A Hybrid Wave Pipelined Network Router

Jabulani Nyathi, *Member, IEEE,* and José G. Delgado-Frias, *Senior Member, IEEE*

***Abstract*—In this paper, a novel hybrid wave pipelined bit-pattern associative router (BPAR) is presented. A router is an important component in communication network systems. The BPAR allows for flexibility and can accommodate a large number of routing algorithms. In this study, a hybrid wave pipelined approach has been proposed and implemented. Hybrid wave pipelining allows for the reduction of the delay difference between the maximum and minimum delays by narrowing the gap between each stage of the system. This approach yields narrow "computing cones" that could allow faster clocks to be run. This is the first study in wave pipelining that deals with a system that has substantially different pipeline stages. The BPAR has three stages: condition match, selection function, and port assignment. In each stage, data delay paths are tightly controlled in order to optimize the proper propagation of signals. Internal control signals are generated to ensure that data propagates between stages in a proper fashion. Results from our study show that using a hybrid wave pipelining significantly reduces the clock period. The hybrid wave pipelined system described in this paper has been fabricated using a 0.5-μm technology.**

***Index Terms*—Bit-pattern associative memory, computer network address decoder, delay balancing, dynamic ternary content addressable memory, hybrid wave pipelining, wave pipelined clock.**

I. Introduction

COMMUNICATION between any two network nodes is usually achieved by means of network routers. The function of a network router is to receive, forward, and deliver messages. The router system transfers messages based on a routing algorithm which is a crucial element of any communication network [1], [2]. Given the reconfiguring requirements for many network systems, a number of routing algorithms as well as network topologies must be supported. Thus, there is a need for a very high performance flexible router to support these requirements. Having the capability of changing the routing algorithm at run time could facilitate smart interconnects and adaptability that allow changes on the network topology for different applications.

A router intended for a number of routing algorithms and/or network topologies must accommodate a number of routing requirements. It is of great importance that the routing algorithm execution time be extremely short. This time dictates how fast a message can advance through the network, since a message cannot be transferred until an output port has been selected by the routing algorithm. Thus, the routing algorithm execution time must be reduced to decrease message delays. Other requirements may include flexibility to accommodate modifications to a network, algorithm and/or topology switching with minimum delay, and programmability to support a large number of routing algorithms and network topologies.

The major approaches to the realization of flexible routers are dedicated processors and lookup table [3]. A dedicated processor allows bit manipulation, required by most algorithms. This type of router is capable of handling a number of networks, but due to its sequential execution of routing algorithms it is a slow approach. The lookup table approach requires the storage in memory of a predetermined routing path for all possible destination nodes. The lookup-table approach reduces the time of determining the output port to a memory access delay, however, large tables are required to store the routing information. A third approach called bit pattern uses an associative approach to determine in parallel, the potential output ports (according to the routing algorithm). The output port is selected by means of a selection function. This scheme has been shown to accommodate a large number of routing algorithms and to have fast execution [3], [4].

In this paper, we present a high-performance hybrid wave pipelined VLSI router based on the bit-pattern approach. Wave pipelining enables pipelining in logic without the use of intermediate registers. In order to realize practical systems using wave pipelining, it is a requirement that accurate system-level and circuit-level timing analysis be done. At system level, generalized timing constraints for proper clocking and system optimization need to be considered. At the circuit level, performance is determined by the maximum circuit-delay difference in propagating signals within a given module. Accurate analysis and strict control of these delays are required in the study of worst-case delay paths of circuits. In this study, a different approach to minimizing the clock period is undertaken. It is termed *hybrid wave pipelining* and hinges on wave pipelining.

We begin by presenting the router's functional organization in Section II, describing the basic circuits of each major module. In Section III, we introduce wave pipelining and derive some equations that describe the hybrid wave-pipelining timing constraint. The design of the hybrid wave pipelined router, its timing requirements, control circuits and simulation results are discussed in Section IV. In Section V, a summary of the delays associated with each module are graphically represented and some concluding remarks appear in Section VI.

II. Router Functional Organization

The bit-pattern associative router (BPAR) scheme supports the execution of routing algorithms that are used in most communication switches. In this section, an overall description

Manuscript received June 14, 2001; revised January 18, 2002. This work was supported in part by the National Science Foundation under Contract CCR9900643. This paper was recommended by Associate Editor Y. Ismail.

The authors are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752 USA (e-mail: jdelgado@eecs.wsu.edu).

Digital Object Identifier 10.1109/TCSI.2002.805705

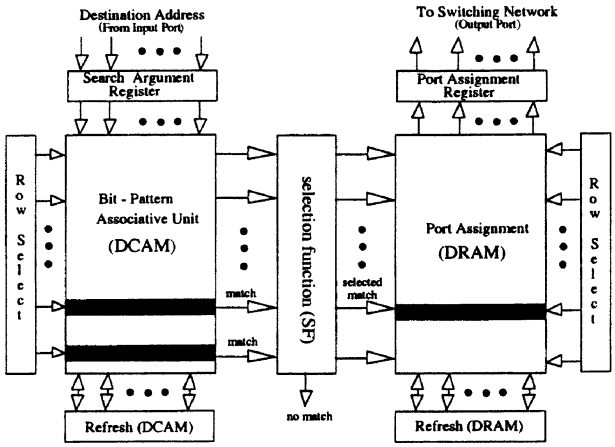

Fig. 1. BPAR organization.

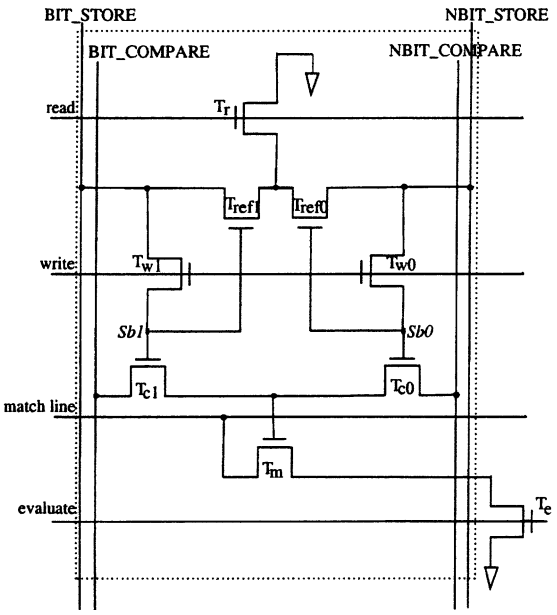

Fig. 2. CMOS circuit for DCAM cell.

of the router organization shown in Fig. 1 is presented. This description is at a functional level, in order to illustrate the requirements and capabilities of the system.

The associative router uses a content addressable memory as its bit-pattern associative unit, and this enables the destination address alternatives to be considered in parallel. The destination address is presented as the input to the bit-pattern associative unit (BPAU) for comparison with the stored data. The patterns stored in the BPAU allow the router to make a decision about the destination port based on the routing algorithm. The address of a destination node $D\ (d_{n-1}, \ldots d_0)$ needs to be compared to the current node's address $C\ (c_{n-1}, \ldots c_0)$. A routing algorithm compares the bits of the two addresses; some bits are ignored since they do not affect the current routing decision. These "don't care" bits usually occur at different positions for each potential path being considered and need to be customized according to the routing algorithm requirements. This in turn imposes a requirement for the BPAU to include "don't care" bits at different positions in each entry. To provide the flexibility required to support multiple interconnection networks and routing algorithms, the BPAR must be programmable. The results of the comparison are passed to the selection function, which then passes the match output with the highest priority to the port assignment to select the word corresponding to the selected output port.

Modules of the BPAR include the following.

- *Search argument register (SAR):* The SAR receives the destination address and passes it to the next stage at the appropriate time.
- *BPAU:* This unit stores the bit patterns for a given interconnection network node and performs a comparison between the destination address and the current address (specified by the bit patterns). All the patterns that match the input address are passed to the following stage; the selection function.
- *Selection function:* The selection function is a priority encoder that selects only one of the BPAU outputs. If more than one pattern has been matched, the one with the highest priority gets selected [5]. The selected match pointer gets passed to the next stage. If no matching pattern is found the selection function sets a no match signal.
- *Port assignment:* This is a memory that stores the output port assignments. The address where the assignment is read is specified by a pointer from the selection function and this assignment is passed to the port assignment register.
- *Port assignment register:* This register holds the assignment which is used to send the message.
- *Row select:* This register is used as a pointer to a row of dynamic CAMs (DCAMs) (BPAU array) and dynamic RAMs (DRAMs) (port assignment) to be programmed to or to be refreshed. It is not used during normal operation.
- *Refresh circuitry.* This circuitry is used for refreshing stored data, since dynamic circuits are used.

The BPAR organization supports three basic operation modes, normal (or matching), programming (or data loading), and refreshing [3]. In normal (matching) mode, the previously loaded data are compared to data presented at the SAR and the results passed to the selection function. The programming mode provides a means by which to initialize the memories with destination addresses, while refreshing enables the replenishing of the loaded data during the normal operation, since dynamic circuitry is being used.

A. DCAM Cell

The DCAM cell implements a comparison between an input and the ternary digit condition stored in the cell. The circuit of a single DCAM cell, shown in Fig. 2, consists of eight and a half transistors; transistor T_e is shared by two cells. The read, write, evaluation, and match line signals are shared by the cells in a word while the BIT_STORE, NBIT_STORE, BIT_COMPARE, and NBIT_COMPARE lines are shared by the corresponding bit in all words of the matching unit. The design uses a precharged match line to allow fast and simple evaluation of the match condition. This condition is represented by the DCAM cell state which is set by $Sb1$ and $Sb0$ node status. This state is stored in the gate capacitance of the transistors T_{c1} and T_{c0}. In Table I, the

TABLE I
REPRESENTATION OF STORED DATA IN A DCAM CELL

$Sb1$	$Sb0$	state
0	0	X(don't care)
0	1	1(one)
1	0	0(zero)
1	1	not allowed

possible DCAM cell stored values are listed. Transistors T_{ref0} and T_{ref1} are used for refreshing.

The DCAM cell performs a match between its stored ternary value and a bit input (provided by BIT_COMPARE and NBIT_COMPARE, which represent a bit and its opposite value, respectively). The match line is precharged to "1." Thus, when a no match exists this line is discharged by means of transistor T_m. Normal operation involves comparing the input data to the patterns stored in the DCAM and determining if a match has been found. During match operation, the input data is presented on the BIT_COMPARE line and its inverse value on the NBIT_COMPARE line. Before the actual matching of these two values is performed, the match line is precharged to "1" which indicates a match condition. The matching of the input data and the stored data is performed by means of an exclusive-or operation which is implemented by the two transistors (T_{c1} and T_{c0}) that hold the stored value.

B. Selection Function and Port Assignment

The selection function should be designed to ensure the deterministic execution of the routing algorithms. With a given input (i.e., destination address) and a set of patterns stored in the matching unit, the port assignment should always be the same. The priority allows only the highest priority pattern that matches the current input to pass on to the port assignment memory. Below is the Boolean expression for the encoded priority

$$EP_i = M_i \cdot P_i$$

where M_i and P_i are the matchline and the priority status of ROW_i, respectively.

A priority lookahead scheme has been proposed and implemented, as reported in [5]. The port assignment memory or RAM holds information about the output port that has to be assigned after a bit pattern that matches the current input is found. This memory is proposed to be implemented using a dynamic approach. The selected row address is passed from the priority encoder. The cells in this row are read and their data is latched in the port assignment register. The DRAM structure is able to perform an OR function per column when multiple RAM rows are selected; this is when multiple matches are passed on. The DRAM structure is explained in [3].

III. HYBRID WAVE PIPELINING

In this section, we describe the timing constraints for conventional, wave, and hybrid wave-pipelining approaches. In conventional pipelining, intermediate latches are used in addition to the input and output registers. Intermediate latches (registers) ensure that when the leading edge of the system clock comes, data get propagated from one stage to the next, in a synchronous manner. In a system setup like this there is only one set of data between register stages. The clock-cycle time in conventional pipelining is governed by the worst-case operation of the stage with the largest delay. This results in all stages being clocked at the rate of the slowest stage. The equation that defines clock-cycle time for conventional pipelining is

$$T_{\text{clk}} > d_{\max(j)} + T_S - T_H + \Delta_{\text{clk}} \tag{1}$$

where $d_{\max(j)}$ is the maximum delay of stage j (the stage with the largest delay), with T_S and T_H being the register setup and hold times respectively. Δ_{clk} is the uncontrolled clock skew.

Wave pipelining is an approach aimed to achieve high performance in pipelined digital systems by removing intermediate latches or registers [6]. Idle time of individual logic gates within combinational logic blocks can be minimized using wave pipelining. Some of the challenges of designing wave pipelined systems are as follows.

- *Preventing intermixing of unrelated data "waves."* There must be no data overrun in each circuit block.
- *Balancing delay paths.* Delay paths must be equalized to reduce major differences between maximum and minimum delays which have a direct impact on the system's performance.

The requirements stated above are the most important design issues in wave pipelining. The timing constraints of the wave-pipelining scheme can easily be understood by studying the time–space diagram shown in Fig. 3. Some variables appearing in Figs. 3 and 4 need to be defined; these are as follows.

$D_{\min}$	Minimum propagation delay through the combinational logic.
$D_{\max}$	Maximum (worst-case) delay in the combinational logic.
T_{clk}	Clock period.
T_S, T_H	Register setup and hold times.
Δ	Constructive clock skew.
Δ_{clk}	Register's worst-case uncontrolled clock skew.
D_R	Register's propagation delay.
T_L	Time at which data at the output register can be sampled.
$d_{\max(i)}$	Maximum delay encountered in propagating data within stage i.
$d_{\min(i)}$	Minimum delay encountered in propagating data within stage i.
$D_{\text{min\_hold}}$	System's overall delay difference.

Due to delay differences between inputs, a stage could start in a path which is not the appropriate one. We refer to this potential problem as false start. Intermediate stages could have false starts with those furthest from the input latches being the most prone to this problem. These problems are avoided using the hybrid wave-pipelining approach by holding signals so that the next stage does not start until all the signals from the previous stage are available. A common engineering practice is to consider the worst-case delay ($D_{\max}$), to ensure that the system runs properly. $D_{\max}$ plays a very important role in the system's performance and safe regions of operation. The shortest delay path ($D_{\min}$), on the other hand, indicates the earliest time the results

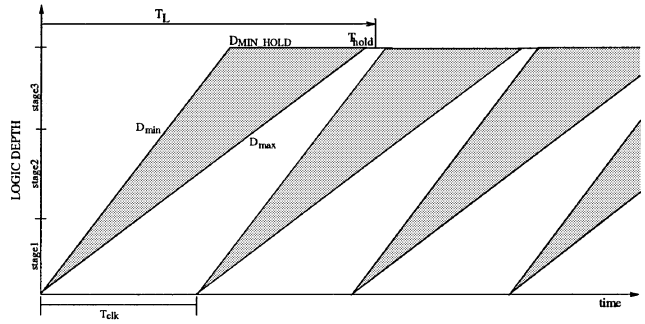

Fig. 3. Temporal/spatial diagram wave pipelined system.

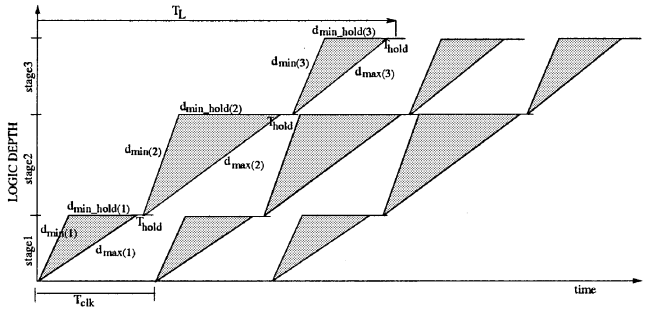

Fig. 4. Temporal/spatial diagram of a hybrid wave pipelined system.

could emerge. In many computer/digital systems each stage has a significantly different function and circuitry; wide variations in delays ($D_{\min}$ and $D_{\max}$) may not be tolerated.

Timing constraints for the proposed hybrid wave-pipelining approach are derived in the same fashion as in [6]–[8]. The equations for the hybrid wave-pipelining scheme are denoted by the subscript h. The hybrid wave-pipelining temporal/spatial diagram is shown in Fig. 4. The computational cones in this diagram have been arranged to represent each stage within the design.

Unlike wave pipelining, the clock-cycle time of the hybrid wave-pipelining scheme is constrained by the delay difference of a stage with the largest delay variation. This raises the need to examine intermediate stages in order to determine the one with the largest delay difference. The wave-pipelining scheme on the other hand has the system's clock-cycle time determined by considering the overall difference between the values of the maximum and minimum delays from the input latch to the output latch. Fig. 4 shows computational cones of the hybrid wave-pipelining scheme and the equations that follow can be derived from this time–space diagram.

For hybrid wave pipelining, the clock-cycle time is not constrained by T_L, the time at which data at the output registers can be sampled, but by how early new data can be admitted into the pipe as dictated by the stage with the largest delay variation. The output-latching time is however still of significant importance and analysis of the hybrid wave-pipelining timing constraints will start by examining T_L. It must be noted, however, that the equation describing the clock-cycle time can be derived without considering the latching time. In the hybrid wave-pipelining scheme, the lower bound of the output latching time T_L for a system with n stages is described as follows:

$$T_{L_h} \geq D_R + \sum_{i=1}^{n} \left(d_{\max(i)} + T_{\text{hold}(i)}\right) \tag{2}$$

where T_{hold} is given by $T_{\text{hold}} = T_S + \Delta_{\text{clk}} - T_H$. The upper-bound of the output latching time is given by

$$T_{L_h} < T_{\text{clk}_h} + D_R + \sum_{i=1}^{n-1} \left(d_{\min(i)} + d_{\text{min\_hold}(i)}\right) + d_{\min(n)}. \tag{3}$$

Given that $d_{\max(i)} = d_{\min(i)} + d_{\text{min\_hold}(i)}$, (3) can be written as

$$T_{L_h} < T_{\text{clk}_h} + D_R + \sum_{i=1}^{n-1} \left(d_{\max(i)} + T_{\text{hold}(i)}\right) + d_{\min(n)}. \tag{4}$$

Equations (2) and (3) take into consideration the intermediate stages of the design. The minimum delays and the hold times of each stage are considered. The effort that goes into minimizing delay differences per stage results in a considerable reduction of the overall system's delay variation. If further derivations are carried out by combining (2) and (4), the clock period for the hybrid wave-pipelining approach is determined to be

$$T_{\text{clk}_h} > \left(d_{\max(n)} - d_{\min(n)}\right) + T_{\text{hold}(n)}. \tag{5}$$

Equation (5) determines the shortest possible clock-cycle time in hybrid wave pipelining, given that stage n has the largest delay difference ($d_{\text{min\_hold}}$). Having the largest $d_{\text{min\_hold}}$ at the end of the pipeline simplifies the analysis, however, the possibility of the largest $d_{\text{min\_hold}}$ being at a stage other than the nth stage must be accounted for. For a case in which the ith stage has the largest delay difference, we introduce the condition

$$T_{\text{clk}_h} > d_{\text{min\_hold(i)}} + T_{\text{hold}(i)}. \tag{6}$$

This condition implies that the stage with the largest delay difference determines the clock-cycle time of the system. The equation that describes the shortest possible clock-cycle time in the wave-pipelining scheme is

$$T_{\text{clk}_w} > (D_{\max} - D_{\min}) + T_{\text{hold}}. \tag{7}$$

The full derivation of (7) can be found in [6] and considers the register constraints based on the overall logic depth without minimizing the intermediate nodes' delay differences. Based on the above analysis it is apparent that the hybrid wave pipelined approach allows for the clock signal's period to be reduced, hence an increase in performance. A close examination of (5) and (7) shows that $T_{\text{clk}_h} < T_{\text{clk}_w}$ since T_{clk_h} is the delay difference of a single stage plus the overhead (T_{hold}) as opposed to the systems' overall delay difference plus the overhead. This guarantees reduction of the time during which data is unstable, hence the sampling window. The comparison of the wave-pipelining and hybrid wave-pipelining schemes is provided in Table II.

TABLE II
CLOCK CYCLE TIME FOR PIPELINING SCHEMES

Wave-pipelining	Hybrid wave-pipelining
Delay balancing is achieved by buffer insertion in the shortest path.	Delay balancing is achieved by the use of latches (buffer insertion helps to further increase performance).
The clock rate is determined by considering the entire system difference $(D_{max} - D_{min})$.	The clock rate is determined by considering the stage with the largest delay variation $max(d_{min\_hold})$.
Data propagation between stages relies not only on the system clock but also on logic and balancing delays.	Data propagation from one stage to the next is a function of the delays through the stage and synchronization with a wave pipelined clock.

IV. HYBRID WAVE-PIPELINED ROUTER

In this section we identify the signals that require synchronization within each module of the hybrid wave pipelined router, showing the timing requirements at each stage and providing the novel circuits designed to provide delay balancing. Also included in this section are simulation results depicting the delays involved within each stage.

A. Timing Requirements

Propagating coherent data waves from one pipeline stage to the next without the use of a distributed system clock is achieved by balancing the delays within each pipeline stage. Delay balancing is necessary since the data within a circuit module takes different paths and experiences different delays depending on the operation, resulting in the data arriving at the input of the next stage at different times. The data that has the shortest delay arrives at the input to the next stage earlier than data that takes the critical path (longest delay), hence the use of intermediate latches between pipeline stages (modules). Burleson *et al.* [6], have suggested *buffer insertion* as a method of delay balancing. Buffers are inserted in the shortest signal path to force these delays to be equal to the worst-case delay, which in turn permits the signals to arrive at the input of the next stage at the same time without the use of intermediate latches nor the system clock. A different approach is considered in this study. It does not involve buffer insertion, but uses the delays in the design of the appropriate signals that will enable data to propagate through the stages coherently without the use of distributed clock nor intermediate latches. The method used will not be limited in use to the BPAR circuitry and should find application in many circuits that employ the wave pipelining technique to improve performance. The circuits designed to enable data to propagate from one stage to the next operate in parallel with the stage circuitry. Thus, no time overhead is introduced by use of the control circuitry designed to enable hybrid wave pipelining. Only one control circuit per stage is used to provide control for all the entries of the particular stage, therefore, hardware overhead is minimal. The area occupied by the control circuits designed to enable hybrid wave pipelining is a mere 3% of the total router chip area.

The difference between the maximum (worst-case) delay and the minimum delay is of particular interest in designing the timing scheme of a wave pipelined BPAR. There will be no buffer insertion to balance the delays, instead circuitry is designed to provide very precise timing sequence. The advantage of this approach is that it is not affected by scaling or use of a different technology since all the modules are affected by these changes the same. The only major limiting factor becomes the worst-case delay of the circuit. The valid clock pulse width (PW) can be made short, and it cannot be less than $D_{\min} + (D_{\max} - D_{\min})$, i.e., $PW \geq D_{\min} + (D_{\max} - D_{\min})$. This expression includes the rise and fall times of the clock and is based on measurements taken from SPICE simulations. This gives the minimum pulse width that will enable application of hybrid wave pipelining to the BPAR without intermixing the data "waves" within any given pipeline stage. This provides the basis for the design of the clock pulses required to cause the data to propagate through the pipeline stages. The system clock passes data from the SAR to the bit-pattern associative unit. It has been established that the BPAR requires the following signals for its operation.

- *Read and write signals for the memories (DCAM and DRAM).* The write signal must be raised to a logic 1 only during programming (storing of current address) and refreshing of the stored data; this is done when the row select signal selects a specific word to write to within the BPAU array. Reading occurs when there is a need to refresh the contents of the words stored within the array. The read and write signals are generated whenever the stored data has been sensed to degrade.
- *Bit lines.* These lines pass the input to the BPAU array during programming as well as normal operation and they need to be precharged before reading from the array. There is thus a need to design dedicated circuitry to generate the precharge signal for the bit lines.
- *The evaluate signal.* The evaluate signal is responsible for determining if a matching condition exists in the bit-pattern associative unit. Dedicated circuitry to provide the proper evaluate signal has to be designed based on $d_{\max(i)}$ and $d_{\min(i)}$, where $d_{\max(i)}$ is the maximum delay path for a signal propagating through stage i and $d_{\min(i)}$ the minimum delay in propagating a signal through stage i of the BPAR.
- *The precharge signal.* The match line of the BPAU is designed to be normally on, and this is achieved by precharging it before evaluation occurs. A precharge signal, therefore, has to be designed and it must occur at appropriate times. The signals $d_{\min(1)}$ and $d_{\max(1)}$ play an important role in the design of the precharge signal for the match line.
- *The signal that permits the data "waves" to be passed to the selection function.* The signal that allows the match output to be passed to the selection function is generated by using the evaluate signal and sizing transistors with the threshold voltage of the transistor used to latch the match outputs.
- *Precharge for the selection function.* This signal is simply the inverse of the evaluate signal.
- *The priority status signal.* Within the selection function the priority status signal is of importance. If a match has been found in any of the entries other than the last entry of

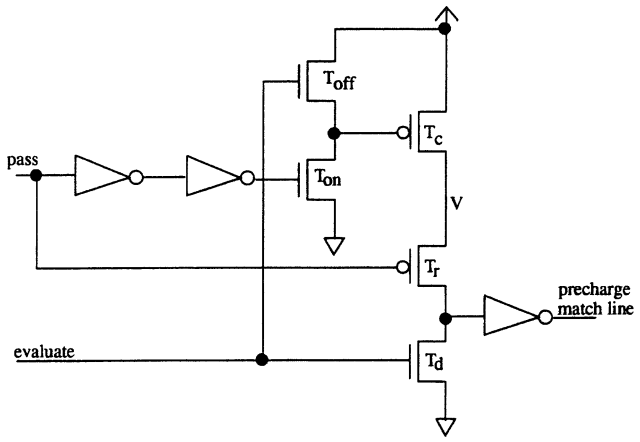

Fig. 5. Generating the match line precharge signal.

the matching unit, the selection function has to detect it as a match with the highest priority and must then propagate a priority status indicating to the entries below that a match has been found. In this design a priority status $P_i = 0$ indicates that a match with the highest priority has been found and hence kills all the other possible matches below the current one. The delays experienced by this signal are of importance since it can be used to determine when new input can be applied to the SAR.

- *The signal required to pass the selected match output to the port assignment.* This signal is required to gate the selected match output determined to have the highest priority by the selection function to the next stage. The system clock is not used in the selection function, therefore, the signals already generated have to be used to provide precise timing to pass the selected match output to the port assignment memory. This signal serves to select the appropriate output port assignment.

The signals described above form the basis for proper timing of the hybrid wave pipelined BPAR. The primary objective is to reduce clock-cycle time by allowing the delays to propagate data from stage to stage with the use of neither intermediate latches nor distributed clocks. As clock rates continue increasing, clock distribution is becoming a major problem; wave pipelining could alleviate this problem by significantly reducing the clock load as well as the number of inter-stage latches [9].

B. BPAU Wave Signals

The requirement for precharging the match line is that this occurs before evaluation takes place and after the output of the match line has been passed to the next stage. As a result the precharge signal is designed using the signals used to evaluate and pass the match line output to the next stage. The circuit in Fig. 5 is designed to generate the match line precharge signal. This arrangement ensures that the precharge signals will operate synchronously with the wave.

The evaluate signal must go to "1" after the data passed to the DCAM for comparison has stabilized on the bit lines (BIT_COMPARE and NBIT_COMPARE). There is circuitry that generates this signal once all the lines have been set to their appropriate input values. This circuit is shown in Fig. 6. The inverse of the clock

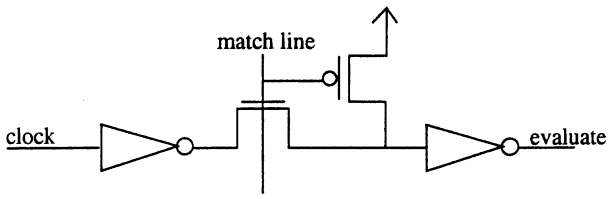

Fig. 6. Generating the evaluate signal.

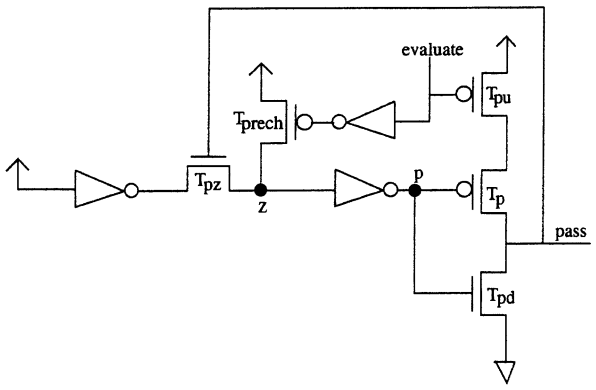

Fig. 7. Generating the pass signal.

is passed to an intermediate node when the match line is at "1"; this intermediate node is precharged to "1" whenever the match line is at "0." The inverse of the intermediate node is evaluate signal. Using the match line to pass the clock signal and set the evaluate signal to "0" when evaluation is completed provides the proper duration for the evaluate signal to stay at "0" or "1." Passing the clock signal when the match line is at logic 1 basically determines that the data on the bit line has had enough time to stabilize on the bit lines.

Once evaluation has been completed the match line outputs can be passed to the selection function. Thus, the pass signal must immediately be active following completion of the evaluation process. The circuit used to generate the pass signal is shown in Fig. 7. The pass signal is designed to mimic the path that a zero on the match line (nonmatching condition) takes once evaluation completes. Passing a "0" to the selection function provides the maximum delay that can be experienced in passing the matching unit's outputs to the selection function and, therefore, constitutes the worst-case propagation delay for this operation. The circuits in Figs. 6 and 7 have been designed to sense the duration and voltage levels of these signals. All the signals presented to this point depend on the system clock.

Another signal of importance is within the selection function. This is the signal that enables the port assignment select signals, i.e., the output of the selection function; the pointer to the selected word of the port assignment. It is required that if a match with the highest priority has been established, the ones below this match output must be disabled. The priority status performs this function along with an enable signal, which has been designed based on the fact that it takes longer to pass "0" from the bit-pattern associative unit. The required delay between sets of four selection function cells is arrived at by inserting delay elements within the circuit. This approach ensures the prevention of false starts. The circuit designed to generate the *enable* signal is shown in Fig. 8.

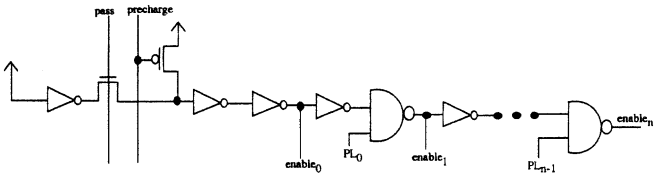

Fig. 8. Enabling the selection-function outputs.

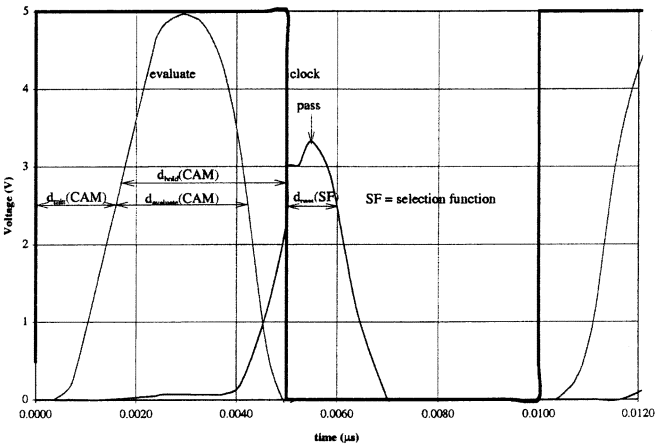

Fig. 9. Plot of BPAU control signals.

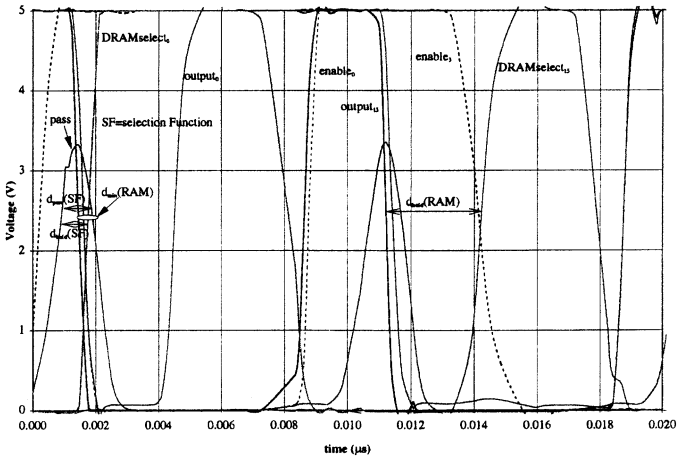

Fig. 10. Plots of enable signals.

C. Hybrid Wave-Pipelining Signals

The signals generated by the above circuits appear in Fig. 9 along with the clock. The delays of the first two stages are clearly marked in the figure to show a correlation with the hybrid wave pipelined approach. On Fig. 9 the minimum delays, $d_{\min}$ and hold times d_{hold} of each stage are shown. Also the duration of the evaluate and pass signals are depicted. Using 0.5-μm technology, 2.5 V is sufficient to determine a logic 1.

Once the output of the match lines have been received by the selection function, a decision needs to be made when more than one matching condition has been received. The selection function is designed to propagate a priority status P_i to the entry below it indicating whether it has registered a match. This priority must be propagated very fast to prevent false starts. Some of the signals of importance in this scheme are shown in Fig. 10. The selection function's critical operation occurs when the first

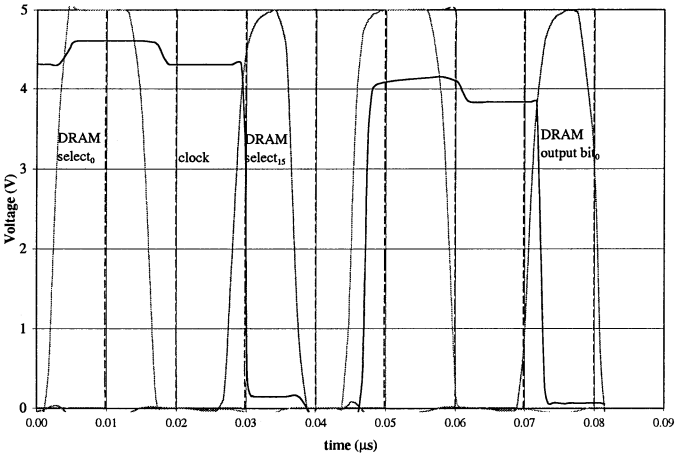

Fig. 11. Plot of DRAM pointers.

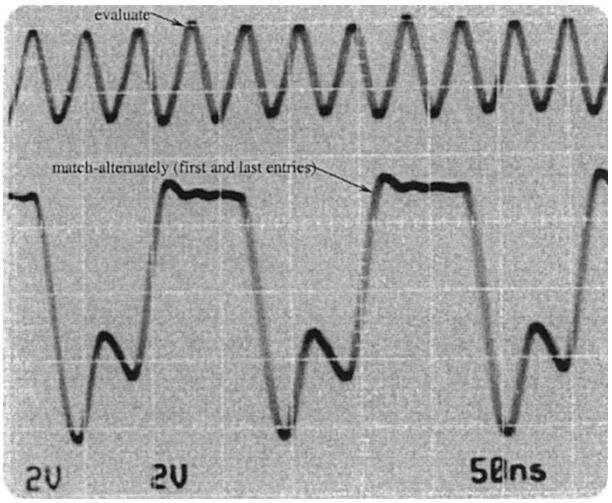

Fig. 12. Matching the first and last entries alternately.

and last entries of the selection function simultaneously receive inputs indicating that matching conditions have been found in the corresponding BPAU entries. The first entry of the selection function has to propagate a "0" to the last entry of the selection function to prevent generation of a pointer to the DRAM by the last entry. Signals to lessen the possibility of false starts are generated and these are labeled *enable*. They are designed to ensure that even if a false start occurs the pointer to the DRAM array is not established. In Fig. 10, delays from the second stage of the BPAR and those of the third stage are shown along with signals used to enable the DRAM pointers for the first and last entries from a setup in which the last entry always finds a match and the first entry finds a match every other clock cycle. The delays involved are shown. The plots in Fig. 11 are those of the DRAM pointers to the first and last entries of the DRAM. They serve to show that an output port assignment can be read from the DRAM array every one and a half clock cycles. Also ensuring that the matching condition occurs every other clock cycle for the first entry of the BPAU, while the last entry always finds a match, makes for a good check for false starts. These tests were also verified through the testing of the fabricated chip and some of the test results appear in Fig. 12.

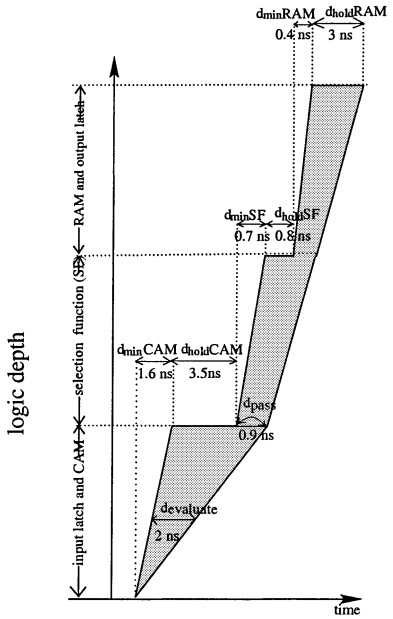

Fig. 13. Hybrid wave pipelined delays translated to computational cone.

V. BPAR Computational Cones

Fig. 13 shows the delays associated with each stage (as shown in Figs. 9–11). The *computational cone* (Fig. 13) show the three stages of the BPAR. Hybrid wave pipelining has enabled for the delay difference per stage to be reduced. Unlike wave pipelining where $D_{\max}$ for the system provides the upper bound for the clock period, hybrid wave pipelining shows that each stage has it's maximum delay that can be manipulated independent of other stages to obtain optimal timing.

BPAU completes worst-case operation (computation) in 5.1 ns. This time includes the input latch delays, clock skew, and hold and setup times for this stage. To accommodate the latest data wave arriving at this stage, additional time during which this data can still be passed to the next stage is allocated. This time appears in the figure as d_{pass} with a value of 0.9 ns. For data arriving late at the BPAU, at least 2 ns is the time during which this data can still be processed within this stage. The hold time for the selection function is very short, almost equal to the minimum delay of this stage. This short hold time is a direct result of the selection function design, which ensures propagation of the priority status to entries below the current one very fast by means of the priority lookahead. The port assignment has a minimum delay of 0.4 ns and requires 3 ns of hold time. It also accommodates the latest "wave" for the current data. Reduction of the gap between $d_{\max}$ and $d_{\min}$ at each stage is presented here graphically with the delays displayed to show how the clock period is made shorter in this design.

VI. Concluding Remarks

An extremely fast router capable of accommodating a number of routing algorithms and networks has been presented in this paper. The router has three stages namely condition match, selection function, and port assignment resulting in a logic depth that grows from the input latch (input register) of the condition match unit to the output latch (output register) of the port assignment unit. Data experience delays as they propagate through the stages and these delays increase with increasing logic depth. The hybrid wave-pipelining approach narrows the gap between $D_{\min}$ and $D_{\max}$ resulting in a reduction of the clock-cycle time. This study is the first of its kind; it examines the delays within each module of the design and minimizes them individually. In each module, data-delay paths are tightly controlled in order to optimize the proper propagation of signals. Delay balancing and stage-input synchronization are achieved by using internal control signals. These signals are generated to ensure that minimization of the delay difference ($d_{\min} - d_{\max}$) within each stage is achieved and passing of data between stages occurs in a proper fashion. Circuitry that generates these signals occupies less than 3% of the router area. Issues pertaining to signal generation, uneven delays, and timing are some of the important issues in wave pipelining and they have been addressed in this study to synchronize the pipeline. The BPAR presented in this paper has been designed, implemented and tested using a 0.5-μm CMOS technology.

Acknowledgment

The authors would like to thank the anonymous reviewers of this paper for their constructive comments and suggestions.

References

[1] D. Clark and J. Pasquale, "Strategic directions in networks and telecommunications," *ACM Comput. Surveys*, vol. 28, no. 4, pp. 679–690, 1996.
[2] T. Leighton, "Average case analysis of greedy routing algorithms on arrays," in *Proc. 2nd Annu. ACM Symposium on Parallel Algorithms and Architectures*, Crete, Greece, 1990, pp. 2–10.
[3] J. G. Delgado-Frias, J. Nyathi, and D. H. Summerville, "A programmable dynamic interconnection network router with hidden refresh," *IEEE Trans. Circuits Syst. I*, vol. 45, pp. 1182–1190, Nov. 1998.
[4] D. H. Summerville, J. G. Delgado-Frias, and S. Vassiliadis, "A flexible bit-associative router for interconnection networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, pp. 477–485, May 1996.
[5] J. G. Delgado-Frias and J. Nyathi, "A high-performance encoder with priority lookahead," *IEEE Trans. Circuits Syst. I*, vol. 47, Sept. 2000.
[6] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave pipelining: A tutorial and research survey," *IEEE Trans. VLSI Syst.*, vol. 6, pp. 464–474, Sept. 1998.
[7] C. T. Gray, W. Liu, and R. K. Cavin, III, "Timing constraints for wave pipelined systems," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 987–1004, Aug. 1994.
[8] W. K. C. Lam, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Valid clock frequencies and their computation in wave pipelined circuits," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 791–807, July 1996.
[9] C. T. Gray, W. Liu, and R. K. Cavin, III, *Wave Pipelining: Theory and CMOS Implementation.* Norwell, MA: Kluwer, 1994.

Jabulani Nyathi (M'02) received the B.S. degree from Morgan State University, Baltimore, MD, in 1994, and the M.S. and Ph.D degrees from the State University of New York (SUNY), Binghamton, in 1996 and 2000, respectively, all in electrical engineering.

He is currently an Assistant Professor at the School of Electrical Engineering and Computer Science, Washington State University, Pullman. He has held academic positions at SUNY, Binghamton (Adjunct Lecturer and Visiting Assistant Professor 1998–2001). His research interests include VLSI design, interconnection networks, embedded systems, and computer architecture.

Dr. Nyathi is a Member of the Tau Beta Pi.

José G. Delgado-Frias (S'81–M'86–SM'90) received the B.S. degree from the National Autonomous University of Mexico, Mexico city, Mexico, the M.S. degree from the National Institute for Astrophysics, Optics and Electronics, Puebla, Mexico, and the Ph.D. degree from Texas A&M University, College Station, TX, all in electrical engineering.

He is a Professor with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, where he holds the Centennial Boeing Chair in Computer Engineering. Prior to this appointment, he was a Faculty Member with the Electrical and Computer Engineering Department, University of Virginia, Charlottesville, and the Electrical and Computer Engineering Department, State University of New York (SUNY), Binghamton. He was a Post-Doctoral Research Fellow with the Engineering Science Department, University of Oxford, Oxford, U.K. His research interest includes VLSI design, computer architecture/organization, network routers, and optimization using genetic algorithm. He has co-authored over 100 technical papers and co-edited three books. He has been granted over 25 patents.

In 1994, Dr. Delgado-Frias received the SUNY System Chancellor's Award for Excellence in Teaching. He is a Member of the Association for Computer Machinery, American Society for Engineering Education, and Sigma Xi.