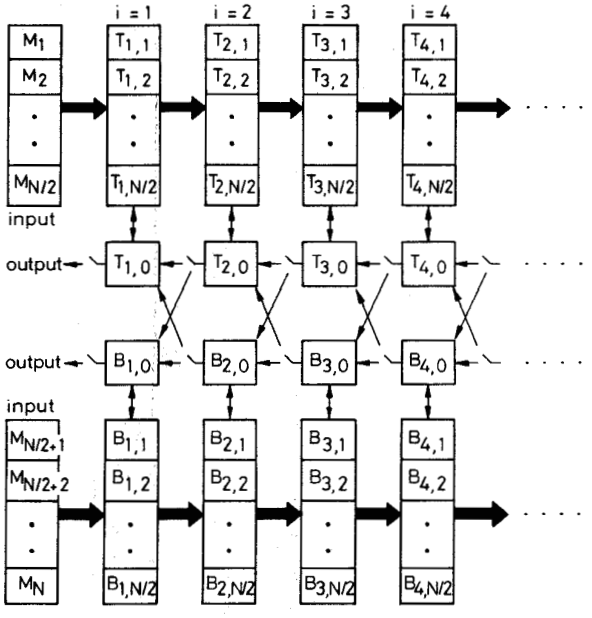

Fig. 7 *Architecture for type II MISPQ*

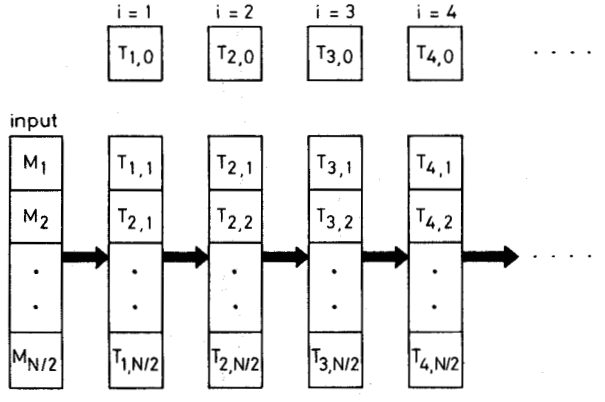

Fig. 8 *Insert input metrics and shift metrics to right in top subslice*
(Note that mechanism is the same in bottom subslice)

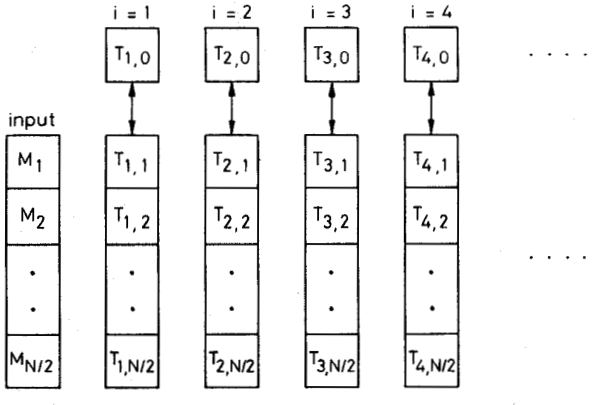

Fig. 9 *Rearrange metrics in top subslice*
(Note that mechanism is the same in bottom subslice)

(*b*) Mechanism of this algorithm:

(i) Insert N metrics simultaneously into the queue and shift each metric at position $T_{i,j}$ to its right, i.e. to $T_{i+1,j}$, and $B_{i,j}$ to $B_{i+1,j}$, where $i = 1, 2, \ldots; j = 1, 2, \ldots, N/2$. Note that the metrics at $T_{i,0}$ and $B_{i,0}$ in each

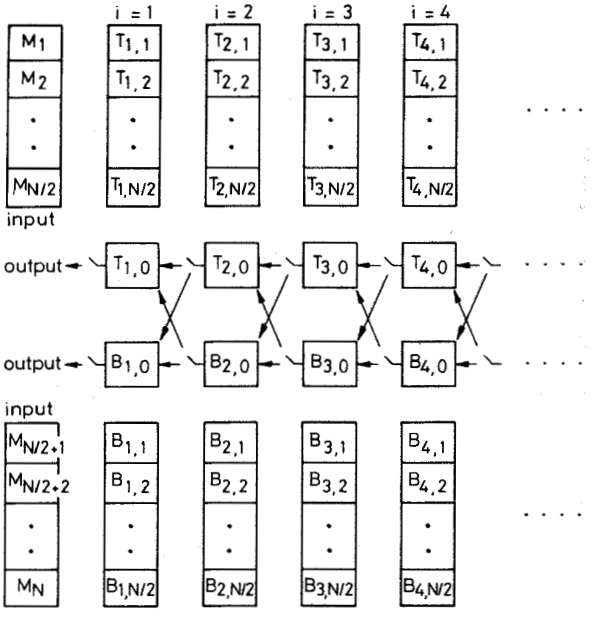

Fig. 10 *Extract best metric and shift top metric to right*

ith slice are not shifted, as shown in Fig. 8, i.e. for $i = 1, 2, \ldots,$

insertion:

$$M_1 \rightarrow T_{1,1}, M_2 \rightarrow T_1, \ldots, M_{N/2} \rightarrow T_{1,N/2}$$

and

$$M_{N/2+1} \rightarrow B_{1,1}, M_{N/2+2} \rightarrow B_{1,2}, \ldots, M_N \rightarrow B_{1,N/2}$$

then

shifting:

$$T_{i,1} \rightarrow T_{i+1,1}, T_{i,2} \rightarrow T_{i+1,2}, \ldots, T_{i,N/2} \rightarrow T_{i+1,N/2}$$

$$B_{i,1} \rightarrow B_{i+1,1}, B_{i,2} \rightarrow B_{i+1,2}, \ldots, B_{i,N/2} \rightarrow B_{i+1,N/2}$$

(ii) Rearrange metrics in each subgroup of processors, i.e. in each subslice. Move the best metric in each subslice to its local top position ($T_{i,0}$ for the top subslice and $B_{i,0}$ for the bottom subslice). The positions of the other metrics are trivial, as shown in Fig. 9, i.e. for $i = 1, 2, \ldots,$

$T_{i,0} \leftarrow$ best metric among $(T_{i,0}, T_{i,1}, \ldots, T_{i,N/2})$

$B_{i,0} \leftarrow$ best metric among $(B_{i,0}, B_{i,1}, \ldots, B_{i,N/2})$

$T_{i,1}, T_{i,2}, \ldots, T_{i,N/2}$

$\leftarrow$ the other $N/2$ metrics in the top subslice

$B_{i,1}, B_{i,2}, \ldots, B_{i,N/2}$

$\leftarrow$ the other $N/2$ metrics in the bottom subslice

(iii) Shift the best metric in $P_{i,0}$ to $P_{i-1,0}$, and then the best metric in $P_{1,0}$ is extracted, i.e. for $i = 2, 3, \ldots,$

If (*metric in* $T_{i,0}$) $\geqslant$ (*metric in* $B_{i,0}$), *and*
(*metric in* $T_{i-1,0}$) $\geqslant$ (*metric in* $B_{i-1,0}$)

shift metric in $T_{i,0}$ to $T_{i-1,0}$

If (*metric in* $T_{i,0}$) $\geq$ (*metric in* $B_{i,0}$), *and* (*metric in* $T_{i-1,0}$) $<$ (*metric in* $B_{i-1,0}$)

shift metric in $T_{i,0}$ to $B_{i-1,0}$

If (*metric in* $T_{i,0}$) $<$ (*metric in* $B_{i,0}$), *and* (*metric in* $T_{i-1,0}$) $\geq$ (*metric in* $B_{i-1,0}$)

shift metric in $B_{i,0}$ to $T_{i-1,0}$

If (*metric in* $T_{i,0}$) $<$ (*metric in* $B_{i,0}$), *and* (*metric in* $T_{i-1,0}$) $<$ (*metric in* $B_{i-1,0}$)

shift metric in $B_{i,0}$ to $B_{i-1,0}$

For $i = 1$, *i.e. in the frontend, the better metric in* $T_{1,0}$ *and* $B_{1,0}$ *is extracted out.*

(iv) Rearrange metrics in each subslice again. Move the best metric in each subslice to its local top position ($T_{i,0}$ for the top subslice and $B_{i,0}$ for the bottom subslice). The positions of the other metrics are trivial, as shown in Fig. 9, i.e. for $i = 1, 2, \ldots,$

$T_{i,0} \leftarrow$ best metric among $(T_{i,0}, T_{i,1}, \ldots, T_{i,N/2})$

$B_{i,0} \leftarrow$ best metric among $(B_{i,0}, B_{i,1}, \ldots, B_{i,N/2})$

$T_{i,1}, T_{i,2}, \ldots, T_{i,N/2}$

$\leftarrow$ the other $N/2$ metrics in the top subslice

$B_{i,1}, B_{i,2}, \ldots, B_{i,N/2}$

$\leftarrow$ the other $N/2$ metrics in the bottom subslice

Now recombine the subgroups $(T_{i,1}, T_{i,2}, \ldots, T_{i,N/2})$ and $(B_{i,1}, B_{i,2}, \ldots, B_{i,N/2})$ in the ith slice into a group $(P_{i,1}, P_{i,2}, \ldots, P_{i,N})$, i.e. let $P_{i,1} = T_{i,1}$, $P_{i,2} = T_{i,2}$, $\ldots$, $P_{i,N/2} = T_{i,N/2}$, and $P_{i,N/2+1} = B_{i,1}$, $P_{i,N/2+2} = B_{i,2}$, $\ldots$, $P_{i,N} = B_{i,N/2}$. And processors $T_{i,0}$ together with processor $B_{i,0}$ are treated as a single processor $P_{i,0}$. It is found that the mechanism of type II MISPQ is exactly the same as that of type I MISPQ. For example, result for the rearrangement operation in each subslice of type II MISPQ is that the best metric in the ith slice will reside at $P_{i,0}$, as expected in type I MISPQ. The result for the metric insertion and metrics shifting-right operations of type II MISPQ can also be easily checked to be the same as that of type I MISPQ. We propose a theorem similar to theorem 1. According to this, the best metric in type II MISPQ is extracted for every cycle.

Theorem 2: In every ith slice, where $i = 1, 2, \ldots,$ if the processors $T_{i,0}$ and $B_{i,0}$ are viewed as a single processor $P_{i,0}$, and processors $(T_{i,1}, T_{i,2}, \ldots, T_{i,N/2})$ and $(B_{i,1}, B_{i,2}, \ldots, B_{i,N/2})$ are recombined into $(P_{i,1}, P_{i,2}, \ldots, P_{i,N})$, then for a type II MISPQ, after any number of insertion or deletion (extraction) operations, the kth good metric is stored in some processor $P_{i,j}$, where $P_{1,0} \leqslant P_{i,j} \leqslant P_{k,0}$.

Proof of theorem 2: As for theorem 1 but first the processors $T_{i,j}$ and $B_{i,j}$ should be recombined in the manner described.

3 Implementations for MISPQ

This Section considers the implementations for both types of MISPQ. First, the circuit developed previously for type I MISPQ is reviewed; secondly, some modifications on that circuit are discussed. The modified circuit is shown to be more suitable for MISPQ than the previous circuit. The circuit for type II MISPQ is then considered. Complexities of the last-mentioned two circuits are compared for different values of N.

3.1 Circuit realisations for type I MISPQ

The parallel-entry systolic priority queue designed by Lavoie, Haccoun, and Savaria [5 6] is the first VLSI realisation for the MISPQ (the parallel-entry systolic priority queue is similar to a type I MISPQ with the number of inputs $N = 2$). This circuit can complete all operations in a single clock cycle, where each clock cycle consists of two phases Φ_1 and Φ_2. All circuits discussed in this Section are also operated according to a two-phase clocking scheme duplicated in Fig. 11.

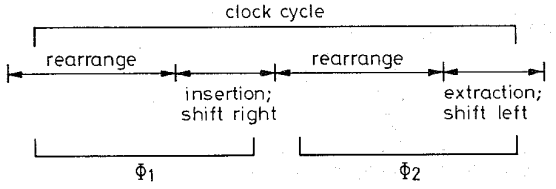

Fig. 11 *Timing diagram for new systolic priority queue*

Φ_1: phase 1 for one cycle
Φ_2: phase 2 for one cycle

To ensure that the parallel-entry systolic priority queue works following the timing diagram in Fig. 11, each slice of processors in Fig. 1 can be implemented as Fig. 12 [6]. As shown there, three comparators and nine

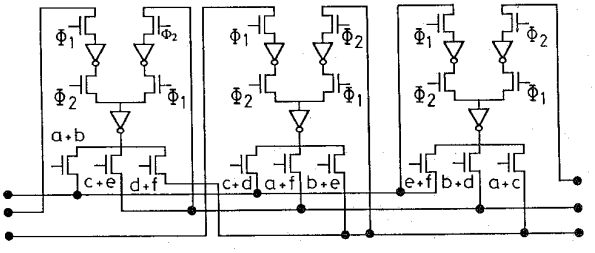

Fig. 12 *Transistors in slice of PESPQ [6]*

a: $A_{3i+2} \geq A_{3i+3} \geq A_{3i+4}$
b: $A_{3i+2} \geq A_{3i+4} > A_{3i+3}$
c: $A_{3i+3} > A_{3i+2} \geq A_{3i+4}$
d: $A_{3i+3} \geq A_{3i+4} > A_{3i+2}$
e: $A_{3i+4} > A_{3i+2} \geq A_{3i+3}$
f: $A_{3i+4} > A_{3i+3} > A_{3i+2}$
n-type transistor
inverter

controlled transmission gates are required. Each transmission gate is controlled by a signal composed of combinational AND logic with two inputs. Note that each signal corresponds to the pairwise comparison result for some processor with the other two processors, although they are expressed as $a + b$, $c + d$, or $e + f$ in Fig. 12. This architecture can be extended to the case where there are $N + 1$ processors in each slice, i.e. to the N-inputs case. In such a case, C_2^{N+1} comparators and $(N + 1)^2$ controlled transmission gates are required for this architecture. And each transmission gate is controlled by a signal composed of combinational AND logic with N inputs. Each signal corresponds to the pairwise comparison result for some processor with the other N processors. The circuit for four-input systolic priority queue shown in Fig. 14 is used as an example. Ten comparators and 25 controlled transmission gates are required here.

From this example it is found that the complexity of this architecture will become unfeasible if N increases further.

It is interesting to see that the operations of this circuit in Fig. 12 are not exactly the same as the operating rules

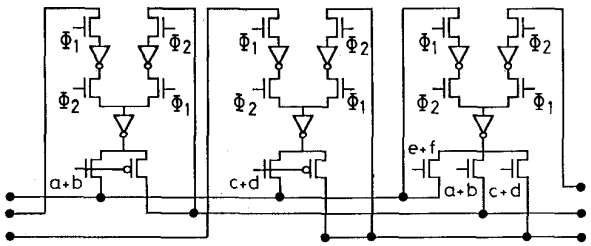

Fig. 13 *Transistors in slice of modified PESPQ*
a: $A_{3i+3} \geq A_{3i+4} \geq A_{3i+5}$
b: $A_{3i+3} \geq A_{3i+5} > A_{3i+4}$
c: $A_{3i+4} > A_{3i+3} \geq A_{3i+5}$
d: $A_{3i+4} \geq A_{3i+5} > A_{3i+2}$
e: $A_{3i+5} > A_{3i+3} \geq A_{3i+4}$
f: $A_{3i+5} > A_{3i+4} > A_{3i+3}$
p-type transistor ≡

for type I MISPQ. The best metric in each slice does not always reside at a predetermined processor. As shown next, there is some advantage in complexity if a circuit is designed following the rules for type I MISPQ. A modified circuit for the PESPQ is depicted in Fig. 13. This

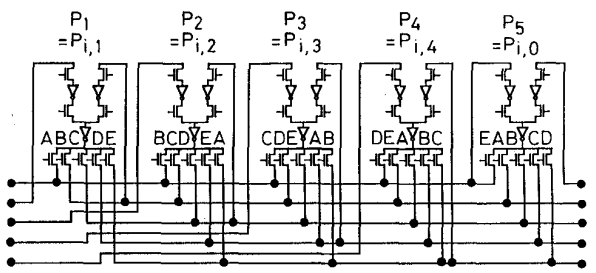

Fig. 14 *Circuit for realising slice in type I four-input systolic priority queue*
signal A = condition where P_1 is largest
signal B = condition where P_2 is largest
signal C = condition where P_3 is largest
signal D = condition where P_4 is largest
signal E = condition where P_5 is largest

circuit is operated strictly following the operating rules for type I MISPQ. As shown in Fig. 13, this architecture is a little simpler than that in Fig. 12, i.e. seven controlled transmission gates are used here. When extended to the case of N-inputs, C_2^{N+1} comparators and $3N + 1$ controlled transmission gates are required for this architecture. Each transmission gate is controlled by a signal, which is composed of combinational AND logic with N inputs.

The number of controlled transmission gates required in this modified circuit is less than that required in the original circuit. The reason is that, in the original circuit, the metric in each processor in a slice may be transferred to one of $N + 1$ other processors in each clock cycle. Since there are $N + 1$ processors, thus a total of $(N + 1)(N + 1)$ controlled transmission gates are required. Obeying the operating rules of type I MISPQ, in the modified circuit, the metric in each processor in a slice (except the processor containing the best metric) will be transferred to the top processor (if the metric is best) or a fixed processor. In this case, only two controlled transmission gates are required, no matter what the value N is. The processor containing the best metric is connected to the other $N + 1$ processors. Thus a total of $2N + (N + 1)$ controlled transmission gates are required in the modified circuit, which is simpler than the original circuit when implementing a type I MISPQ. The example, when the modified circuit is used to implement a four-input systolic priority queue, is shown in Fig. 15.

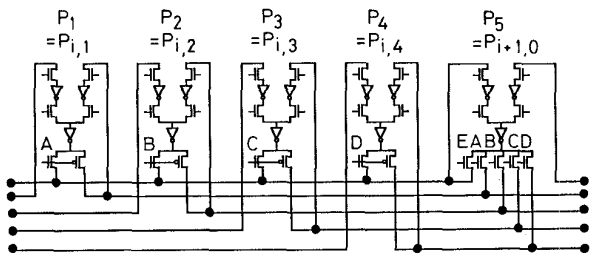

Fig. 15 *Circuit for realising slice in modified type I MISPQ*
Signals as for Fig. 14

The number of controlled transmission gates is 13, which is about half of that in the original circuit.

3.2 Circuit realisation for type II MISPQ

As shown in Fig. 7, a type II MISPQ with N inputs is roughly a combination of two $N/2$ inputs systolic priority queues, thus it is expected that the circuit for type I MISPQ with $N/2$ inputs may be adopted to implement a type II MISPQ.

The circuit for the type II MISPQ is shown in Fig. 16; the control signals for each gate are shown in Fig. 18. Since the signals for controlling a systolic priority queue are complicated, they are defined in Fig. 17. The functions of gates in this circuit are described in Fig. 19, with two group of functions for two phases, respectively. The reader may follow Figs. 16–19 to check the mechanisms of the type II MISPQ. An example is given later to illustrate the operation of this circuit. The circuit in Fig. 16 can be repeated to form a systolic priority queue with larger storage capacity. But it is very important to note that, in the I/O port, this circuit will be different. Referring to Fig. 16 with $i = 1$, it is found that the real body of a systolic priority queue should start from $T_{1,0}$ and $B_{1,0}$. Those transmission gates $PG_{0,1}$ to $PG_{0,6}$ will be eliminated and replaced by a switch to select $T_{1,0}$ or $B_{1,0}$ as the best. Processors $T_{0,1}$, $T_{0,2}$, $B_{0,1}$, and B_0, are replaced by the input port.

Since a type II MISPQ is operated as two independent queues in clock phase Φ_1, all the Boolean values for TLE_i and BL_i should be false at Φ_1 to make all the connections (through some 'plug-in' gates PG_{ij}) between the top subslices and the bottom subslices opened. The output port for a commonly used comparator [9] is modified so that at phase Φ_1 the comparison result will always be 00 as shown in Figs. 20–23. And at phase Φ_2, the comparators should work normally so that exchange of metrics between subslices is possible. In such a case, TLE_i is true when $T_{i,0}$ is larger than or equal to $B_{i,0}$. Otherwise, BL_i is true.

The processor $TB_{i+1,0}$ or $BT_{i+1,0}$ mentioned in Fig. 16 is not a fixed processor. At phase Φ_1, $TB_{i+1,0}$ represents $T_{i+1,0}$ and $BT_{i+1,0}$ represents $B_{i+1,0}$, respectively. But they both represent the larger of $T_{i+1,0}$ and $B_{i+1,0}$ at phase Φ_2. Such arrangements are to satisfy the mechanism of the type II MISPQ; i.e. at phase Φ_1, $T_{i,0}$ is compared with $T_{i-1,1}$, $T_{i-1,2}$, ..., and $T_{i-1,N/2}$, and $B_{i,0}$ is compared with $B_{i-1,1}$, $B_{i-1,2}$, ..., and $B_{i-1,N/2}$. But at phase Φ_2, the larger of $T_{i+1,0}$ and $B_{i+1,0}$ is to be com-

pared with ($T_{i+1,1}$, $T_{i-1,2}$, ..., $T_{i-1,N/2}$) or with ($B_{i-1,1}$, $B_{i-1,2}$, ..., $B_{i-1,N/2}$). The comparator used for comparing $TB_{i+1,0}$ with $T_{i,j}$ or $BT_{i+1,0}$ with $B_{i,j}$ (for $j = 1, 2, \ldots, N/2$) may be a specially designed three-input comparator. But additional logic gates will be inserted, thus the time for comparison operation will increase. To avoid this, two comparators are used, as shown in Figs. 24–26.

As shown in Fig. 16, 13 comparators and 22 controlled transmission gates are required in a slice of the type II MISPQ with four inputs. The number of input signals for controlling a gate range from four to six. When extending this architecture to N inputs case, there are $4C_2^{N/2+1} + 1$ comparators and $3N + 10$ controlled transmission gates required. This is counted as follows. The pairwise comparisons among the $(N/2) + 1$ processors $T_{i,1}$, $T_{i,2}$, ..., $T_{i,N/2}$, and $T_{i+1,0}$ in the top subslice require $C_2^{N/2+1}$ comparators. An equal amount of processors are required for comparing $B_{i+1,0}$ with $T_{i,1}$, $T_{i,2}$, ..., $T_{i,N/2}$. And the same case is in the bottom subslice. Finally, one additional comparator is required for comparing $T_{i,0}$ with $B_{i,0}$, thus a total of $4C_2^{N/2+1} + 1$ comparators are required.

The number of transmission gates required in the top subslice is $3(N/2) + 1$ from the architecture of modified type I MISPQ in Section 3.1. The same number is required in the bottom subslice. But there are additional eight transmission gates for the exchange of data between the top and bottom subslices. Thus a total of $2(3N/2 + 1) + 8$ transmission gates are required. The number of inputs to the combinational logic gates ranges from $(N/2) + 2$ to $(N/2) + 4$ as seen from Fig. 18. It is noted here that the number of comparators for the type II MISPQ is $4C_2^{N/2+1} + 1$ because the three-input comparator is implemented in a direct manner as shown in Figs.

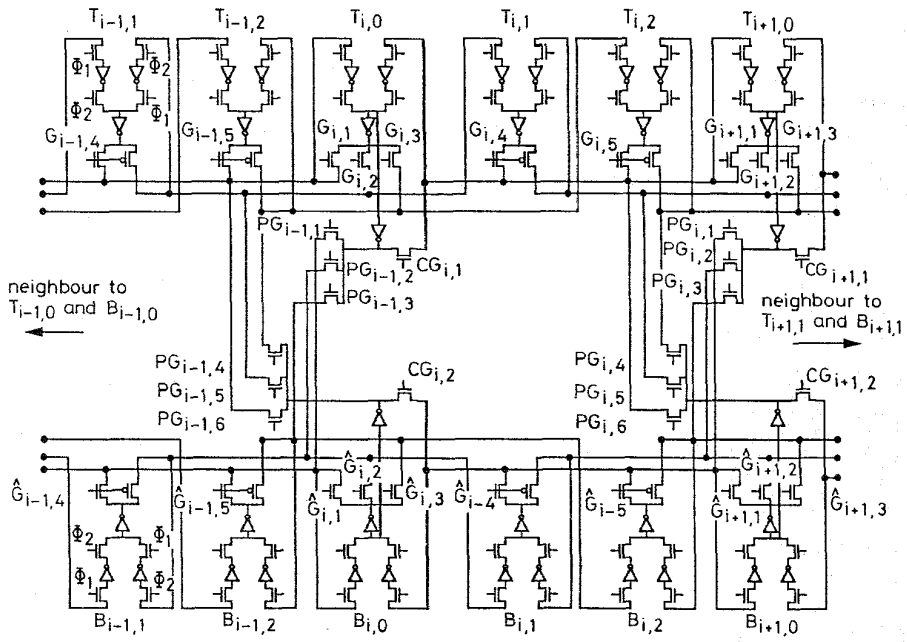

Fig. 16 *Circuit realisation for slice in MISPQ ($N = \infty$) and neighbouring configuration*

Signals representing relationships between metrics in processors		
a_i: $T_{i,1} \geq T_{i,2} \geq TB_{i+1,0}$	b_i: $T_{i,1} \geq TB_{i+1,0} > T_{i,2}$	c_i: $T_{i,2} > T_{i,1} \geq TB_{i+1,0}$
d_i: $T_{i,2} \geq TB_{i+1,0} > T_{i,1}$	e_i: $TB_{i+1,0} > T_{i,1} \geq T_{i,2}$	f_i: $TB_{i+1,0} > T_{i,2} > T_{i,1}$
$\hat{a}_i$: $B_{i,1} \geq B_{i,2} \geq BT_{i+1,0}$	$\hat{b}_i$: $B_{i,1} \geq BT_{i+1,0} > B_{i,2}$	$\hat{c}_i$: $B_{i,2} > B_{i,1} \geq BT_{i+1,0}$
$\hat{d}_i$: $B_{i,2} \geq BT_{i+1,0} > B_{i,1}$	$\hat{e}_i$: $BT_{i+1,0} > B_{i,1} \geq B_{i,2}$	$\hat{f}_i$: $BT_{i+1,0} > B_{i,2} > B_{i,1}$
TLE_i: $T_{i,0}$	BL_i: $B_{i,0} > T_{i,0}$	Φ_1: when in phase Φ_1

Note: $TB_{i,0}$ and $BT_{i,0}$ both represent the larger of $T_{i,0}$ and $B_{i,0}$ at Φ_2, but $TB_{i,0}$ represents $T_{i,0}$ and $BT_{i,0}$ represents $B_{i,0}$ at Φ_1.

Fig. 17 *Signals in slice*

Upper half slice		Lower half slice	
$G_{i,4}$	$(TLE_i + \Phi_1)(a_i + b_i)$	$\widehat{CG}_{i,4}$	$(BL_i + \Phi_1)(\hat{a}_i + \hat{b}_i)$
$G_{i,5}$	$(TLE_i + \Phi_1)(c_i + d_i)$	$\widehat{CG}_{i,5}$	$(BL_i + \Phi_1)(\hat{c}_i + \hat{d}_i)$
$G_{i+1,1}$	$(TLE_{i+1} + \Phi_1)(TLE_i + \Phi_1)(e_i + f_i)$	$\widehat{CG}_{i+1,1}$	$(BL_{i+1} + \Phi_1)(BL_i + \Phi_1)(\hat{e}_i + \hat{f}_i)$
$G_{i+1,2}$	$(TLE_{i+1} + \Phi_1)(TLE_i + \Phi_1)(a_i + b_i)$	$\widehat{CG}_{i+1,2}$	$(BL_{i+1} + \Phi_1)(BL_i + \Phi_1)(\hat{a}_i + \hat{b}_i)$
$G_{i+1,3}$	$(TLE_{i+1} + \Phi_1)(TLE_i + \Phi_1)(c_i + d_i)$	$\widehat{CG}_{i+1,3}$	$(BL_{i+1} + \Phi_1)(BL_i + \Phi_1)(\hat{c}_i + \hat{d}_i)$
$PG_{i,1}$	$TLE_{i+1}BL_i(\hat{e}_i + \hat{f}_i)$	$PG_{i,6}$	$BL_{i+1}TLE_i(e_i + f_i)$
$PG_{i,2}$	$TLE_{i+1}BL_i(\hat{a}_i + \hat{b}_i)$	$PG_{i,5}$	$BL_{i+1}TLE_i(a_i + b_i)$
$PG_{i,3}$	$TLE_{i+1}BL_i(\hat{c}_i + \hat{d}_i)$	$PG_{i,4}$	$BL_{i+1}TLE_i(c_i + d_i)$
$CG_{i+1,1}$	BL_{i+1}	$CG_{i+1,2}$	TLE_{i+1}

Fig. 18 *Combination of signals for gate control*

24–26. This results in a total of $2C_2^{N/2+1} + 1$ comparators idle during the operation of type II MISPQ. If the comparators are shared between the top and bottom subslices, half amount of them can be saved and only $2C_2^{N/2+1} + 1$ comparators are required. Of course, more connection lines are required for those metrics to be transferred to the comparators in the counter part subslice. To be fair, such a comparator-sharing scheme is not considered when comparing the complexities of the three architectures discussed in this paper.

The complexities of type II MISPQ and the two versions of type I MISPQ are shown in Table 1. It is found

Upper half slice			Lower half slice		
	functions at Φ_1	functions at Φ_2		functions at Φ_1	functions at Φ_2
$G_{i,4}$	transfer $T_{i,1}$ to $T_{i+1,0}$ or to $T_{i+1,1}$	transfer $T_{i,1}$ to $T_{i,0}$ or loop $T_{i,1}$ back to itself	$\hat{G}_{i,4}$	transfer $B_{i,1}$ to $B_{i+1,0}$ or to $B_{i+1,1}$	transfer $B_{i,1}$ to $B_{i,0}$ or loop $B_{i,1}$ back to itself
$G_{i,5}$	transfer $T_{i,2}$ to $T_{i+1,0}$ or to $T_{i+1,2}$	transfer $T_{i,2}$ to $T_{i,0}$ or loop $T_{i,2}$ back to itself	$\hat{G}_{i,5}$	transfer $B_{i,2}$ to $B_{i+1,0}$ or to $B_{i+1,2}$	transfer $B_{i,2}$ to $B_{i,0}$ or loop $B_{i,2}$ back to itself
$G_{i+1,1}$	loop $T_{i+1,0}$ back to itself or no operation	transfer $T_{i+1,0}$ to $T_{i,0}$ or no operation	$\hat{G}_{i+1,1}$	loop $B_{i+1,0}$ back to itself or no operation	transfer $B_{i+1,0}$ to $B_{i,0}$ or no operation
$G_{i+1,2}$	transfer $T_{i+1,0}$ to $T_{i+1,1}$ or no operation	transfer $T_{i+1,0}$ to $T_{i,1}$ or no operation	$\hat{G}_{i+1,2}$	transfer $B_{i+1,0}$ to $B_{i+1,1}$ or no operation	transfer $B_{i+1,0}$ to $B_{i,1}$ or no operation
$G_{i+1,3}$	transfer $T_{i+1,0}$ to $T_{i+1,2}$ or no operation	transfer $T_{i+1,0}$ to $T_{i,2}$ or no operation	$\hat{G}_{i+1,3}$	transfer $B_{i+1,0}$ to $B_{i+1,2}$ or no operation	transfer $B_{i+1,0}$ to $B_{i,2}$ or no operation
$PG_{i,1}$	always open, no operation	transfer $T_{i+1,0}$ to $B_{i,0}$ or no operation	$PG_{i,6}$	always open, no operation	transfer $B_{i+1,0}$ to $T_{i,0}$ or no operation
$PG_{i,2}$	always open, no operation	transfer $T_{i+1,0}$ to $B_{i,1}$ or no operation	$PG_{i,5}$	always open, no operation	transfer $B_{i+1,0}$ to $T_{i,1}$ or no operation
$PG_{i,3}$	always open, no operation	transfer $T_{i+1,0}$ to $B_{i,2}$ or no operation	$PG_{i,4}$	always open, no operation	transfer $B_{i+1,0}$ to $T_{i,2}$ or no operation
$CG_{i+1,1}$	always open, no operation	loop $T_{i+1,0}$ back to itself or no operation	$CG_{i+1,2}$	always open, no operation	loop $B_{i+1,0}$ back to itself or no operation

Notes: (i) First function for each gate control signal is function when that signal is true, and second function (i.e. function described after 'or ...') is when it is false; (ii) 'No operation' means gates controlled by those signals can be neglected at that time

Fig. 19 *Functions of control gates*

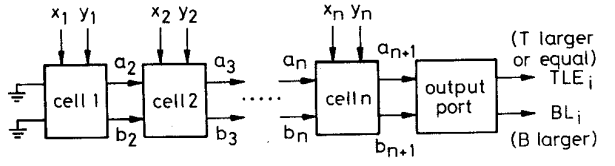

Fig. 20 *Comparator for comparing $T_{i,0}$ and $B_{i,0}$: architecture for iteration realisation*

Metric of $T_{i,0} = x_1, x_2, \ldots, x_n$; metric of $B_{i,0} = y_1, y_2, \ldots, y_n$

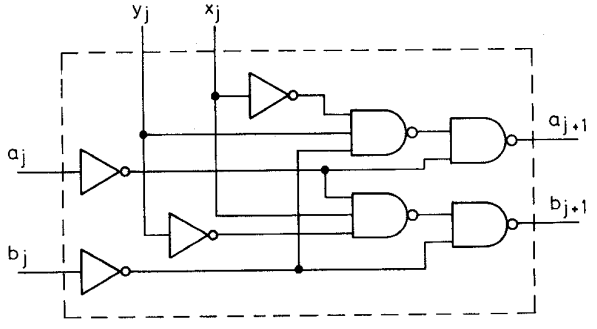

Fig. 21 *Typical cell for one bit*

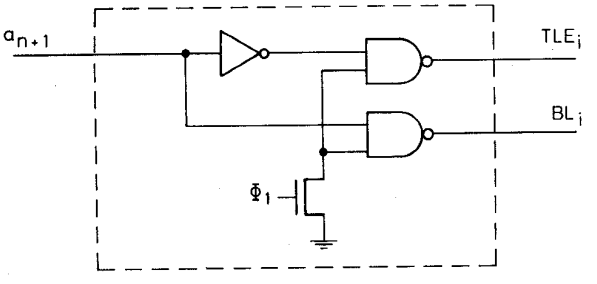

Fig. 22 *Output port of Fig. 20*

a_ib_i	$a_{i+1}b_{i+1}$				Condition until this stage
	$x_iy_i = 00$	01	11	10	
00	00	10	00	01	$T = B$
01	01	01	01	01	$T > B$
10	10	10	10	10	$T < B$

Fig. 23 *Truth tabel for Figs. 20–22*

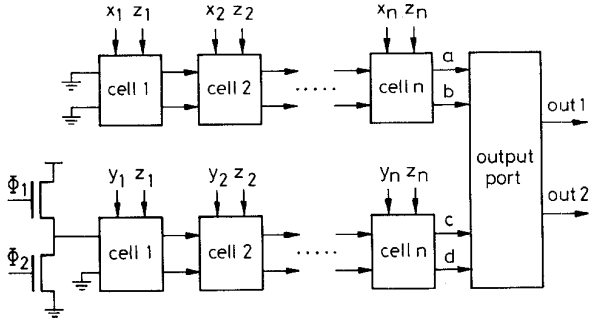

Fig. 24 *Comparator architecture for comparing the larger of $T_{i,0}$ and $B_{i,0}$ with $T_{i-1,j}$ ($j = 1, 2, \ldots, n/2$)*

Comparator is also used to compare $B_{i,0}$ and $T_{i,0}$ with $B_{i-1,j}$, depending on where comparator is located. Metric of $T_{i,0}$ is $x_1, x_2, \ldots, x_n$, metric of $B_{i,0}$ is $y_1, y_2, \ldots, y_n$, metric of $T_{i-1,j}$ is $z_1, z_2, \ldots, z_n$

out1 / out2	Condition
00	max ($T_{i,0}$, $B_{i,0}$) is equal to $T_{i-1,j}$
01	max ($T_{i,0}$, $B_{i,0}$) is larger than $T_{i-1,j}$
10	max ($T_{i,0}$, $B_{i,0}$) is smaller than $T_{i-1,j}$

Fig. 25 *Output conditions for Fig. 24*

that the type II MISPQ is preferable when the number of inputs $N \geqslant 16$. In the case of 16 inputs, the numbers of comparators for the type II MISPQ and the modified type I MISPQ are similar (145 : 136), and the numbers of

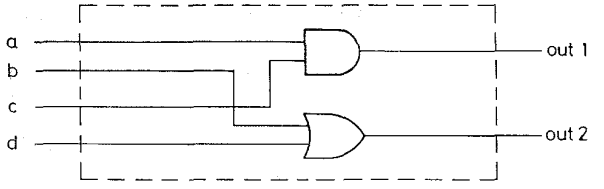

Fig. 26 *Output port for Fig. 24*

Table 1: Complexity comparisons for three circuits

	Type I MISPQ	Modified type I MISPQ	Typ II MISPQ
Comparators	C_2^{N+1}	C_2^{N+1}	$4C_2^{N/2+1}+1$
Controlled transmission gates	$(N+1)^2$	$3N+1$	$3N+10$
Signals for connected to one transmission gate	N	N	$(N/2)+2$ or $(N/2)+4$

transmission gates for these two circuits are similar (58 : 49). But the number of inputs for controlling a transmission gate in the type II MISPQ is about half of that in the modified type I MISPQ. For the case of eight inputs, the selection between these two schemes is not so straightforward. But, generally speaking, the modified type I MISPQ is preferred in this case. This is because the numbers of comparators and transmission gates are lesser (41 : 36 and 34 : 25), and the number of inputs for controlling a gate is similar to that for type II MISPQ. For the cases where inputs are smaller than eight, the modified type I MISPQ is preferred.

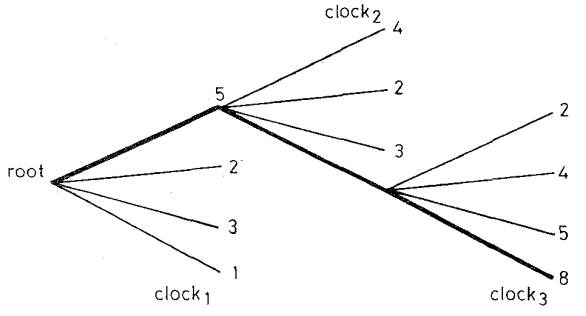

Fig. 27 *Sequential search on quaternary tree*

An example demonstrates the operations of the type II MISPQ. Fig. 27 shows the quaternary tree on which the sequential search is conducted. The metric of each node is written on the tree. These metrics will be stored in and/or extracted from the MISPQ during the sequential search. The clock cycles are also labelled on the tree to denote the sequence of search operations. In Fig. 28, from step (*a*) to step (*c*), the metrics distributions in the MISPQ for each clock cycle during the sequential search are shown. For convenience's sake, processors in the MISPQ are replaced by blocks and the connection lines are neglected. In the following, we describe the transmission procedure for each metric in detail so that the circuit of MISPQ can be verified by following this example with reference to Figs. 16–19.

As shown in Fig. 28, the blocks correspond to the circuit in Fig. 16 with $i = 1$, where $T_{0,0}$ and $B_{0,0}$, not shown, are used as output ports. Processors $T_{0,j}$ and $B_{0,j}$ ($j = 1, 2$) are used as input ports. Those processors behind (and including) $T_{1,0}$ and $B_{1,0}$ form the real body of a systolic priority queue. Note that those transmission gates $PG_{0,1}$ to $PG_{0,6}$ are replaced by a simple switch controlled by TLE_1 and BL_1.

Referring to step (*a*) in Fig. 28, the queue is filled with the smallest metrics at the beginning, in this example -1 is used. At phase Φ_1 of any clock, the plug-in gates $PG_{1,j}$ (for $j = 1$ to 6) are opened as can be checked from Fig. 18, because the signals BL_i and TLE_i (for $i = 1, 2, \ldots$) are always forced to be 0 (false) at phase Φ_1. Thus from $clock_1$ to $clock_3$, when at phase Φ_1, one can treat the processors in the top subslices and those in the bottom subslices as two independent groups. At Φ_1 of $clock_1$, the metric 5 in $T_{0,1}$ is transferred to $T_{1,0}$ through the gate $G_{0,4}$. And the metric -1 in $T_{1,0}$ is transferred to $T_{1,1}$ through the gate $G_{1,2}$. The metric 2 in $T_{0,2}$ is transferred to $T_{1,2}$ through the complement gate of $G_{0,5}$. At the bottom, the metric 3 in $B_{0,1}$ is transferred to $B_{1,0}$ through the gate $\hat{G}_{0,4}$. And the metric -1 in $B_{1,0}$ is

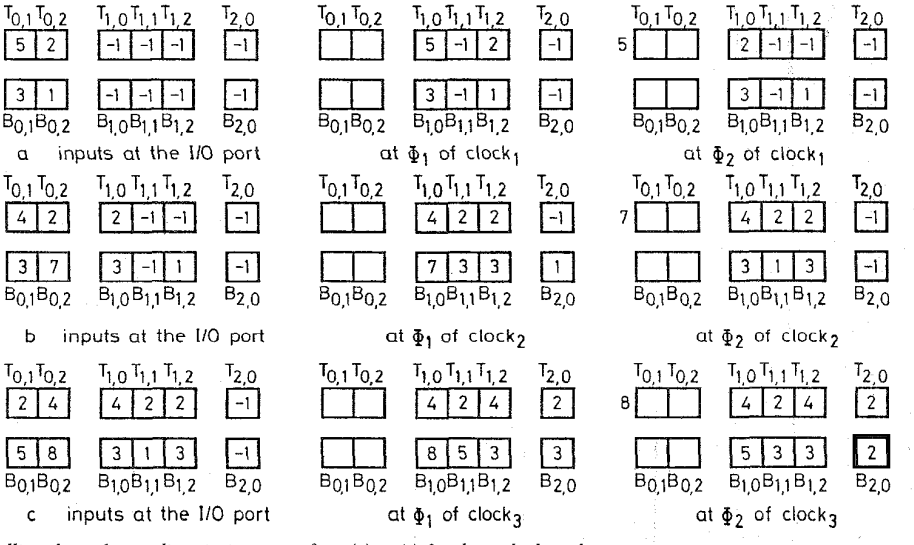

Fig. 28 *Metrics travelling through systolic priority queue from (a) to (c) for three clock cycles*

(*a*) inputs at I/O port at Φ_1 of $clock_1$ at Φ_2 of $clock_1$
(*b*) inputs at I/O port at Φ_1 of $clock_2$ at Φ_2 of $clock_2$
(*c*) inputs at I/O port at Φ_1 of $clock_3$ at Φ_2 of $clock_3$

transferred to $B_{1,1}$ through the gate $\hat{G}_{1,2}$. The metric 1 in $B_{0,2}$ is transferred to $B_{1,2}$ through the complement gate of $\hat{G}_{0,5}$. Although those -1 metrics will also be redistributed following the rules defined by the control signals; it is trivial to describe the details.

At Φ_2 of clock$_1$, the metric 5 in $T_{1,0}$ is transferred to $T_{0,0}$ (extracted out) through the gate $G_{1,1}$. The metric 2 in $T_{1,2}$ is transferred to $T_{1,0}$ through the gate $G_{1,5}$. The metric -1 in $T_{2,0}$ is transferred to $T_{1,2}$ through the gate $G_{2,3}$, since metric in $T_{2,0}$ is equal to that in $B_{2,0}$ so that metric in $T_{2,0}$ is transferred (TLE_2 is true). Metric -1 in $T_{1,1}$ is looped back to itself through the complement gate of $G_{1,4}$. Furthermore, since the result of comparing $T_{1,0}$ (5) and $B_{1,0}$ (3) makes BL_1 to be 0 (false), the metrics 3, -1, and 1 in processors $B_{1,0}$, $B_{1,1}$, and $B_{1,2}$ are forced to loop back to themselves through the gate $CG_{1,2}$, the complement gate of $\hat{G}_{1,4}$, and the complement gate of $\hat{G}_{1,5}$.

Referring to step (*b*) of Fig. 28, at Φ_1 of clock$_2$, four inputs are inserted into the queue again. At this time, the metric 4 in $T_{0,1}$ is transferred to $T_{1,0}$ through the gate $G_{0,4}$. And the metric 2 in $T_{1,0}$ is transferred to $T_{1,1}$ through the gate $G_{1,2}$. The metric 2 in $T_{0,2}$ is transferred to $T_{1,2}$ through the complement gate of $G_{0,5}$. At the bottom, the metric 7 in $B_{0,2}$ is transferred to $B_{1,0}$ through the gate $\hat{G}_{0,5}$, and the metric 3 originally in $B_{1,0}$ is transferred to $B_{1,2}$ through the gate $\hat{G}_{1,3}$. The metric 1 originally in $B_{1,2}$ will be transferred to $B_{2,0}$ through the gate $\hat{G}_{1,5}$. The metric 3 in $B_{0,1}$ is transferred to $B_{1,1}$ through the complement gate of $\hat{G}_{0,4}$.

At Φ_2 of clock$_2$, the metric 7 in $B_{1,0}$ is transferred to $T_{0,0}$ (extracted out) through the gate $\hat{G}_{1,1}$ (and a selection switch not shown), because BL_1 is true. The metric 3 in $B_{1,1}$ is transferred to $B_{1,0}$ through the gate $\hat{G}_{1,4}$. The metric 1 in $B_{2,0}$ is transferred to $B_{1,1}$ through the gate $\hat{G}_{2,2}$, because the comparison of $T_{2,0}$ (-1) and $B_{2,0}$ (1) makes the signal BL_2 be 1 (true). Metric 3 in $B_{1,2}$ is looped back to itself through the complement gate of $\hat{G}_{1,5}$. Furthermore, since the result of comparing $T_{1,0}$ (4) and $B_{1,0}$ (7) makes TLE_1 to be 0 (false), the metrics 4, 2, and 2 in processors $T_{1,0}$, $T_{1,1}$, and $T_{1,2}$ are forced to loop back to themselves through the gate $CG_{1,1}$, the complement gate of $G_{1,4}$, and the complement gate of $G_{1,5}$.

Referring to step (*c*) of Fig. 28, at Φ_1 of clock$_3$, four inputs are inserted into the queue. The metric 8 in $B_{0,2}$ is transferred to $B_{1,0}$ through the gate $\hat{G}_{0,5}$, and the metric 3 originally in $B_{1,0}$ is transferred to $B_{1,2}$ through the gate $\hat{G}_{1,3}$. The metric 3 originally in $B_{1,2}$ is transferred to $B_{2,0}$ through the gate $\hat{G}_{1,5}$. The metric 5 in $B_{0,1}$ is transferred to $B_{1,1}$ through the complement gate of $\hat{G}_{0,4}$. At the top, the metric 4 in $T_{0,2}$ is to be compared with the metric 4 originally in $T_{1,0}$ and the metric 2 in $T_{0,1}$. At this point, the importance of precisely defining the relationships and positions for all processors is clear. Since the two best metric are the same, how the data will flow depends completely on the definitions. According to control rules shown in Fig. 18, the condition d_0 is true, thus the metric 4 in $T_{0,2}$ will be transferred to $T_{1,0}$ through $G_{0,5}$. And the metric 4 originally in $T_{1,0}$ will be transferred to $T_{1,2}$ through the gate $G_{1,3}$. The metric 2 in $T_{0,1}$ is transferred to $T_{1,1}$ through the complement gate of $G_{0,4}$. The metric 2 originally in $T_{1,1}$ is transferred to $T_{2,0}$ through the gate $G_{1,4}$. Again, the definitions determined the flow paths for those data originally in $T_{1,1}$ and $T_{1,2}$.

At Φ_2 of clock$_3$, the metric 8 in $B_{1,0}$ is transferred to $T_{0,0}$ (extracted out) through the gate $\hat{G}_{1,1}$ (and a selection switch not shown). The metric 5 in $B_{1,1}$ is transferred to $B_{1,0}$ through the gate $\hat{G}_{1,4}$. The metric 3 in $B_{2,0}$ is transferred to $B_{1,1}$ through the gate $\hat{G}_{2,2}$, at the same time, the metric 2 in $T_{2,0}$ is looped back to itself through the gate $CG_{2,2}$. Since the result of comparing $T_{1,0}$ (4) and $B_{1,0}$ (8) makes TLE_1 to be 0 (false), the metrics 4, 2, and 4 in processors $T_{1,0}$, $T_{1,1}$, and $T_{1,2}$ are forced to loop back to themselves through the gate $CG_{1,1}$, the complement gate of $G_{1,4}$, and the complement gate of $G_{1,5}$. The metric 2 in $B_{2,0}$ comes from $T_{3,0}$ not shown in Fig. 28. If the systolic priority queue contains only eight elements as shown in Fig. 28, this metric will be lost forever.

Other new architectures may be developed following the spirit of type II MISPQ. The MISPQ may be implemented with smaller queues, where each queue is used for processing a part of data in this MISPQ. For example, an eight-input systolic priority queue can be implemented with two four-input systolic priority queues. Also, it may be implemented with four parallel-entry systolic priority queues. Following this idea, one can develop the type III MISPQ, type IV MISPQ, and so on. Although this idea may be extended further so that the control signals in each subslice and the comparison operations will be less sensitive to the value of N, care must be taken when considering the actual benefits generated by using such architecture. This is because, as the number of subslices in a slice increases, the number of transmission gates for the interconnections between these subslices also increases. For example, if a four-input systolic priority queue is implemented with four type I single-input SPQs, there are 24 transmission gates between these queues, a large number when considering the complexity of these single-input SPQs. On the other hand, if 16-input SPQ is implemented, it is a good choice to divide it into four type I four-input SPQs.

5 Conclusion

A new algorithm and architecture for multiple inputs systolic priority queue (MISPQ) has been presented. This new architecture is built in a hierarchical form, i.e., one NISPQ consists of several smaller queues as basic components. Compared withthe previously developed systolic priority queues [3–6], this MISPQ is promising when $N \geqslant 16$. By modifying the previous circuit for systolic priority queue [5, 6], a circuit for type I MISPQ has also been developed. The number of controlled transmission gates for this circuit is proportional to N instead of N^2 for the original circuit. This modified circuit is shown to be suitable for implementing MISPQ when N is eight or smaller. From the analysis in Section 3.2 it is found that the benefits of the type II MISPQ are corrupted because of use of additional $2C_2^{N/2+1}$ comparators when comparing $TB_{i+1,0}$ or $TB_{i+1,0}$ with $T_{i,j}$ or $B_{i,j}$. Thus the well defined comparator sharing schemes or multiple-input comparators are desired for future studies for the MISPQ with smaller N.

Notice that in a sequential decoder, each node on the tree may contain information bits ranging from 30 to 70. Although some information bits may be stored in the RAM so that only the metric and the address in the RAM are stored by the systolic priority queue [6], the number of bits may still be large. For such cases, how to arrange the I/O pins becomes a problem. By implementing the queue containing only the metrics on one chip and the queues containing other bits on other chips may be a solution. However, the complexity will increase and

the operating speed become slower since the control signals should be transferred between chips.

6 References

1 GUIBAS, L.J., and LIANG, F.M.: 'Systolic stacks, queues, and counters'. Conference on *Advanced research in VLSI*, MIT, 1982

2 LEISERSON, C.E.: 'Systolic priority queue'. Proceedings of Caltech conference on *VLSI*, 1979

3 CHANG, C.Y.: 'Systolic array architecture for convolutional decoding algorithms: Viterbi algorithm and stack algorithm'. PhD dissertation, University of California, Los Angeles, 1986

4 CHANG, C.Y., and YAO, K.: 'Systolic array architecture for the sequential stack decoding algorithm', *Proc. SPIE*, 1986, **696,** pp. 196–203

5 LAVOIE, P., BELZILE, J., TOULGOAT, M., HACCOUN, D., and SAVARIA, Y.: 'VLSI design of a systolic priority queue chip for sequential decoders'. Proceedings of 1988 Canadian conference on *VLSI* (Halifax, Nova Scotia, Canada), 1988, pp. 1–9

6 LAVOIE, P., HACCOUN, D., and SAVARIA, Y.: 'A systolic architecture for fast stack sequential decoders', *IEEE Trans.*, 1994, **COM-42,** pp. 324–335

7 BELANGER, N., HACCOUN, D., and SAVARIA, Y.: 'A multiprocessor architecture for multiple path stack sequential decoders', *IEEE Trans.*, 1994, **COM-42,** pp. 951–957

8 KUO, H.C., and WEI, C.H.: 'Sequential decoding of convolutional codes by a compressed multiple queue algorithm', *IEE Proc.-Comm.*, 1994, **141,** (4), pp. 212–222

9 ROTH, C.H.: 'Fundamentals of logic design'. West Publishing Company, 1985

10 LIN, S., and COSTELLO, D.J.: 'Error control coding: fundamentals and applications'. Prentice-Hall, New Jersey, 1983