# A Study of Pipeline Architectures for High-Speed Synchronous DRAM's

Hoi-Jun Yoo

***Abstract*— The performances of SDRAM's with different pipeline architectures are examined analytically on the basis of the time required to refill the on-chip cache of a Pentium CPU. The analysis shows that the cycle time of the conventional pipeline structures cannot be reduced because of its difficulty in distributing the access time evenly to each pipeline stage of the column address access path. On the contrary, the wave pipeline architecture can make the access path evenly divided and can increase the number of pipeline stages to achieve the shortest cache refill time. But the wave congestion at the output terminal of the wave pipeline path caused by access time fluctuation narrows the valid time window. The parallel registered wave pipeline architecture can remove the effect of access time fluctuation so that the cycle time is defined only by the data pulse width. If the data pulse width $t_w < 2$ ns, even 500-MHz clock frequency can be obtained.**

***Index Terms*—DRAM chips, pipelines.**

## I. Introduction

THE synchronous DRAM (SDRAM) has been widely accepted as a DRAM which enhances speed without introducing revolutionary changes into the conventional DRAM architecture [1]–[8]. Owing to the synchronization of internal timing signals with the external clock which is shared with the CPU, it is easy to incorporate many high-speed functions into the SDRAM. Contrary to the asynchronous DRAM, in the SDRAM the consideration about set-up and hold times is relatively simple because the internal circuits have enough timing margin through the latches inserted for the synchronization. The burst read operation is a typical high-speed operation in the SDRAM. In this mode, once a word line is selected by a row address and a column address is accepted, the consecutive data on the same word line can be read by the internal column address counter synchronized with the external clock.

Different pipeline architectures have been widely employed in the SDRAM for high-speed operation [1], [3]–[8]. However, little discussion on their performances or comparison with each other has been reported. In this paper, the cache refill time of the different SDRAM architectures is examined analytically as the measure of the efficiency of the SDRAM's. And the pipeline architectures for high-speed SDRAM's are analyzed by the formula developed in the present study, and the design guidelines for fast pipeline architecture are discussed.

## II. On-Chip Cache Refill Time

For optimization of the program execution time of a computer system, the system clock cycle time has to be chosen to meet the needs of not only CPU but also cache and memory. The program execution time $t_{\text{exec}}$ is given as

$$t_{\text{exec}} = N_{\text{read}}\tau + N_{\text{read}}Mt_{\text{total}} + N_{\text{store}}n_{\text{write}}\tau \qquad (1)$$

where $N_{\text{read}}$, $N_{\text{store}}$, $M$, $n_{\text{write}}$, $\tau$, and $t_{\text{total}}$ are number of read references reaching a cache, number of stores in the program, cache miss rate, average number of cycles to complete a write operation, cycle time, and cache refill time, respectively [9]. Among them, $t_{\text{total}}$ or the total time required to replace the cache with new data from the SDRAM can be used to evaluate the performance of the SDRAM in the computer system without L2 cache. The relationship between system clock cycle time $\tau$ and the worst case access time $t_{\text{acc}}$ of the SDRAM determines the $t_{\text{total}}$. Here, only the internal sense amplifier cache hit is considered for simplicity. That is, the delay time from row activation to read-command input is excluded, and only the burst read operation is considered. This can be justified by the fact that most DRAM architectures have almost the same row path delay. If $t_{\text{acc}}$ is the memory access time under the worst conditions, such as high temperature, low $V_{\text{CC}}$, and large process fluctuation, the smallest integer $m$ which satisfies

$$m\tau \geq t_{\text{acc}} \qquad (2)$$

can be found, where $m$ is the CAS latency of the SDRAM. If $n$ clock cycles are necessary for the data refill and $\tau = \alpha t_{\text{acc}} (0 < \alpha < 1)$, the total time for the data refill is given as

$$t_{\text{total}} = m\tau + (n-1)\tau = (m+n-1)(\alpha t_{\text{acc}}). \qquad (3)$$

In case of the SDRAM, $n$ is equivalent to the burst length. From (2), it is clear that the shortest cache refill time is obtained when $\alpha = 1/m$, or when the number of the pipeline stage is equal to $m$.

Pentium chip incorporates Harvard organization, such as an 8-Kbyte code cache and an 8-Kbyte data on-chip cache [10]. The line size is 32 bytes or 256 b and the data bus width is 64 b. If cache miss occurs, four cycle times are necessary to replace the data in the on-chip cache, that is $n = 4$. When the secondary cache is omitted and only SDRAM main memory is used, its speedup factor referenced to the conventional asynchronous DRAM $(m = 1)$ and the total cache refill time can be written as [11]

$$\text{Speedup} = \frac{4m}{m+3} \qquad (4)$$

$$t_{\text{total}} = \left(1 + \frac{3}{m}\right) t_{\text{acc}}. \qquad (5)$$

From the analysis, it is found that with a fixed value of $t_{\text{acc}}$, the bigger $m$ is, the larger the speedup or the shorter the total refill time can be. However, as $m$ increases, the number of

Manuscript received September 27, 1996; revised March 11, 1997.

The author is with the Department of Electronic Engineering, Kangwon National University, Kangwon-Do, 200-701, Korea.

Publisher Item Identifier S 0018-9200(97)06307-5.

latches inserted into the path to synchronize their operations increases, causing more delays to the access path. This imposes a limitation on the maximum value of $m$ and it is optimized as

$$m_0 = \sqrt{\frac{t_{\rm acc} A_{\rm tot}}{t_l A_l}} \tag{6}$$

where $t_l$, $A_{\rm tot}$, and $A_l$ are the delay time through a latch, the cost factor of the access path, and the cost factor of a latch, respectively [12]. The cost factor can be interpreted as the silicon area. As can be seen from (6), with the increase of the delay time of a latch $(t_l)$ and area of a latch $(A_l)$, the optimum $m$ decreases.

## III. Comparisons of Different Pipeline Architectures

Fig. 1(a) shows the three-stage pipeline architecture of the column address path [1], [5]. The first stage is the address buffer stage. The second stage is composed of a column decoder and an I/O sense amp. The third stage is the data output buffer stage. Although the three stages are intended to be divided equally with respect to the propagation delay, the second stage, an analog amplification stage, has the longest delay because its further subdivision by latches may cause the signal loss. As investigated in Section II, for this architecture the total time has the smallest value when the cycle time $\tau = t_{\rm acc}/3$. However, the delay time of the second stage is larger than $t_{\rm acc}/3$, or $\tau > t_{\rm acc}/3$, which means that the three-stage pipeline architecture is not an optimum one.

In order to overcome the delay inconsistency, the second stage is split further and an additional latch is inserted into the read data (RD) line as shown in Fig. 1(b) [3]. This results in the reduction of the delay time in the second stage with a loss of signal's S/N ratio and increase of $m$, the CAS latency, from three to four. If this approach is to be more effective than the three-stage pipeline, $t_{\rm total}$ of the four-stage pipeline, $7\alpha_2 t_{\rm acc}$, should be less than that of three-stage pipeline, $6\alpha_1 t_{\rm acc}$, where $\alpha_1$ and $\alpha_2$ are the proportional coefficients of the cycle time $\tau$ to the access time $t_{\rm acc}$ in three-stage and four-stage pipelines, respectively. If $\alpha_1$ is assumed to be 1/3 and both architectures have the same $t_{\rm acc}$, the cycle time of the four-stage pipeline should be less than $(2/7)t_{\rm acc}$, or the propagation delay time of the each stage in the four-stage pipeline should be less than $(2/7)t_{\rm acc}$. This requirement is too stringent to be fulfilled due to S/N ratio degradation and the additional latch leads to the increase of the data access time to make this approach ineffective.

Another approach is to add a parallel path to the longest delay stage of the pipeline as shown in Fig. 1(c) [4]. This is known as the prefetch architecture. Once a read operation starts, a column select line and its consecutive one are activated simultaneously. From the second data, the cycle time of the second stage becomes only half of that in the three-stage pipeline architecture with no change in the number of the pipeline stages. In this architecture, the cycle time can be chosen to meet the condition for the shortest total time, $\tau = t_{\rm acc}/3$. However, Y-addresses are not free to be input randomly.

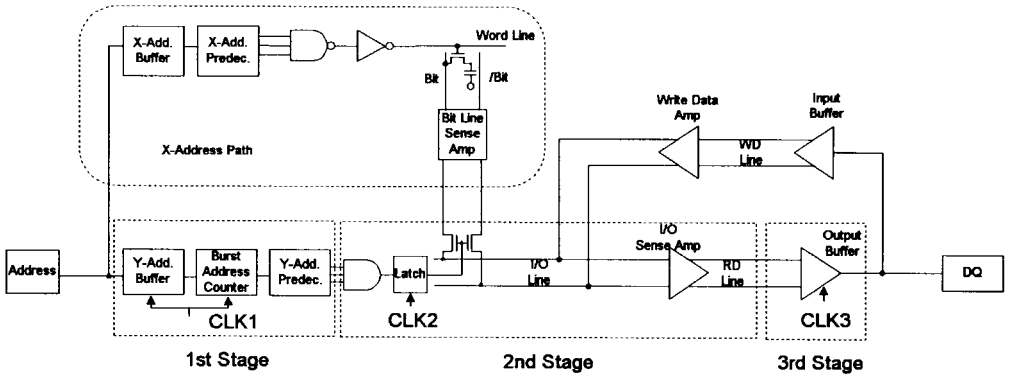
The approaches mentioned above have one common basic idea; in order to decrease the clock cycle time, they insert more latches into the column address access path for synchronization of signal flow. On the contrary, no latches are inserted in the wave pipeline architecture as shown in Fig. 2(a). Since no clocked storage elements are present in the data path, overheads such as clocking overhead $t_l$ and partitioning overhead $A_l$ in (6) do not exist. As a result, the wave pipeline can increase the clock rate by increasing $m$ while the access time remains practically unchanged [13].

In a wave pipeline architecture shown in Fig. 2(a), a column address is sampled by a short pulse generated from the clock buffer circuit. The pulsed signal that comes from the Y-address decoder selects the column switch which then makes the bit line sense amp launch a data pulse into the I/O line, without waiting for the previous pulse to be latched by the output buffer register. The number of pipeline stages is determined by the number of waves transferring the access path at one time. If the worst case delay time from the Y-address input to the data output is $t_{\rm acc}$ and the number of pipeline stages is $m$, the data pulse width $t_w$ should satisfy the condition of $t_w < t_{\rm acc}/m$. Because of the variations of the rising/falling times and the effects of capacitive loads at each node in the wave pipeline path, the pulse width of the wave gets broader with the propagation through the path as depicted in Fig. 3(a). Although the relationship $t_w < t_{\rm acc}/m$ is satisfied at the address buffer, it is not the case at the end of the pipeline stage. Unless the pulse width is less than $t_{\rm acc}/m$, the waves can overlap with each other along the pipeline path. In order to keep the pulse width narrow, the wave pipeline requires pulsed data and pulsed control signals. The post charge logic or self-resetting logic is a good candidate for the generation of the pulsed signals [6], [14]. In an example given in [6], the pulse width is kept as four-inverter delay, which is less than 2 ns.

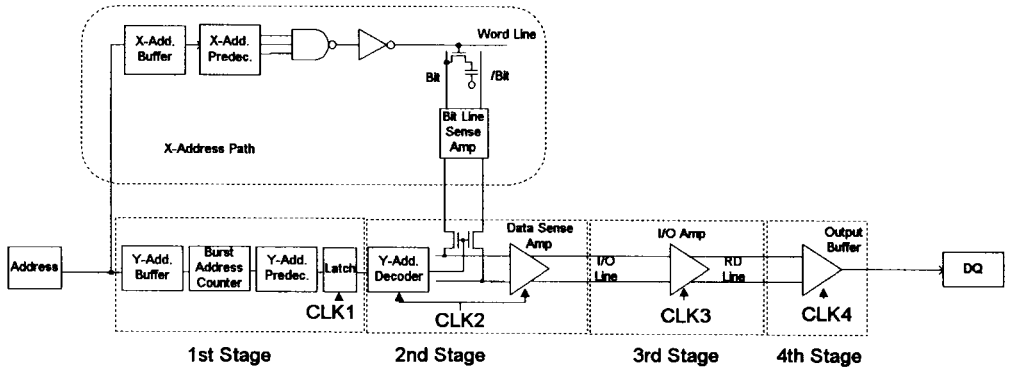
## IV. Parallel Registered Wave Pipeline Architecture

For the high-speed reliable operation, the wave pipeline architecture should remove not only the overlap of waves in a path due to the pulse width broadening but also the collision of waves from different paths at the end terminal of the pipeline path. When cell 1 and cell 2 of Fig. 3(a) are the farthest and the nearest cells from the data output buffer, respectively, and if data from cell 2 follows the data from the cell 1 which needs $t_{\rm sink}$ to leave the pipeline path at the end of the path, the second data must be launched into the pipeline path after $t_1 - t_2 + t_{\rm sink}$ from the first wave launching, where $t_1$ and $t_2$ are the delay of cell 1 and cell 2, respectively. Unless this condition is satisfied, the data waves collide at the output buffer.

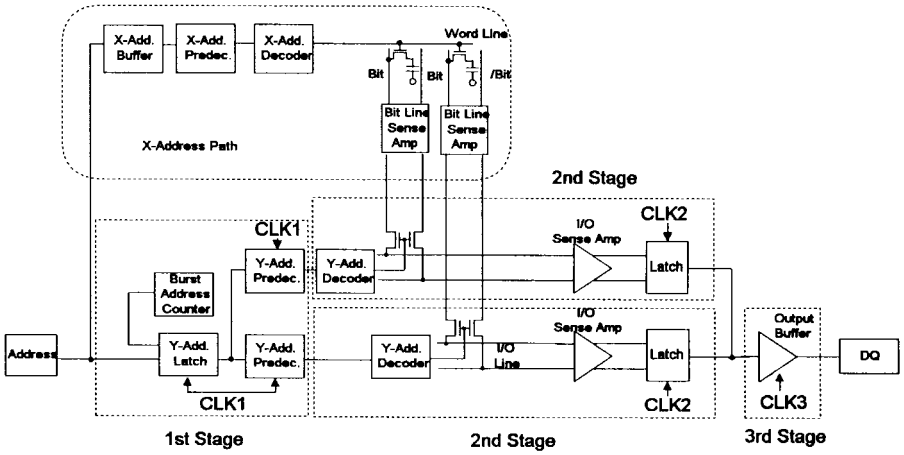
Fig. 3(b) shows variation of the $t_{\rm valid}$ with the propagation of the data pulse through the column address access path when the external clock cycle time is $\tau$. In addition to the access time difference caused by the path length difference as shown in Fig. 3(a), there exists fluctuation in the access time due to the internal noise and variations in temperature, process, and supply voltage in the SDRAM chip. If $t_1$ under the worst conditions and $t_2$ under best conditions are $t_{\rm acc}$ and $t'_{\rm acc}$, respectively, the minimum cycle time for the wave pipeline




(a)




(b)




(c)

Fig. 1. Schematic diagrams of pipeline architectures of SDRAM: (a) three-stage pipeline, (b) four-stage pipeline, and (c) prefetch pipeline architecture.

architecture of Fig. 3(b) is restricted to be

$$\tau = t_{\text{acc}} - t'_{\text{acc}} + t_{\text{valid}} \tag{7}$$

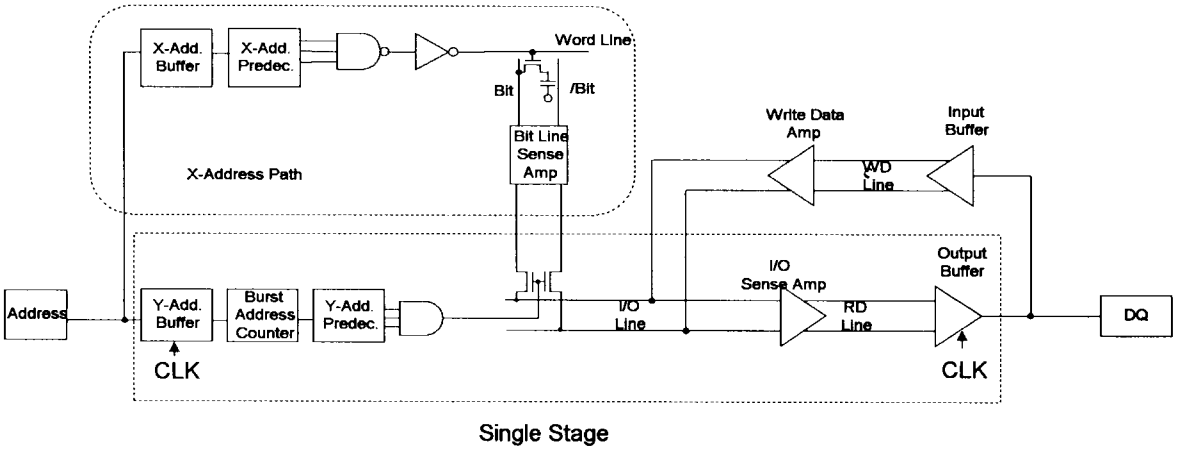
where $t_{\text{valid}} (= t_{\text{sink}} + \text{margin})$ is the time window during which the external device can capture the data [15].

If the wave pipeline architecture is to be more effective than the conventional pipeline architecture, $\tau$ obtained from (7) must be less than $t_{\text{acc}}/m$ or

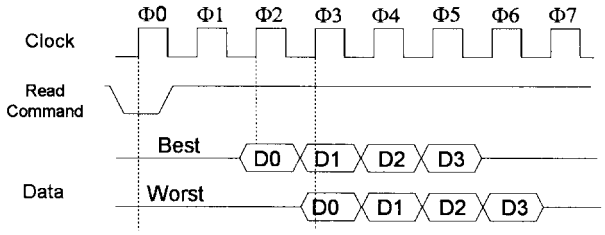
$$\left(1 - \frac{1}{m}\right) t_{\text{acc}} < t'_{\text{acc}} - t_{\text{valid}}. \tag{8}$$

In the simple CMOS gate, $t'_{\text{acc}}$ is near to $t_{\text{acc}}/2$. As $m$ and $t_{\text{valid}}$ increase, (8) is more difficult to be satisfied in the SDRAM where the difference between $t_{\text{acc}}$ and $t'_{\text{acc}}$ is large.

The valid time window can be controlled by attaching a parallel register stack to the end of the wave pipeline path as shown in Fig. 4(a). In this architecture, with $r$ registers stacked parallel at the end of the pipeline, a control circuit directs the data waves from the I/O sense amp to registers one by one, for example, the first data from the I/O sense amp to the first register and the second data to the second register, and so on. Each register holds the data for $r - 1$ cycle time or till the

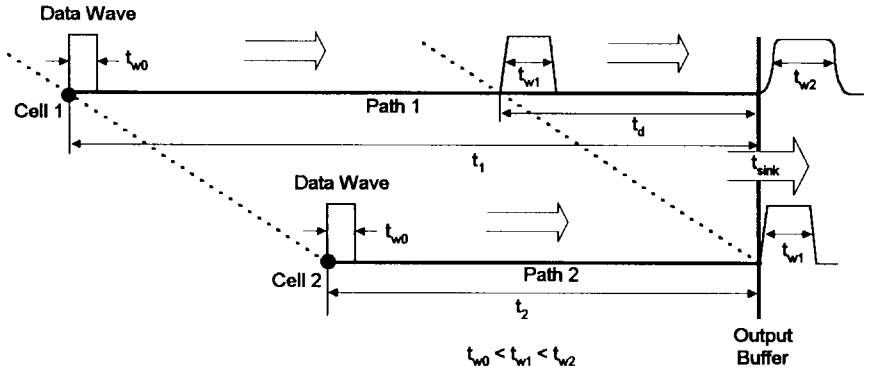



(a)

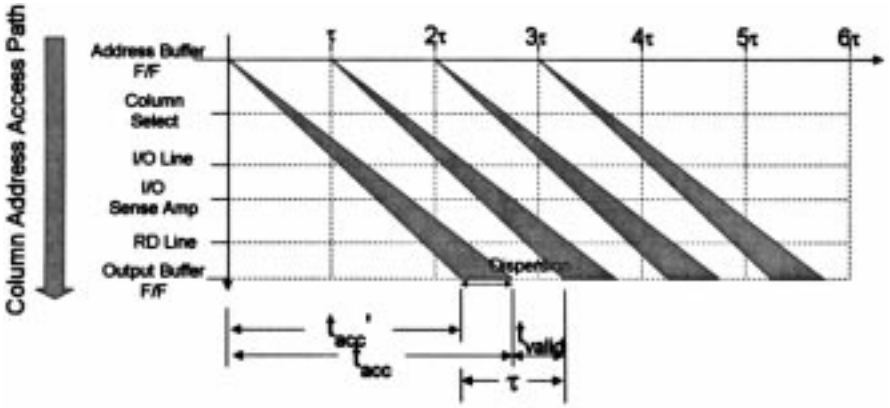



(b)

Fig. 2. (a) A schematic diagram of a wave pipeline architecture and (b) its timing diagram.

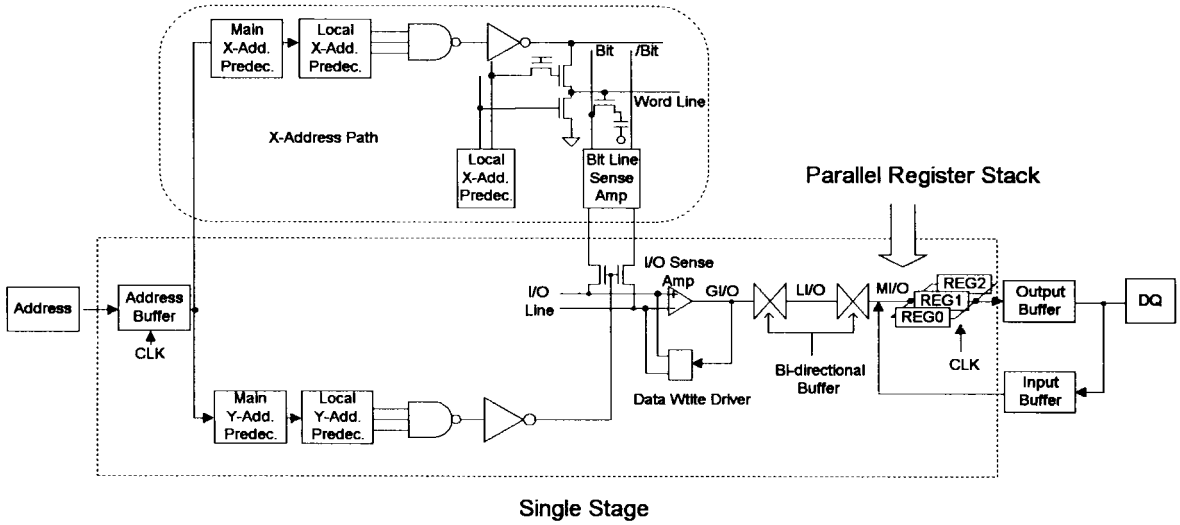



(a)

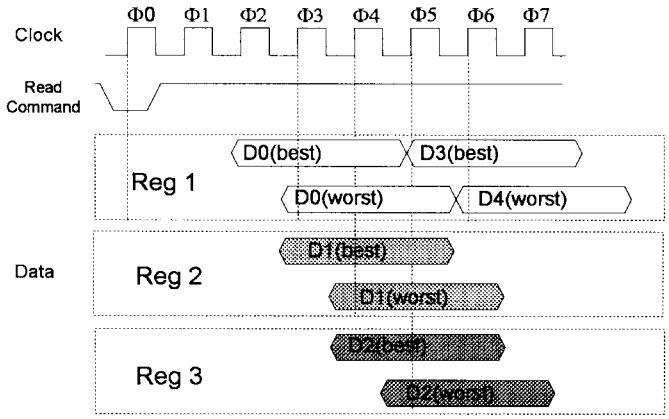



(b)

Fig. 3. (a) A schematic diagram of the wave propagation through the pipeline path and (b) a logic depth timing diagram.




(a)




(b)

Fig. 4. (a) A parallel registered wave pipeline architecture with three parallel registers and (b) its timing diagram.

next data replace the previous ones. Therefore, the valid time increases up to $t_{\text{valid}} = r\tau - (t_{\text{acc}} - t'_{\text{acc}})$. The minimum cycle time is given as

$$\tau = \frac{1}{r}\left(t_{\text{acc}} - t'_{\text{acc}} + t_{\text{valid}}\right). \tag{9}$$

By a proper choice of the value of $r$, $\tau$ can be reduced to be less than that of the conventional wave pipeline. Especially, when $r$ is chosen to be $m$, the number of the pipeline stages, and $\tau = t_{\text{acc}}/m$, the $t_{\text{valid}}$ is equal to $t'_{\text{acc}}$. This means that if $r$ is larger than $m$, there is no wave overlap problem at the output stage caused by the path delay variations. Therefore, the cycle time can be optimized only by narrowing the pulse width by using the self-resetting logic even in the presence of large fluctuations in access time.

Fig. 4(b) shows the timing diagrams of the parallel registered pipeline architectures. The data outputs under the best and worst conditions are shown. In the conventional wave pipeline, the first data under the worst conditions outputs at the same time as the second data does under the best conditions as shown in Fig. 2(b). To avoid the data collision, the on-chip cache has to change the data acquisition timing according to the environmental and process variations, which is not realistic. The parallel registered wave pipeline architecture resolves the acquisition timing variation by holding the data for a longer period of time than one cycle. In the example of Fig. 4, a three-register stack is attached at the output terminal of the three-stage wave pipeline path. The first data is latched by the register 0 and second data by register 1, third data by register 2, and fourth data by register 0 again. The latching timings under the best and worst conditions differ from each other as for the conventional wave pipeline architecture. However, the data D0 can be acquired at $\Phi 3$ under both the best and worst conditions since the register holds the data for three cycle times. D1, D2, and D3 can be acquired at $\Phi 4$, $\Phi 5$, and $\Phi 6$, respectively, without any valid time window restriction. Another advantage of this architecture is that latency programming and control can be easily implemented as explained in [6].

## V. Discussions and Conclusions

Table I summarizes the characteristics of the different pipeline architectures. Although the number of stages may increase more than four by narrowing the pulse width in the wave pipeline architecture, their path delay time $t_{\text{acc}}$ is even

TABLE I
COMPARISONS OF THE DIFFERENT PIPELINE ARCHITECTURES

| Pipelines | CAS Latency m | Path Delay $t_{acc}$ | Cycle Time $\tau$ | $t_{total}$ | Speedup | Area |
|---|---|---|---|---|---|---|
| 3-stage | 3 | $t_{acc}+3t_l$ | $\geq t_{acc}/3$ | $\geq 2t_{acc}$ | 2 | 1 |
| 4-stage | 4 | $t_{acc}+4t_l$ | $\geq t_{acc}/4$ | $\geq 7t_{acc}/4$ | 2.3 | 1.09 |
| Prefetch | 3 ~ 4 | $t_{acc}+3t_l$ | $\geq t_{acc}/3$ | $7t_{acc}/3$ | 2.3 | 1.07 |
| Wave | $t_{acc}/\tau$ | $t_{acc}$ | $\geq t_{acc}-t_{acc}'+t_{valid}$ | $\geq t_{acc}+3\tau$ | $\frac{4}{1+3\tau/t_{acc}}$ | 0.97 |
| Parallel Registered Wave | $t_{acc}/t_w$ | $t_{acc}$ | $\geq t_w$ | $\geq t_{acc}+3t_w$ | $\frac{4}{1+3t_w/t_{acc}}$ | 0.98 |

$t_w$: Pulse Width

$t_l$: Delay time through latch

shorter than those of the conventional pipeline architectures because they do not experience the latch delay $t_l$, and in the conventional pipeline architectures, the cycle time $\tau$ is limited by the number of latches in the pipeline path. In the wave pipeline architecture the limitation on the cycle time results from the intrinsic pulse width broadening and access time fluctuations caused by the environmental variations. On the contrary, the cycle time of the parallel registered wave pipeline architecture with self-resetting CMOS logic is restricted by only the data pulse width $t_w$. If $t_w$ is chosen to be less than 2 ns, the system clock frequency of 500 MHz can be obtained.

The lower bounds of $t_{\rm total}$ of the three-stage and four-stage pipeline architectures are $2t_{\rm acc}$ and $7t_{\rm acc}/4$, respectively. The prefetch architecture has $t_{\rm total}$ of $7t_{\rm acc}/3$. The wave pipeline architecture may have a shorter $t_{\rm total}$ if $\tau < t_{\rm acc}/4$ is satisfied, which is, however, unpredictable. In the parallel registered wave pipeline architecture, $t_{\rm total}$ can be made to converge to $t_{\rm acc}$ by choosing $t_w$ to be far smaller than $t_{\rm acc}$. Therefore, parallel registered wave pipeline architecture can have the smallest $\tau$ and $t_{\rm total}$ among the considered pipeline architectures.

The speedup of SDRAM's has a value of two to four. The three-stage pipeline architecture has a smallest speedup, two, and the parallel registered wave pipeline architecture has the maximum value, nearly four, by choosing $t_w \ll t_{\rm acc}$. If the chip area of the three-stage pipeline architecture is chosen as a unit reference, the four-stage pipeline architecture shows 9% increase and prefetch architecture has a 7% increase because they have complicated clock control circuits and clock distribution lines in the DRAM core area. On the contrary, the wave pipeline architectures do not have layout burden for clock distribution circuits and latches. Although the parallel registered wave pipeline architecture has register stacks at each data output buffer, its layout burden is relatively small because the periphery area near the data output buffer has enough room for extra layout.

In conclusion, this paper shows that the parallel registered wave pipeline architecture has the best performance among different pipeline structures even in the presence of the access time fluctuation. In addition, the parallel registered wave pipeline architecture is shown to be the best candidate for the high-speed SDRAM where analog operation such as sensing and amplification of cell signal and digital data transfer along I/O lines are mixed together. If the pulse width is chosen to be less than 2 ns, even 500-MHz system clock frequency can be obtained.

## REFERENCES

[1] Y. Takai, M. Nagase, M. Kitamura, Y. Koshikawa, N. Yoshida, Y. Kobayashi, T. Obara, Y. Fukuzo, and H. Watanabe, "250 Mbyte/sec synchronous DRAM using a 3-stage pipelined architecture," in *Symp. VLSI Circuit Dig. Tech. Papers*, June 1993, pp. 59–60.

[2] Y. Choi, M. Kim, T. Kim, S. Lee, H. Lee, C. Park, S. Lee, C. Kim, B. Lee, S. Cho, E. Haq, J. Karp, and D. Chin, "16 Mbit synchronous DRAM with 125 Mbyte/sec data rate," in *Symp. VLSI Circuit Dig. Tech. Papers*, June 1993, pp. 65–66.

[3] A. Fujiwara, H. Kikukawa, K. Matsuyama, M. Agata, S. Iwanari, M. Fukumoto, T. Yamada, S. Okada, and T. Fujita, "A 200 MHz 16 Mbit synchronous DRAM with block access mode," in *Symp. VLSI Circuit Dig. Tech. Papers*, June 1994, pp. 79–80.

[4] Y. Kodama, M. Yanagisawa, K. Shigenobu, T. Suzuki, H. Mochizuki, and T. Ema, "A 150 MHz 4 bank 64 Mbit SDRAM with address incrementing pipeline scheme," in *Symp. VLSI Circuit Dig. Tech. Papers*, June 1994, pp. 81–82.

[5] Y. Sakai, K. Oishi, M. Matsumoto, S. Wada, T. Sakashita, and M. Katayama, "A synchronous DRAM with new high-speed I/O lines method for multimedia age," in *IEICE Trans. Electron.*, vol. E78C, no. 7, pp. 782–788, July 1995.

[6] H. J. Yoo, K. W. Park, C. H. Chung, S. J. Lee, H. J. Oh, K. W. Kwon, J. S. Son, W. S. Min, and K. H. Oh, "A 150 MHz 8-banks 256 M synchronous DRAM with the wave pipelining method," in *ISSCC, Dig. Tech. Papers*, Feb. 1995, pp. 250–251.

[7] S. J. Lee, K. W. Park, C. H. Chung, J. S. Son, K. H. Park, S. H. Shin, S. T. Kim, J. D. Han, H. J. Yoo, W. S. Min, and K. H. Oh, "A low noise 32 bit-wide 256 M synchronous DRAM with column decoded I/O line," in *Symp. VLSI Circuit Dig. Tech. Papers*, June 1995, pp. 113–114.

[8] J. M. Han, J. B Lee, S. S. Yoon, S. J. Jeong, C. Park, I. J. Cho, S. H. Lee, and D. I. Seo, "Skew minimization techniques for 256 Mbit synchronous DRAM and beyond," in *Symp. VLSI Circuit Dig. Tech. Papers*, June 1996, pp. 192–193.
[9] S. A. Przybylski, *Cache and Memory Hierarchy Design.* San Francisco, CA: Morgan Kaufman, 1990.
[10] Intel Corporation, *Pentium Processor Family Developer's Manual*, 1996.
[11] J. L. Hennesy and D. A. Patterson, *Computer Architecture, A Quantitative Approach*. San Francisco, CA: Morgan Kaufman, 1996.
[12] M. R. Zargham, *Computer Architecture, Single and Parallel Systems*. Upper Saddle River, NJ: Prentice Hall, 1996.
[13] D. C. Wong, G. D. Micheli, M. J. Flynn, and R. E. Huston, "A bipolar population counter using wave pipelining to achieve 2.5 $\times$ normal clock frequency," in *ISSCC, Dig. Tech. Papers*, Feb. 1992, pp. 56–57.
[14] T. Williams, "Self-timed clocked logic techniques," presented at Proc. Symp. VLSI Circuit Workshop, June 1996.
[15] C. T. Gray, W. Liu, and R. K. Cavin III, *Wave Pipelining: Theory and CMOS Implementation*. Norwell, MA: Kluwer, 1994.