NoBL: The Fast SRAM Architecture

Introduction

Processors in high performance PCs, workstations, communication equipment, and network applications demand high speed memories. The type of memory required is determined by the system architecture, the application and the processor used. System performance will suffer if the memory sub-system cannot satisfy the processor requirements. In order to improve memory performance several trade-offs can be made. This application note introduces the Cypress NoBL architecture, a revolutionary SRAM architecture designed to improve memory sub-system performance by up to 50% over existing solutions in certain types of applications.

NoBL is a family of Synchronous SRAMs derived from existing Synchronous Pipelined and Flow Through SRAMs. NoBL stands for **No B**us **L**atency, which means that the device can complete read and write operations without any of the latency associated with standard synchronous SRAMs. The NoBL family includes both Pipelined and Flow-through devices. *Table 1* shows the current and planned devices in the family.

Table 1. Devices in the NoBL Family.

Device	SIze	Description
CY7C1334	64K x 32	Pipelined NoBL device
CY7C1333	64K x 32	Flow-through NoBL device
CY7C1350	128K x 36	Pipelined NoBL device
CY7C1351	128K x 36	Flow through NoBL device
CY7C1352	256K x 18	Pipelined NoBL device
CY7C1353	256K x 18	Flow through NoBL device

What is a NoBL?

The existing synchronous pipelined burst SRAMs are optimized for PC cache applications. In these applications, data is written into the SRAM and read out numerous times prior to being replaced. Hence the accesses are dominated by reads with very few READ/WRITE transitions. Adding wait states to such transitions will not seriously degrade system performance. However, not all applications are like that of a L2 cache. Many, like networking applications, experience frequent READ/WRITE transitions. These types of applications can benefit from the elimination of these wait states. The NoBL architecture was designed to optimize memory performance where there are frequent READ/WRITE transitions. By using NoBL devices a system can reduce the number of unused (or “dead”) cycles on the bus to Zero. Applications such as ATM switches and network equipment rely frequently on consecutive READ/WRITE memory accesses. These applications will benefit from the Cypress NoBL device.

In a system with frequent READ/WRITE/READ/WRITE operation, the effective bus utilization using the NoBL device is100%, compared to a effective bus utilization of 50% for a system using **S**tandard **P**ipelined **B**urst **SRAM**s. This increase in Bus utilization will be discussed in detail in a later section.

The Cypress NoBL Device

Figure 1 shows the block diagram of the CY7C1334 which is the pipelined 64K x 32 NoBL device.

Like all pipeline devices, the inputs are sampled on the rising edge of the clock. The output registers for the data are also controlled by the rising edge of the clock. The clock signal is qualified using the $\overline{CEN}$ signal. The $\overline{CEN}$ signal can be used to add wait states to any access of the SRAM.

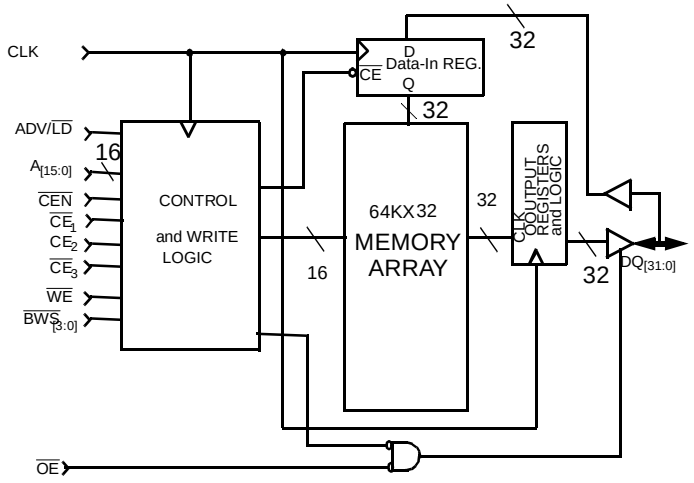

Figure 1. Block Diagram of the CY7C1334, a Pipelined NoBL Device

One of the features of the NoBL architecture is that there is always a fixed offset between address and data, regardless of whether the access is a READ, WRITE, or deselect. For pipelined devices, the data is always available two cycles after the address is clocked in, For a flow-through device, there is a single cycle between address and data.

Three synchronous chip enables ($\overline{CE}_1$, CE_2, $\overline{CE}_3$) and an asynchronous output enable ($\overline{OE}$) provide for easy bank selection and output three-state control. In order to avoid bus contention, the output drivers are synchronously three-stated during the data portion of a write sequence.

The CY7C1334 has a 2-bit on-chip burst counter, which can be used for burst access during a READ or a WRITE.

How Does NoBL Compare to Standard Synchronous SRAMs?

Standard Synchronous SRAM

Figure 2 shows the simplified timing diagram for a standard pipelined SRAM conducting a READ/WRITE/READ/WRITE sequence. On Clock #1, the address for the initial read is latched into the SRAM at the rising edge of this clock. The SRAM drives the requested data onto the bus in clock #2. During Clock cycle 2 and Clock cycle 3 the device cannot initiate a write operation because the read is in progress. In clock #3 the read is completed with the requesting device latching the data from the bus. In clock #4 a single cycle write can be initiated using $\overline{ADSC}$. Once this operation is completed another read access can begin in clock #5. In other words, a read cycle followed by a write cycle will incur 2 dead clock cycles, with clocks #2 and #3 being wasted with no data being transferred.

In a pipelined SRAM, read and write sequences are not symmetrical (Read takes 3 cycles, while Write takes 1 cycle to complete). Therefore these types of devices are more efficient in systems that have burst read or burst write accessed.

Flow Through SRAM

Figure 3 shows the simplified timing diagram for a Flow-Though SRAM conducting the same READ/WRITE/READ/WRITE sequence.

A Flow Through SRAM is also not optimized for back to back read/write operations. The Flow through SRAM also exhibits non symmetrical read and write timing sequences (Read takes two cycles to complete, while write takes one cycle to complete). So in the flow through case, there is one dead cycle for each read/write sequence.

The NoBL

Figure 4 shows the timing diagram of a Pipelined NoBL device in a Read/Write/Read/Write sequence.

In Clock #1, the address for a read access is latched into the SRAM. Because of the internal pipeline the SRAM provides data in clock cycle #3. The NoBL SRAM has advanced logic that allows all accesses to be pipelined. This means that the next write access can be started in clock cycle #2 as shown. The data for the corresponding address is provided in clock cycle #4.

As shown in *Figure 3*, NoBL accesses are completely symmetrical (three clock cycles to complete a READ or WRITE). Therefore all accesses can be fully pipelined with no cycle lost between a read and a write.

SPB SRAM Timings for a R-W-R-W Operation

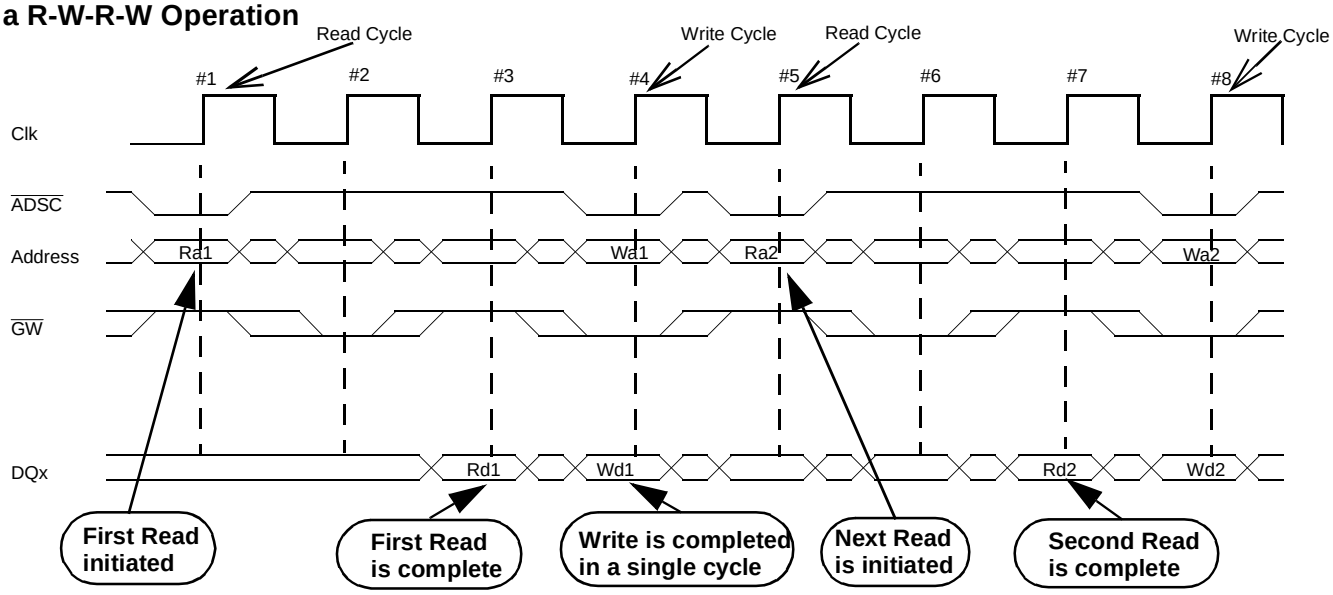

Figure 2. SPB SRAM Timing Diagram

Flow Through Timings for a R-W-R-W Operation

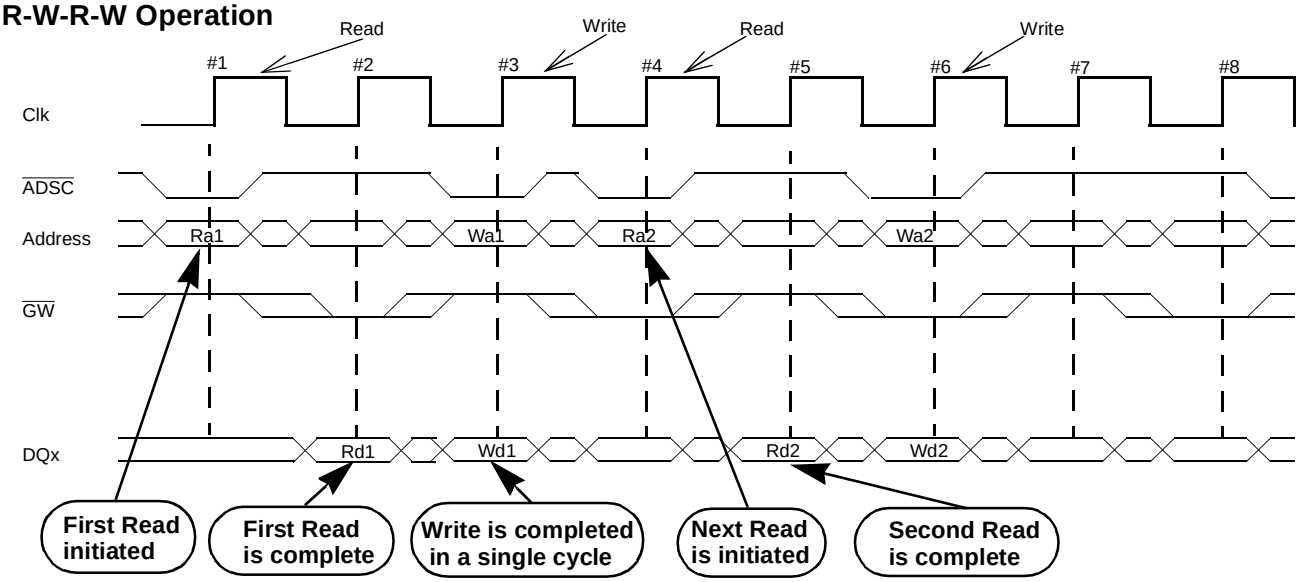

Figure 3. Flow Through SRAM Timing

NoBL Timings for a R-W-R-W Operation

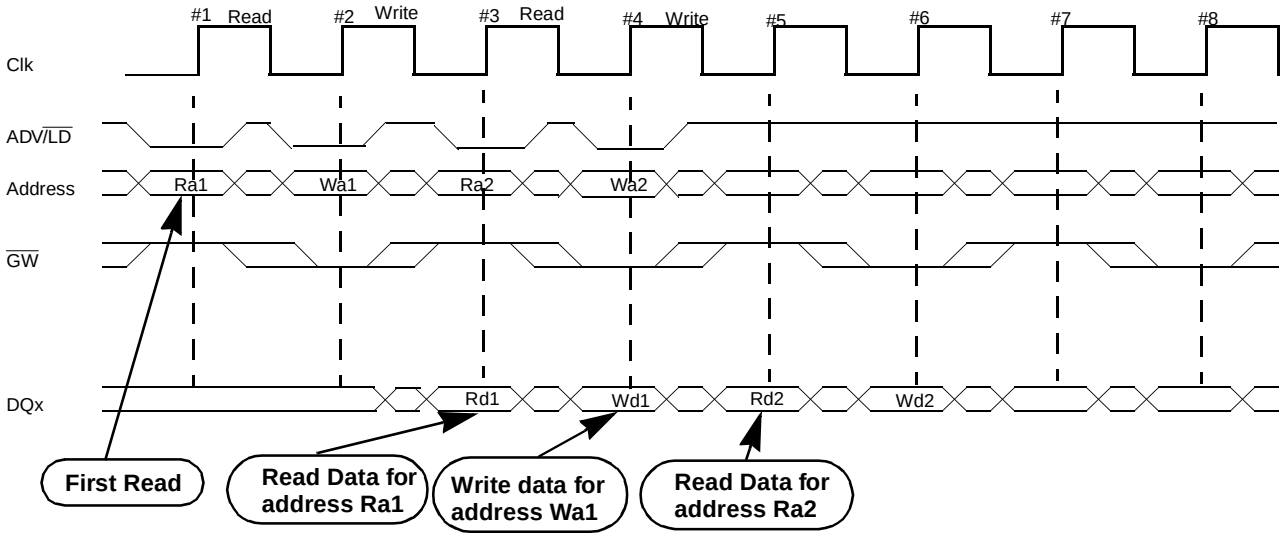

Figure 4. Pipelined NoBL SRAM Timing Diagram

NoBL Timings for a R-W-R-W Operation

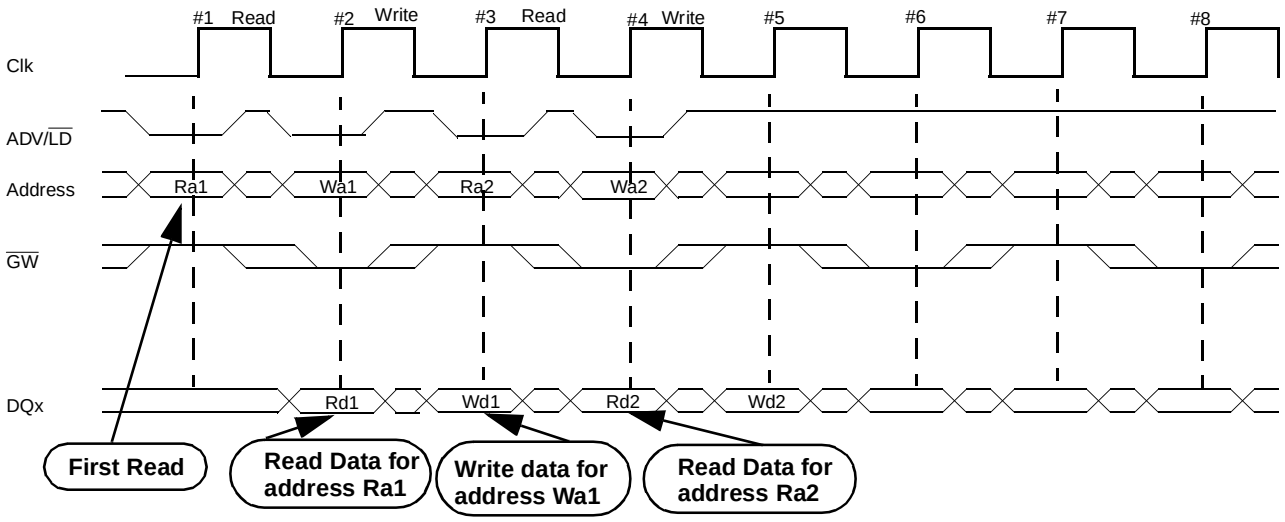

Figure 5. Flow Through NoBL Timing Diagram

Bus Efficiency

Bus efficiency is a metric used to measure the efficiency of a device transferring data over a bus. In the case of an SRAM, this figure shows the number of cycles which are wasted when transferring back to back READ/WRITE data.

Bus Efficiency = Data Xfer Cycles / Total Number of Cycles

For an operation such as a R-W-R-W, the maximum bus efficiency is achieved when data is transferred once every clock. In other words, 100% bus efficiency occurs when the data is transferred on every clock cycle regardless of the operation.

The standard pipelined SRAM takes 8 cycles to complete a R-W-R-W operation, which means it has a bus efficiency of 50%.

The standard flow through SRAM takes 6 cycles to complete a R-W-R-W operation, which means that it has a bus efficiency of 66%.

The NoBL SRAM is designed to complete a data transfer on every cycle and so it has a bus efficiency of 100%.

Table 2 compares the Bus Efficiency of different SRAM's under different conditions.

Table 2. Comparison of the Number of Cycles/Operation for Different Operation.

	Cycles for Operation		
Device	**R-W-R-W**	**R-R-R-R**	**W-W-W-W**
PBSRAM	50%	80%	100%
Pipelined NoBL SRAM	100%	100%	100%
Flow through SRAM	66%	50%	100%
Flow through NoBL	100%	100%	100%

Timing Analysis of the NoBL SRAM

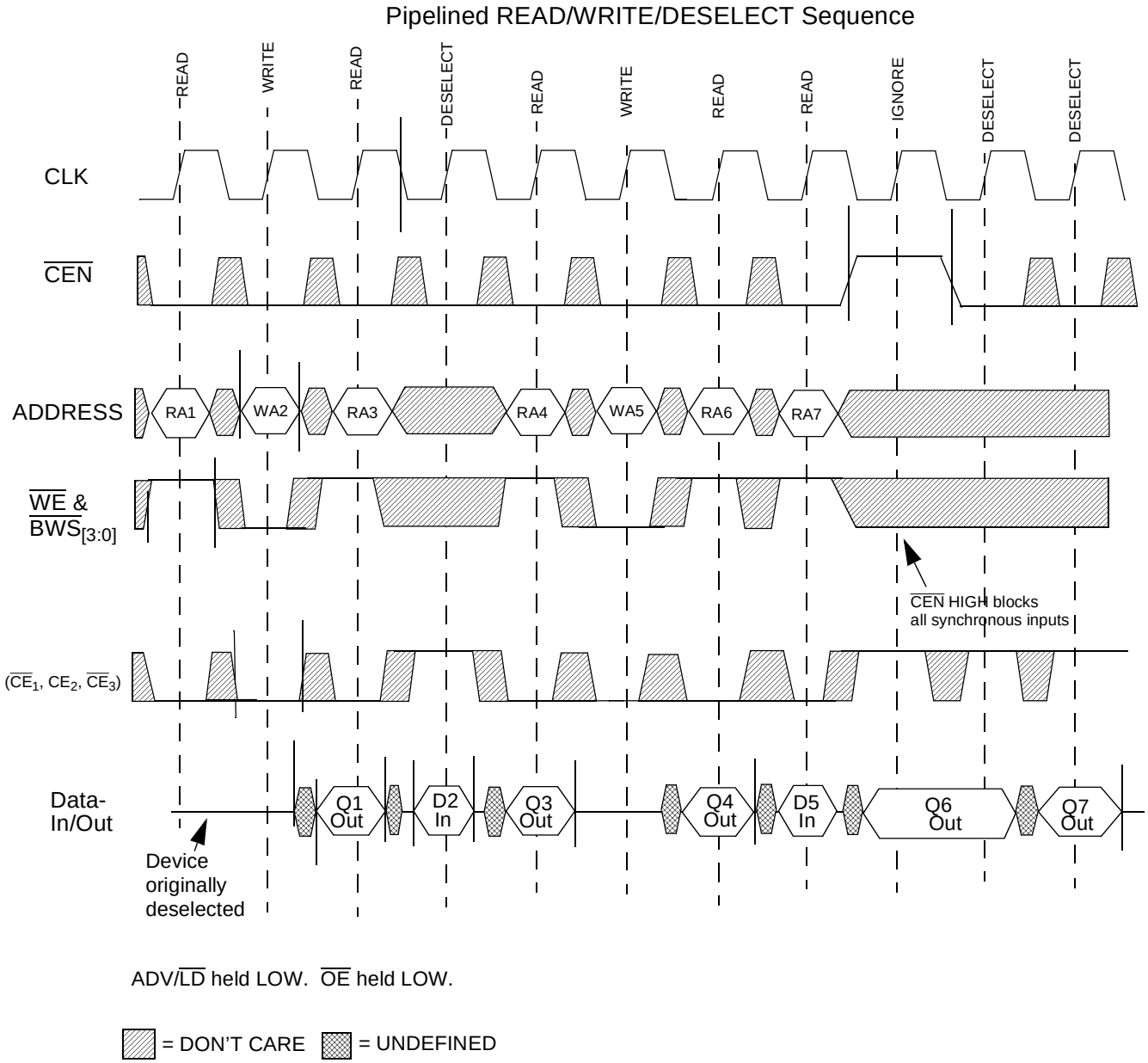

Figure 6. Detailed Timing of the Pipelined NoBL SRAM

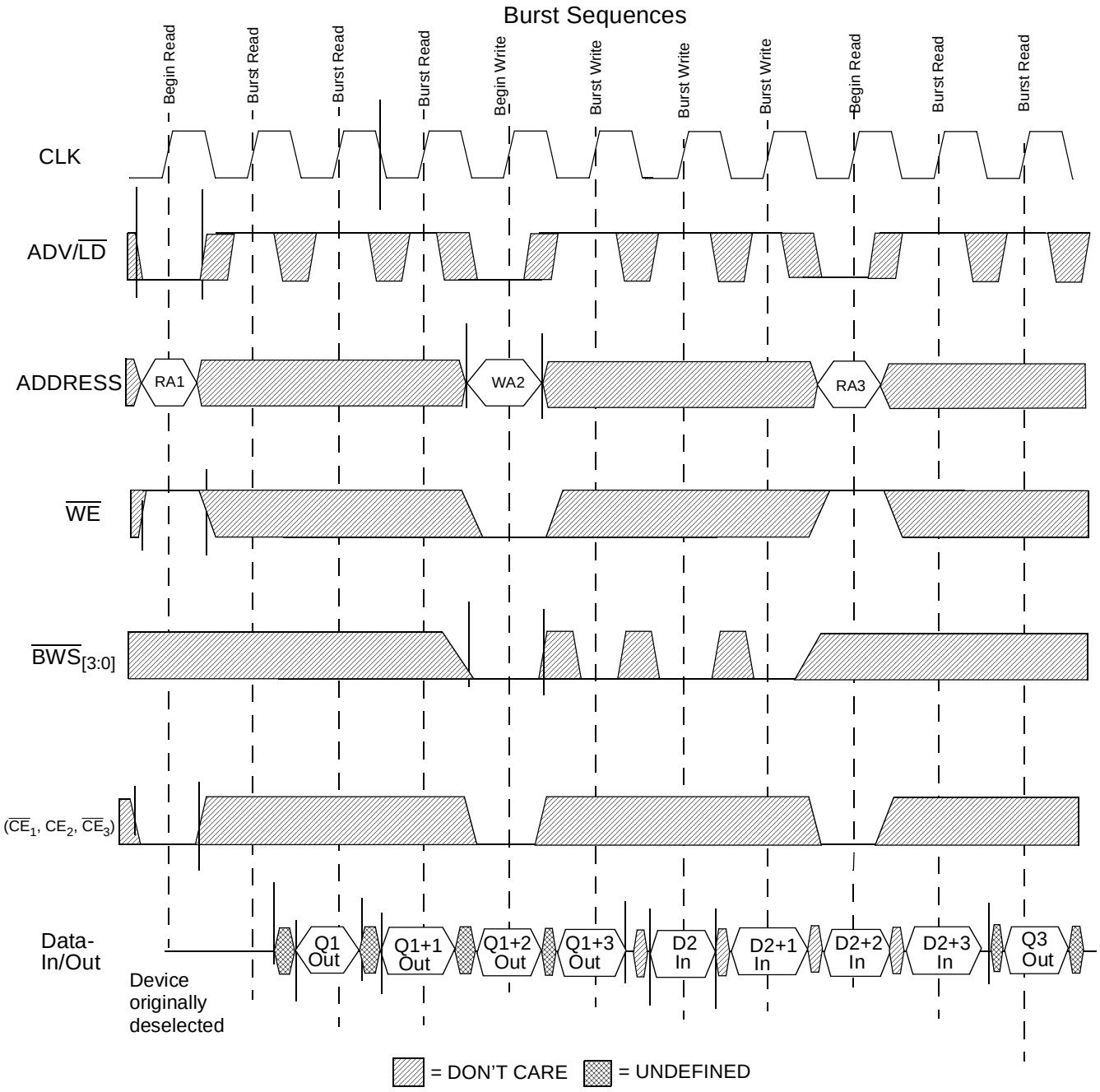

Figure 7. Burst Timing for the Pipelined NoBL Device

There are several timing parameters which have to be looked into while designing the NoBL SRAM into a system.

Figures 6 and *7* show the detailed timings of a pipelined NoBL SRAM.

New access cycles are controlled by the ADV/$\overline{LD}$ signal. Advance/Load input used to advance the on-chip address counter or to load a new address. When this input is HIGH at the rising edge of the clock (and $\overline{CEN}$ is asserted LOW) the internal burst counter is advanced. When this signal is LOW at clock rise, a new address can be loaded via the address lines into the device for an access.

A read cycle is started when the $\overline{WE}$ signal is high on the rising edge of the clock., In the subsequent cycles, if the ADV/$\overline{LD}$ is high, the device starts a burst access shown in *Figure 7*.

The sequence of the burst counter is determined by the MODE input signal. A LOW input on MODE selects a linear burst mode, a HIGH selects an interleaved burst sequence.

Both burst counters use A0 and A1 in the burst sequence, and will wrap-around when incremented sufficiently. After loading a new address to the SRAM by asserting ADV/$\overline{LD}$, $\overline{CEN}$, $\overline{CE}_1$, $\overline{CE}_3$ LOW, and CE_2 HIGH, putting ADV/$\overline{LD}$ HIGH in subsequent clock cycles will increment the internal burst counter regardless of the state of chip enables inputs or $\overline{WE}$. $\overline{WE}$ is latched at the beginning of a burst cycle. Therefore, the type of access (Read or Write) is maintained throughout the burst sequence. *Table 3* shows the Interleaved Burst Sequence, while *Table 4* shows the Linear Burst Sequence.

Table 3. Interleaved Burst Sequence.

First Address	Second Address	Third Address	Fourth Address
Ax+1, Ax	Ax+1, Ax	Ax+1, Ax	Ax+1, Ax
00	01	10	11
01	00	11	10
10	11	00	01
11	10	01	11

Table 4. Linear Burst Sequence.

First Address	Second Address	Third Address	Fourth Address
Ax+1, Ax	Ax+1, Ax	Ax+1, Ax	Ax+1, Ax
00	01	10	11
01	10	11	00
10	11	00	01
11	00	01	10

The NoBL device would start a write access if the $\overline{WE}$ is LOW on a rising edge of the clock with the device selected.

1) t_{CHZ} and t_{CLZ} issues

t_{CHZ} is the time it takes for the NoBL device to place its output drivers into a high-impedance state after the rising edge of the clock. t_{CLZ} is the time it takes for the NoBL device to start driving data onto the data bus (a low impedance state). *Table 5* shows the maximum and minimum timings of the t_{CHZ} and t_{CLZ} as specified on the CY7C1334 data sheet.

Table 5. Values of t_{CHZ} and t_{CLZ}.

Parameter	Min.	Max.
t_{CHZ}	1.5	3.5
t_{CLZ}	2.5	

This appears as if the device is specified to allow data contention between SRAMs sharing a common data bus (due to the overlap of $t_{CHZ(max)}$ and $t_{CLZ(min)}$). This is not the case. The specifications for the two parameters are guaranteed over the entire process, temperature and voltage range. $t_{CHZ(max)}$ is seen at slow corner of the process, high temperature, and low operating voltage. $t_{CLZ(min)}$ is seen at the opposite operating extreme outside of process variations (Fast process, Low temperature, High voltage).

Obviously these two extremes will not exist on the same board at the same time. The NoBL device is designed to drive the bus into High-Z before Low-Z under all operating conditions with approximately a 1ns of delta between the two timings, regardless of processing variations. Therefore, contention on the data bus will not occur between NoBL SRAMs.

2) Chip Selects on NoBL SRAM

The Chip selects on the NoBL SRAM operates in ways different from that of a Synchronous Pipelined Burst SRAM. The NoBL SRAM has three chip selects $\overline{CE}_1$, CE_2, and $\overline{CE}_3$. On a SPB SRAM the $\overline{ADSP}$ signal is ignored when the chip selects are inactive.

In the NoBL, the CE pins are sampled on the rising edge of the clock. None of the pins on the NoBL are masked by any other input. Therefore, all chip enables need to be active to select the device, and any of the three can deselect the device.

3) $\overline{OE}$ Control

The CY7C1334 is a common I/O device, Data should not be driven into the device while the outputs are active. The Output Enable ($\overline{OE}$) can be de-asserted HIGH before presenting data to the DQ_0–DQ_{31} inputs. Doing so will three-state the output drivers. However, the internal logic recognizes when a write is initiated, and synchronously disables the output drivers in order to allow the presentation of the write data. This feature greatly simplifies write sequences and, in most cases, eliminates the need to use $\overline{OE}$ during writes.

Application

As described in the earlier sections, applications using back to back READ-WRITE operations would benefit significantly from the NoBL SRAM.

The NoBL SRAMs eliminate data latency and provide maximum memory bandwidth utilization.

An ATM switch application is used as an example to show the improvement in system performance by using NoBL SRAMs.

ATM switches are applications which require high memory throughput. There are three ways of implementing an ATM switch: Shared memory, Self-routing fabric, Shared Backplane.

The Shared memory architecture is one where a large shared memory is used to buffer the incoming cells before being routed to one of the output ports. *Figure 8* shows a typical shared memory architecture of an ATM switch. The size of the shared memory depends on the number of cells which have to be buffered for each of the output ports. The data rate supported by the switch determines the width of the data bus for the shared memory and the frequency of operation of the memory.

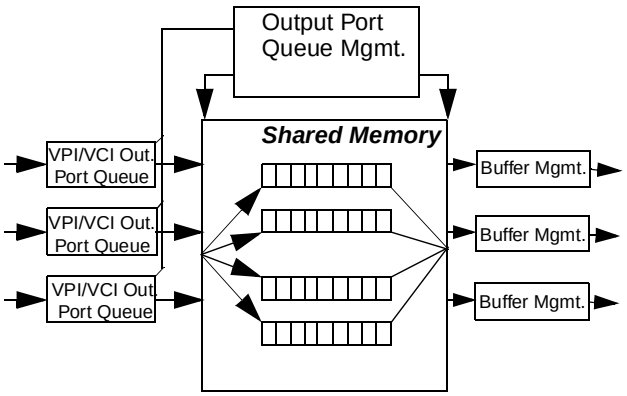

Figure 8. A Shared Memory ATM Switch

Most of the shared memory switches use SRAM's. For this discussion we use a shared memory module which can store up to 26K cells (1 ATM Cell = 53 Bytes) with a target data rate of 19.2 Gbps. Most of the ATM operations involve continuous Writes and Reads of ATM cells.

The most important consideration in using a certain type of memory in this application is its ability to provide the data rate of 19.2 Gbps. There are several ways of achieving the data rate required for such an application. One way of achieving the desired data rate without running the SRAM at very high speed data rate is to use wider data bus widths. Another way of achieving the higher data rate is to run the synchronous SRAM's at a higher clock frequency. The clock frequency required can be calculated using the formula.

Frequency = Data Rate/(Bus Efficiency * Width of the Data Bus)

In this application, let us assume that we use a bus width of 192 bits.

Synchronous BSRAM Solution

Consider the memory array of being made using 64K x 32 SPB SRAMs.

For a 26K cell module, we would need six 64K x 32 SRAMs to create a 64K x 192 memory bank. The memory bank is shown in *Figure 9*.

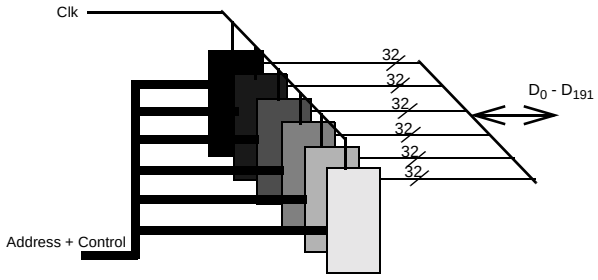

Figure 9. Memory Bank for 26K Cell Module

As discussed in previous sections, the pipelined SRAM cannot complete back to back read/write operations on consecutive cycles. The bus utilization of such a RAM comes to 50%.

To operate this block at 19.2 Gbps, the frequency of operation is:

Frequency = 19.2 Gbps/ (0.5 * 192 bits) = 200 MHz.

The frequency of operation comes to about 200 MHz. So the SRAM's have to operate at 200 MHz. in order to support this data rate.

If the same design is implemented using flow though SRAM's with 66% bus efficiency, the frequency of operation is:

Frequency = 19.2 Gbps / (0.66 * 192) = 151 MHz.

This puts a huge stress on the system design and the availability of such devices.

New Solution using NoBL devices.

One of the solutions available for such applications is the No Bus Latency (NOBL) SRAM. This devices does not need a turnaround time between a write and a read.

The pipeline version of the device is designed to have a standard offset of two cycles for a read and a write.

By using this device, the bus utilization can be improved to 100%.

To operate this block at 19.2 Gbps, the frequency of operation comes to

19.2 Gbps/ (1.0 * 192 bits)

The frequency of operation comes to about 100 MHz.

Conclusion

Therefore a new innovative architecture of SRAM has increased the memory bandwidth to meet system requirements instead of just making the same device faster and faster. By providing a faster memory, NoBL devices will enable faster and faster systems!