Integrated Device Technology, Inc.

IDT79R3051™ MAIN MEMORY AND SYSTEM I/O INTERFACING

APPLICATION NOTE AN-92

By Andrew Ng

INTRODUCTION

The IDT79R3051™ RISController™ family provides a simple, flexible external bus interface to directly support main memory and system I/O resources. The bus interface is straightforward in that it uses a single, multiplexed 32-bit address and data bus and a small number of supporting control signals. The bus interface is adaptable in that it can handle different types and speeds of memory including DRAM, SRAM, and EPROM and different kinds of I/O resources. Thus the simple, flexible R3051 bus interface allows designers to make optimal trade-offs between system speed and cost issues.

MAIN MEMORY DESIGN

The R3051 normally accesses its internal instruction and data cache memories as in Figure 1, while using external main memory as a secondary source of memory as in Figure 5. Since the R3051 contains its own internal instruction and data caches, the complexity of the cache timing and interfacing is kept on-chip, which allows the external interface to be dedicated to main memory and system I/O interfacing. The system interface is decoupled from cache memory by the use of an internal 4-deep read buffer and an internal 4-deep write buffer. The instruction and data cache allow the R3051 to access 1 instruction and 1 data word on each clock cycle. On reads, when a cache miss or an uncachable reference occurs, the R3051 begins an external read cycle which buffers 1 word on non-burst reads and 4 words at a time on burst reads from system I/O and main memory. On writes, the R3051 maintains a write-through cache update policy which simultaneously updates both the data cache and main memory. With the use of its 4-deep write buffer, the R3051 can continue to execute instructions from its instruction cache while the main memory retires up to 4 words from the write buffer.

Read and Write Cycle Protocols

The simple read interface allows a wide range of memories and I/O to be used with the R3051, from slow I/O peripherals to high speed burst accessed DRAM and SRAM. As shown in Figure 2 and 3, the read interface supports both single datum accesses and 4-word burst accesses simply by providing a Burst output signal and by providing dedicated LSB address line outputs Addr(3:2) which are used as a word counter. System I/O or main memory is only required to acknowledge each of the 4 words with the $\overline{\text{RdCEn}}$ input which is used as a read clock enable to latch each word into the 4-deep read buffer. Read interfacing also has the option of using the $\overline{\text{Ack}}$

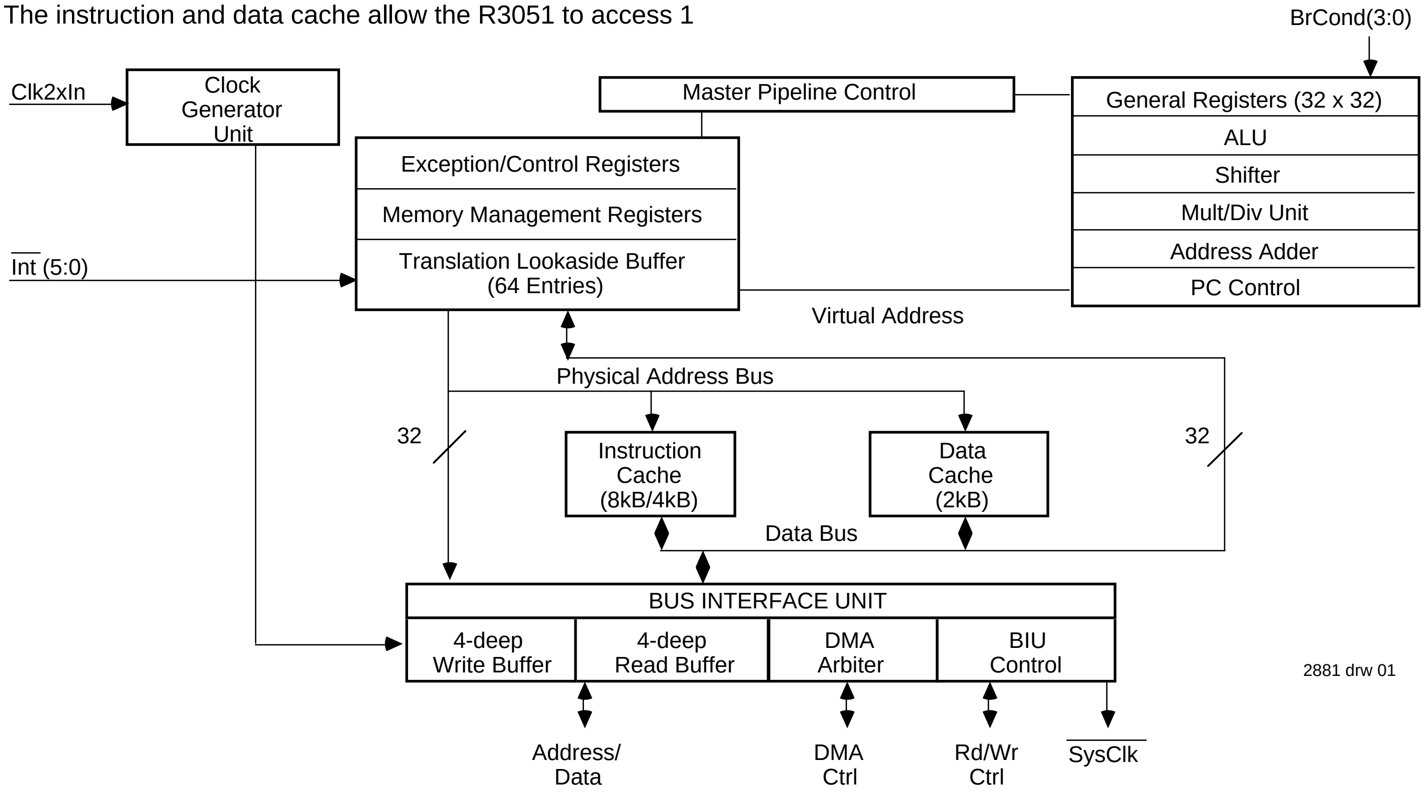

Figure 1. R3051 RISController Internal Architecture

acknowledge input signal to optimally control when the R3051 core restarts its pipeline on burst read cycles.

The simple write interface allows a wide range of memories and I/O to be used with the R3051 by buffering writes from the R3051 core which are done at cache speeds. This allows main memory and I/O to retire write cycles at their own rate of speed by returning $\overline{\text{Ack}}$, to acknowledge that the word has been received as shown in Figure 4.

Basic System Functional Blocks

The following sections will describe the functional blocks that are typical of R3051 main memory and system I/O interfacing. As shown in Figure 5 these blocks include:

- Address De-multiplexing
- Address Decoding and Chip Selection
- Data Transceivers
- Wait-State Controller and Interface Handshaking
- Read/Write Enables and Strobes

The discussion concentrates on the general interface blocks involved when using the following modules:

- SRAM Interfacing
- DRAM Interfacing
- EPROM Interfacing
- I/O Interfacing
- DMA Interfacing

Specific information on using the different memory and I/O types is presented in detail in other application notes.

ADDRESS DE-MULTIPLEXER AND DECODER

The R3051 uses a multiplexed A/D(31:0) bus to output its address and to send and receive data. Thus main memory must de-multiplex the address by using the R3051's Address Latch Enable control signal, ALE, before decoding the address to select chip enables.

Latching A/D(31:0)

Transparent latches such as the IDT54/74FCT373 and the IDT54/74FCT841 pass inputs straight through to the outputs when their Latch Enable input is high. When their Latch Enable input is low, the data in the latches are held constant. The R3051 provides the ALE output for direct connection to the transparent latches' Latch Enable pins. Transparent latches are typically used to allow address decoding to take place when ALE is high and the address begins to become valid, instead of waiting until the latch closes.

The Address Latch Enable, ALE, is designed to clock the address into a transparent latch such as the FCT373. ALE is

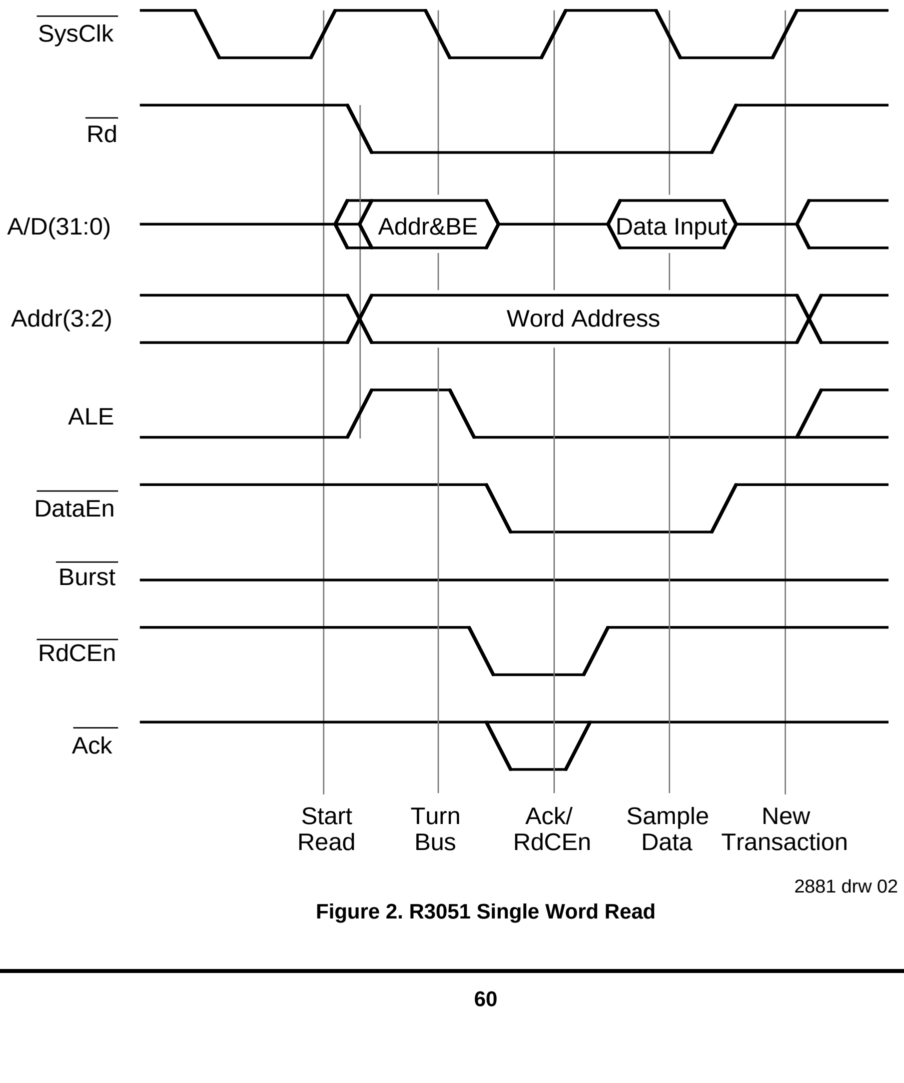

Figure 2. R3051 Single Word Read

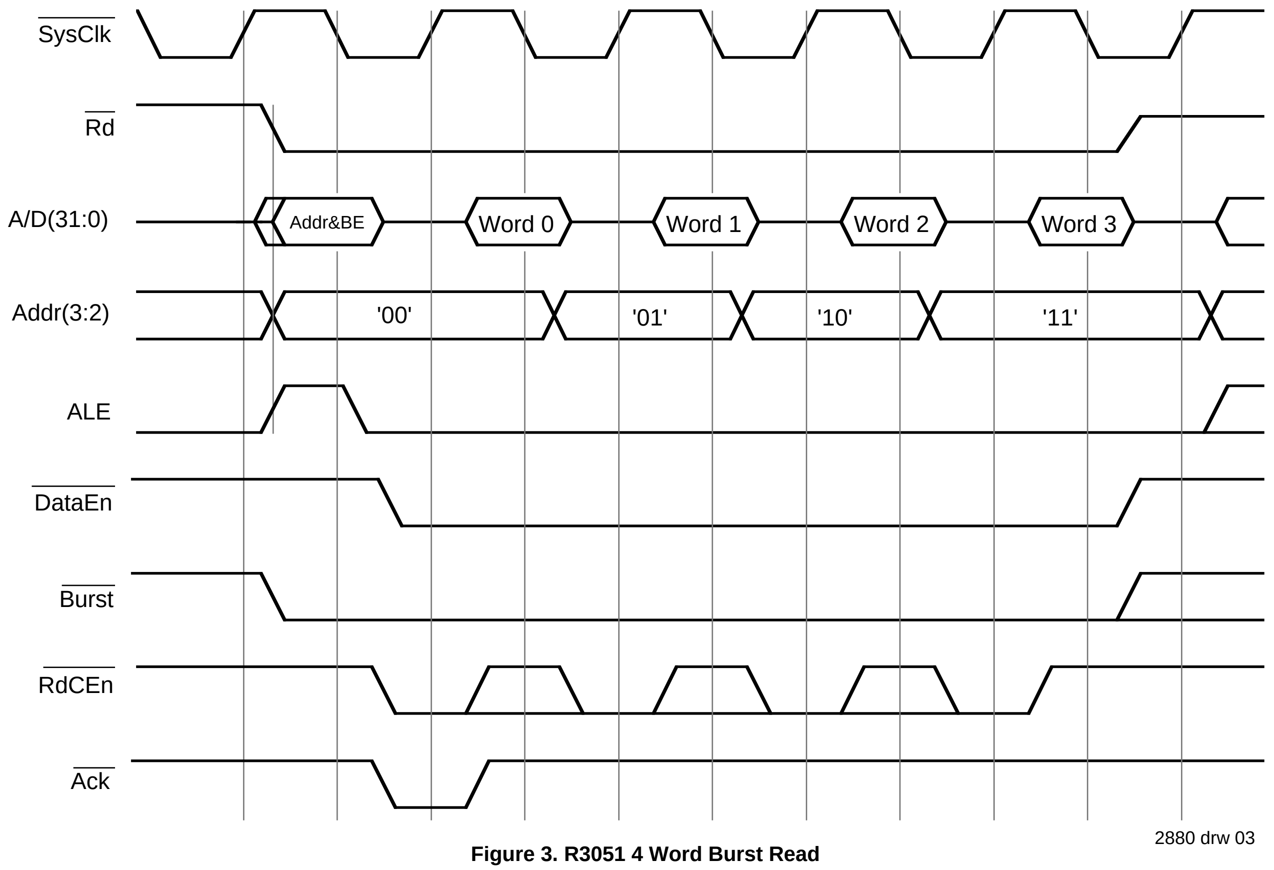

Figure 3. R3051 4 Word Burst Read

also designed to meet the address hold time of latches. As with all high speed processors, ALE should be considered a critical signal. Thus Printed Circuit Board routing should minimize ALE's trace length and crosstalk susceptibility.

Decoding A(31:0)

Address decoding, which selects between the various memory and I/O banks in the system, can be done with IDT54/74FCT138/139 decoders as shown in Figure 6.

The time for the main memory chip selects to become valid in such a scheme is:

$$t_{Decode} = \max(t_{3051ALEProp} + t_{373LEtoO}, t_{3051AddrProp} + t_{373DtoO}) + t_{138Ato\overline{O}} + t_{Cap}$$

Systems that require the chip selects to not have decoding glitches while the address drives to a valid value can register the decoder outputs by using $\overline{\text{SysClk}}$ as the clock and a $\overline{\text{CycleStart}}$ signal as the clock enable. The $\overline{\text{CycleStart}}$ signal is derived from the $\overline{\text{Rd}}$ and $\overline{\text{Wr}}$ control lines so that it asserts at the beginning of every memory cycle.

Decoding Byte Enables with Chip Selects

During the address phase, the R3051 uses the lower 4 bits of the multiplexed A/D(31:0) bus to output $\overline{\text{BE}}$(3:0). Byte enables are used to determine which bytes of each word are being read or written to support partial word accesses. Because $\overline{\text{BE}}$(3:0) are used throughout the memory cycle, they are latched by ALE along with the other A/D bits.

In general, it is permissible to process all reads as 32-bit reads—the processor will only take the data it requested from the bus. However, in write operations, the system must insure that only the specified bytes are written. Thus, the byte enable outputs are used to control this.

There are two ways in which the byte enables may be used:

- Gate the byte enables with the memory chip selects. Thus, only those bytes of memories which will be written are selected. A single write enable can then be presented to all banks of that memory subsystem. This solution requires that each memory sub-system further decode the chip-selects, and thus one decoder per memory sub-system is required.
- Gate the byte enables with the memory chips read/write enables/strobes. Thus, although all of the devices in that bank of memory are "selected", only those bytes to be written are enabled for the writes. This is a common strategy in DRAM sub-systems. Note that the individual byte strobes may be broadcast to all memory systems, and the address decoder will insure that only one sub-system is "Selected". Thus, a single decoder for byte enables can serve the entire memory system.

If the memories being used are 1-bit to 8-bits wide, gating the byte enables with the chip selects can be done. Because the byte enables are predetermined within the R3051 by using the LSB address bits, the endianness of the system, and the

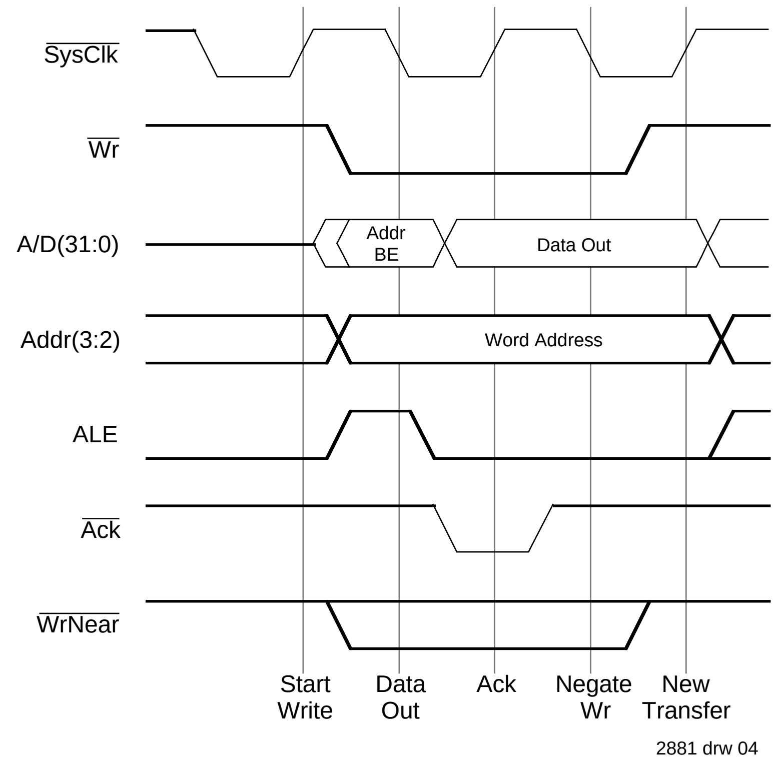

Figure 4. R3051 Single Word Write

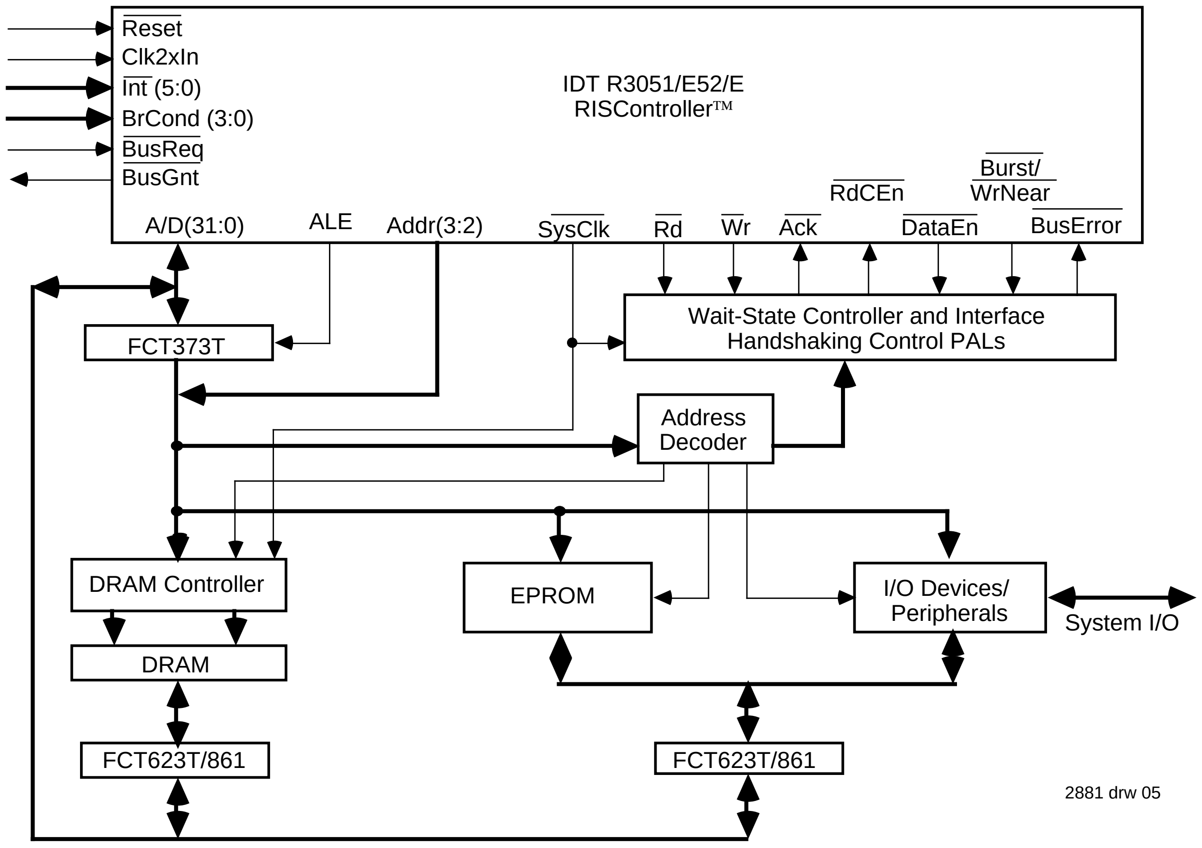

Figure 5. R3051 with Main Memory

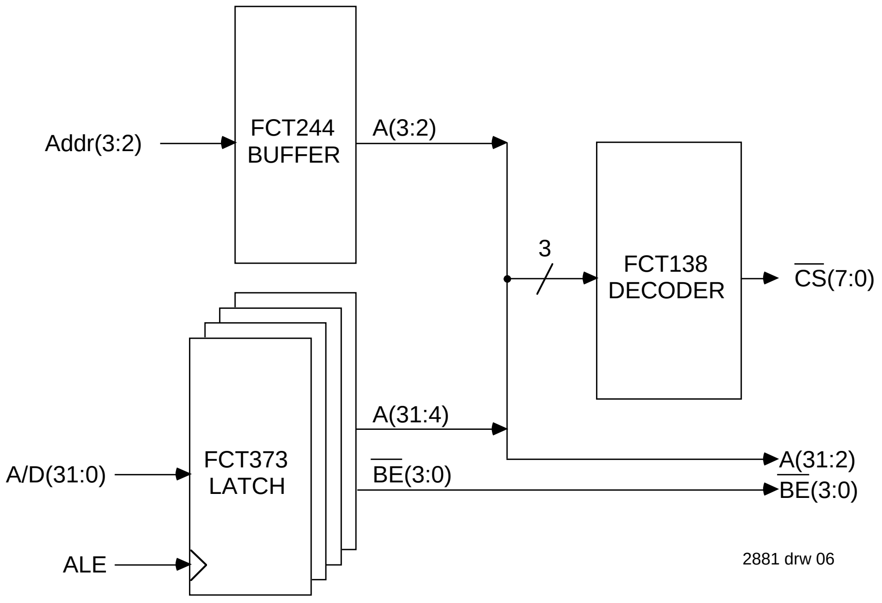

Figure 6. Address De-multiplexer and Decoder

type of load or store instruction, the byte enables have the same timing as the rest of the A/D lines during the address phase when ALE is asserted. This allows a memory decoder to have individual chip selects for each byte of each bank with no timing penalty. An example is shown in Figure 7.

As gating the byte enables with the chip selects usually takes more output pins than gating the byte enables with the read and write enables, the latter is usually preferred. The use of byte enables with read/write enables will be discussed in the read/write enable/strobe section.

Using Addr(3:2)

Since the lower 4 A/D bits are used for byte enables during the R3051's address phase, the R3051 provides the information for addressing words through its Addr(3:2) output pins. The R3051 uses 4 bytes per word and pre-decodes the byte enables instead of providing the 2 LSB address lines. Addr(3:2) are driven throughout external bus cycles and do not require latching. During non-burst read cycles and all write cycles, Addr(3:2) contains the instruction cache miss address. The advantage of dedicating output pins for Addr(3:2) is that during burst read cycles, Addr(3:2) are incremented from 0 to 3 by the R3051 $\overline{\text{RdCEn}}$ protocol so that the system memory system does not have to provide a counter for this function.

Since each memory chip requires Addr(3:2), large memory systems that use Addr(3:2) extensively may want to use buffers. A common strategy may be to provide a buffered version of Addr(3:2) to non-time critical areas of memory (e.g. the boot prom), or to areas which do not perform burst accesses (I/O devices), and directly use the outputs of the R3051 in time-critical areas such as the DRAM control.

The crossover point where buffering is appropriate can be determined by determining if the delay through an IDT54/74FCT244 buffer and the capacitive derating from all the Addr(3:2) inputs driven by the buffer (Addr(3:2) can be buffered for separate branches of memory banks) would be less than the delay from the capacitive derating from all the Addr(3:2) inputs driven directly from the R3051. In addition, the crossover doesn't occur until Addr(3:2) is delayed past when rest of the A(31:4) lines reach their inputs.

$t_{3051Addr(3:2)} + t_{244} + t_{244Cap} \leq \max(t_{3051Addr(3:2)} + t_{3051Cap}, t_{A(31:4)})$

where:

$t_{244Cap} = (\text{sum}(C_{Input/Output}) + C_{244} + t_{Trace} - 50)/33$ pf/nsec

$t_{3051Cap} = (\text{sum}(C_{Input/Output}) + C_{3051} + t_{Trace} - 25)/25$ pf/nsec

Using Diag(1:0)

Some systems may need to know whether a read cycle is cachable or uncachable and whether a cachable read cycle is an instruction or a data fetch. In Figure 8, this information is provided by latching the diagnostic pins, Diag(1:0) with the same latch controls as the address lines. These signals are useful for:

- Decoding whether a reference to the lowest half GB of physical memory is from kseg0 or kseg1.
- Tracing processor execution by knowing which address caused the I-Cache miss.

DATA TRANSCEIVERS

The R3051 uses a multiplexed A/D(31:0) bus to output its address and to send and receive data. Thus main memory must drive or receive data after the R3051 has tri-stated its address. Further, to support high-performance memory systems, the R3051 family is capable of initiating a new bus transaction one-half clock cycle after data is sampled for a read operation.

Determining if Data Transceivers are needed

Multiplexed CPU busses often use data transceivers to separate the memory system from the processor bus. Read cycles require the memory system to stop driving data on the A/D bus before the processor drives the next memory cycle's address. Slow memories with relatively long output disable times cannot meet this limitation without data transceivers. However, some memories, such as the IDT71B256 BiCEMOS™ 32Kx8 Static RAM, have very short access time and output disable time which makes it possible to consider attaching memory device data I/O pins directly to the multiplexed A/D(31:0) bus. Alternatively, in low frequency systems, the amount of time provided by the R3051 may be sufficient for the memory devices attached to the bus.

The key parameter is the memory output disable time, TOZ, which has to be less than 1/2 clock to disable before the next memory's address is driven. In addition the address and data driven from the R3051 is delayed because of the extra capacitance of the memory data I/O pins.

$t_{OZ} \leq t_{SysClk/2} - t_{DisableControl} + \min(t_{3051Addr})$

Data Transceivers also serve to isolate memory banks from each other. In systems with varying speeds of memory, transceiver banks can be used to separate chips with relatively long output disable times from those with relatively quick output disable times. Thus in many systems, fast scratch-pad SRAMs may have their own set of transceivers, while slower EPROMs and I/O peripherals might have a separate set of transceivers.

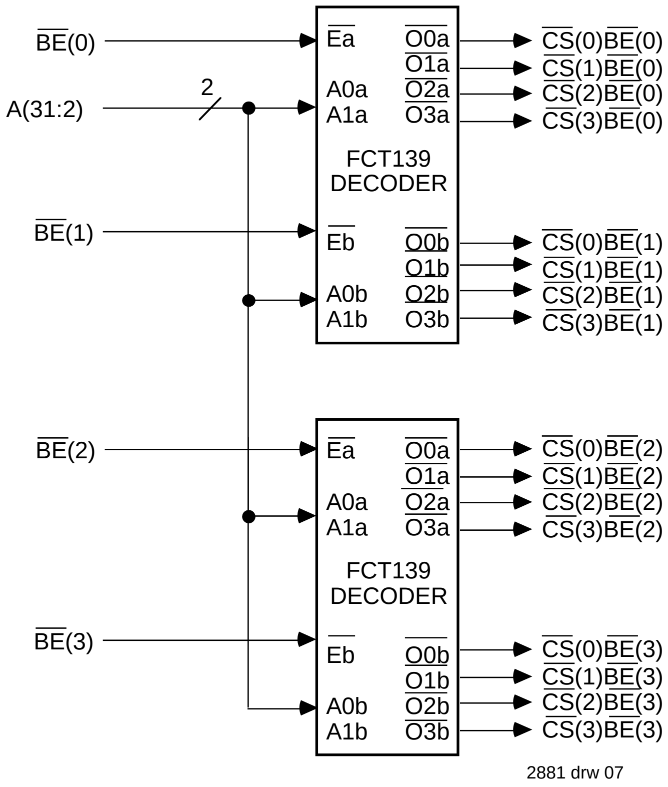

Figure 7. Gating Byte Enables with Chip Selects

Using IDT54/74FCT861's and IDT54/ 74FCT245's for Data Transceivers

Most systems will use slower memories and thus require data buffering through a transceiver interface. There are two basic families of transceiver interfaces:

1: IDT54/74FCT861 with separate enable pins for each direction
2: IDT54/74FCT245 with a direction pin and an enable pin

Using IDT54/74FCT861's for Data Transceivers

The 10-bit transceiver FCT861 approach functionally combines two 10-bit tri-statable FCT827 buffers internally. The 8-bit FCT623T transceiver is similar to the FCT861 except that one of its output enables is active high. On read cycles, if there is only one transceiver bank, then DataEn can be used directly to control the read direction output enable. Otherwise, combinational logic such as an FCT157/257 multiplexer can be used to combine DataEn with the chip selects of the bank whose transceivers need to be enabled (see Figure 16 for a similar common input OR gate circuit). Alternatively, some transceivers, such as the 9-bit IDT54/74FCT863 and the 8-bit IDT54/ 74FCT543 have two logically AND'ed output enables for each direction so that DataEn and the bank chip select can be hooked up directly to the transceiver. State machines using an inverted SysClk can also use a Rd derived signal to synchronously assert and de-assert the read direction output enable.

The write direction output enable can use a signal derived from Wr which asserts at the beginning of the cycle and waits until after the data has been strobed into the memory or I/O device before de-asserting to provide sufficient data setup and hold time. For systems with 1 wait-state or more, the derived write direction enable signal should ideally assert after the A/D bus finishes driving its address phase to reduce switching noise.

The transceiver control's critical timing path is the transition from a read cycle to a write cycle. After a read cycle, slower memory chips take a relatively long time to disable from the data bus. If the next memory cycle is a write, the transceivers will drive data onto the same bus. Such systems can use the second memory cycle's wait-states to delay the assertion of the transceiver's write direction output enable until the first memory cycle's memory has fully disabled. The cutoff for

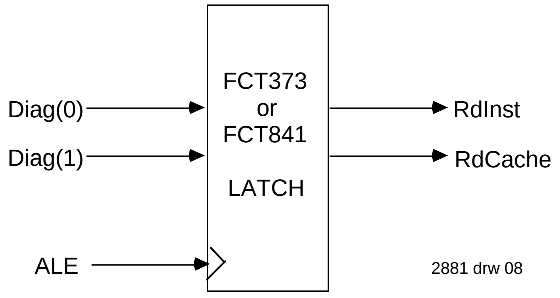

Figure 8. Latching Diag(1:0)

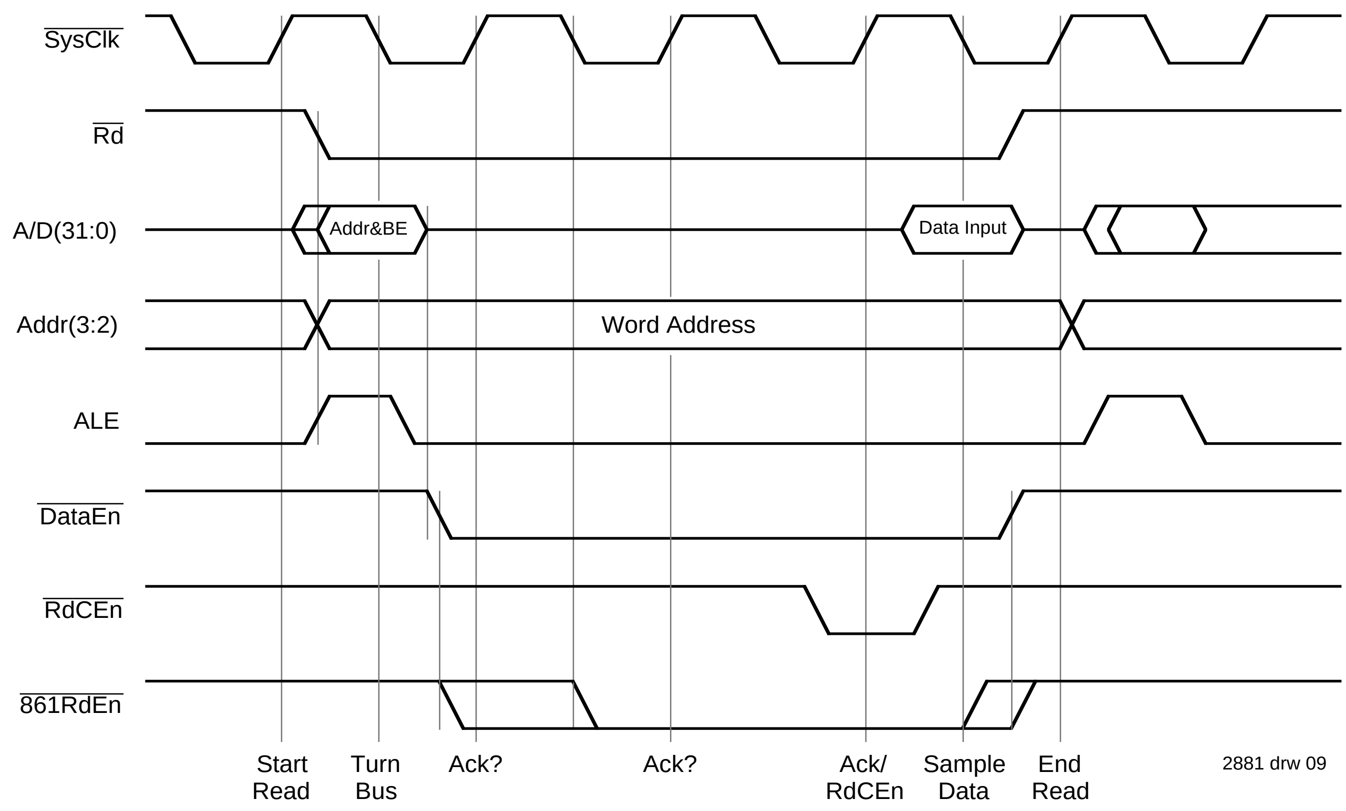

Figure 9a. Timing Diagram of FCT861 Read Direction Enable

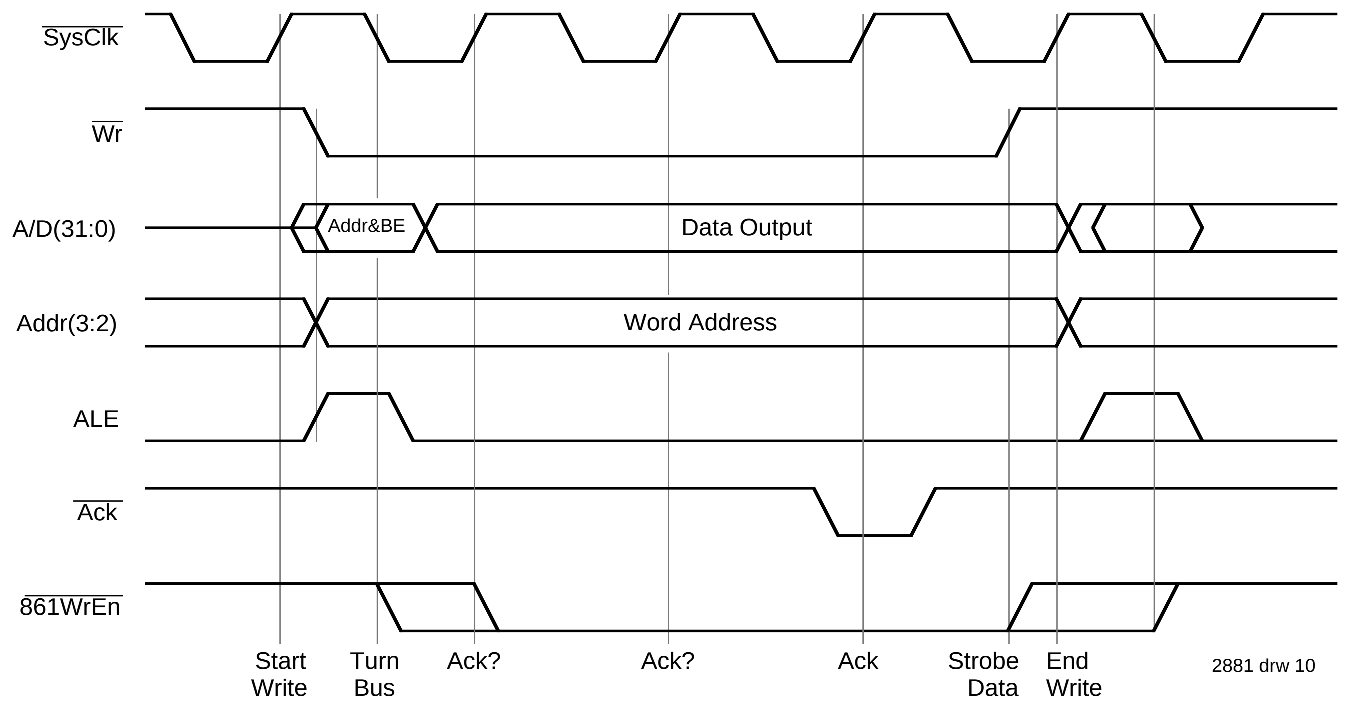

Figure 9b. Timing Diagram of FCT861 Write Direction Enable

determining if the memory output disable time is small enough to require no wait-states is:

$t_{SysClk} >= t_{DisableControl} + t_{MemReadDisable} - t_{WriteData}$

Systems that use memory chips without an output enable pin (i.e., a read is implied for every chip select with no write enable) require special transceiver interfacing in order to support partial word writes. During partial word writes, where only some of the bytes are selected for writing, bytes which are not being written may actually output onto their byte lanes, and thus conflict with the transceiver write direction outputs. In such memory sub-systems, there are two options: only chip select those devices actually being written into; or, only enable those transceivers whose byte lanes are used in this write transfer. Either of these solutions will insure that no bus conflict occurs.

Using IDT54/74FCT245's for Data Transceivers

The 8-bit FCT245 transceiver approach ideally requires that the direction control only be changed when the outputs are disabled to prevent bus contention. Although such systems are easy to design, this general discussion uses the following assumptions:

1: Either a SysClk or SysClk based state machine is used.
2: The memories require at least 1 wait-state.

The output enable of an FCT245 needs to be determined by finding the start and end of the memory cycle, which can be determined by logically AND'ing $\overline{\text{Rd}}$ and $\overline{\text{Wr}}$. The assertion of the output enable can be easily delayed to occur well after the transfer, depending on the number of wait-states in the memory controller. That is, the transceiver only needs to be enabled in time to allow the data to propagate through to the CPU as the read data response is finally returned to the processor. In read cycles, the output may be disabled using the same clock edge as is used by the CPU to negate $\overline{\text{Rd}}$. On write transactions, the transceiver must be enabled until the data set-up and hold time requirements of the memory being written are met, which may extend until the next falling edge of $\overline{\text{SysClk}}$ (note for the R3051, the processor guarantees that valid data will remain for one-half clock cycle after the negation of $\overline{\text{Wr}}$).

The T/$\overline{\text{R}}$ direction pin of the FCT245 should be asserted before the output enable asserts, which can be achieved by using a $\overline{\text{Rd}}$ or $\overline{\text{Wr}}$ derived signal. The direction should be held until the next clock edge after $\overline{\text{Rd}}$ or $\overline{\text{Wr}}$ de-asserts; that is, until after the output enable is de-asserted..

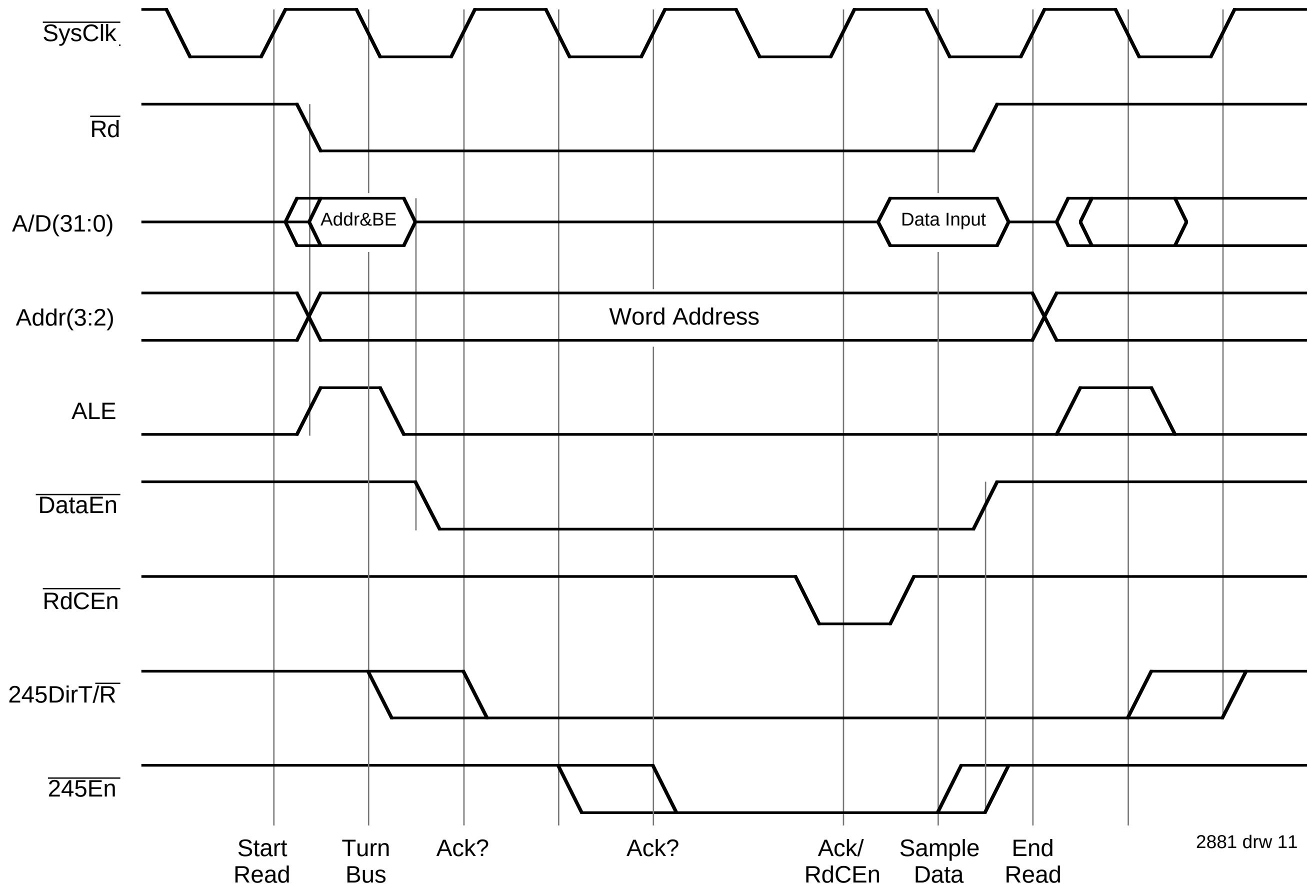

Figure 10a. Timing Diagram of FCT245 Enable and T/$\overline{\text{R}}$ Direction Controls for a Read

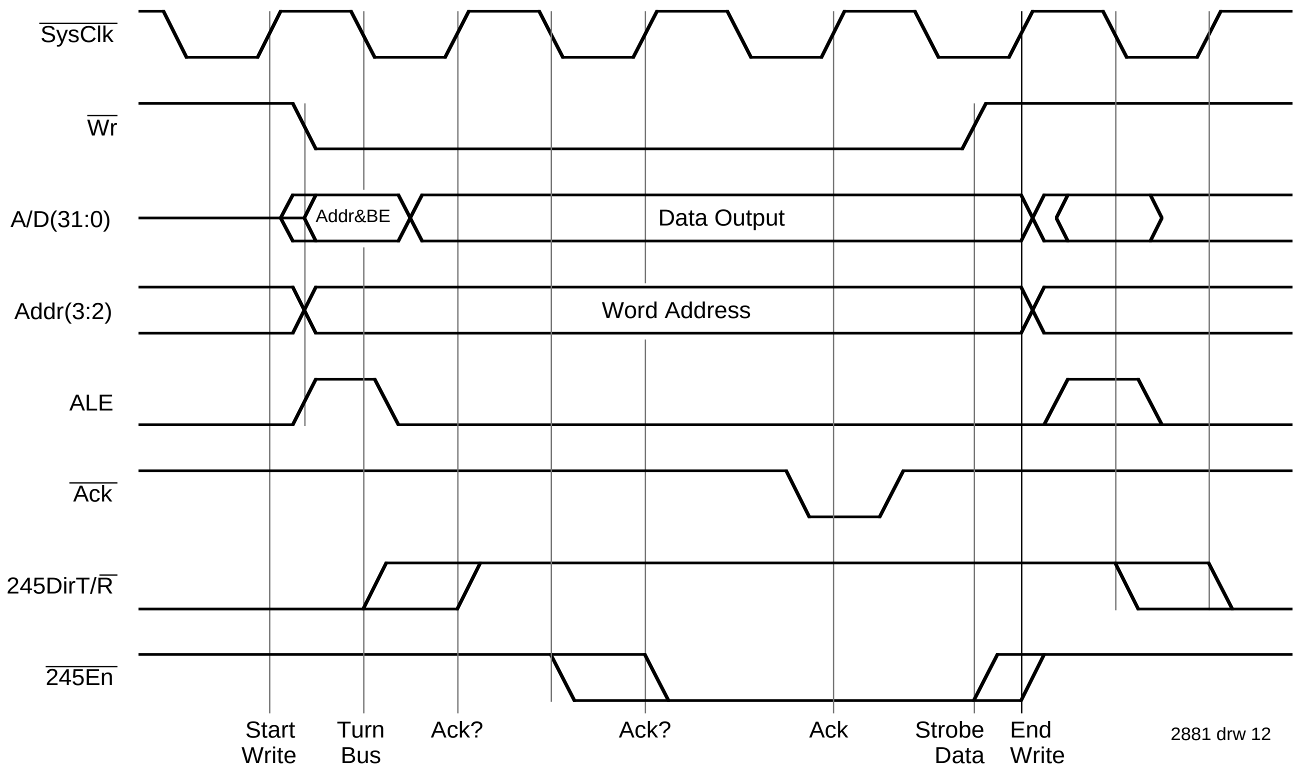

Figure 10b. Timing Diagram of FCT245 Enable and T/R̄ Direction Controls for a Write

Systems that use memories without a dedicated output enable pin require separate byte output enables in the data path, as discussed above.

PULL-DOWN/UP RESISTORS ON R3051 OUTPUTS

The R3051 tri-states its outputs under three conditions:

1: If no external read or write memory cycles are being executed, the A/D bus will tri-state. Control signal outputs will be driven to negated states.
2: If a DMA bus grant is given, all bus interface outputs will tri-state.
3: If the Tri-State reset mode has been invoked, all outputs except SysClk will be tri-stated.

The following paragraphs detail which outputs are affected when the R3051 is in a tri-stated condition.

Pull-down/up Resistors on the A/D Bus

The R3051 tri-states the A/D bus when it finishes a write (or read) cycle and there is not another pending memory cycle that it needs to execute. This situation occurs when the R3051 is getting instructions from its internal instruction cache and it executes a sequence without store instructions. Since the A/D bus can be tri-stated for these periods, it is desirable for the input pins of the address latches and data transceivers to maintain the A/D bus with defined, valid logic values by using pull-up/pull-down resistors. The use of pull-up or pull-down resistors also has the benefit of easing Automatic Test Equipment programming on board-level and in-circuit tests.

Pull-down/up Resistors on Control Lines for DMA

The R3051 has an on-chip Direct Memory Access (DMA) arbiter that allows outside processors and controllers to take control of the external memory systems, and perform transactions. It does this by indicating a request to the R3051, which then tri-states its bus interface to allow it to be driven by the external agent.

During DMA, the R3051 will execute instructions from its internal caches until it has a cache miss, makes an uncacheable reference, or its write buffer becomes full.

An external agent requests bus mastership by asserting the R3051 BusReq input. If BusReq is asserted by the DMA device, the R3051 tri-states its outputs and asserts BusGnt to signal to the DMA device so that it can begin to drive its own memory cycles. During DMA, the R3051 tri-states all outputs except SysClk and BusGnt. During the time that the R3051 and the DMA controller transfer control back and forth, neither one drives the control line outputs (to avoid bus conflicts). In order to properly transfer control, the R3051 control outputs should be kept in their de-asserted state. If the transfer time is relatively short, the system designer may choose to rely on bus capacitance to hold these signals in their negated positions. Alternatively, a more conservative strategy is to hold the bus in a negated position with pull-down or pull-up resistors. Thus Rd, Wr, Burst/WrNear, and DataEn should use pull-up resistors and ALE should use a pull-down resistor.

Pull-down/up Resistors on Control Lines for Tri-State

The R3051 has a reset mode vector which allows the chip to tri-state all its outputs, except SysClk. This mode is attained by asserting Tri-State via SInt(1) while Reset is asserted. In addition to the control lines above, BusGnt is tri-stated. Thus for Automatic Test Equipment programming on board-level and in-circuit testing, a pull-up resistor for BusGnt can be used.

WAIT-STATE CONTROLLER LOGIC

Wait-states are used to extend the number of clocks within a memory transfer to provide sufficient memory access and data setup time for the particular type of memory being accessed. Such control can be provided with a wait-state controller state machine. In general, a wait-state machine has four steps:

1: Detect the beginning of a memory cycle
2: Determine the type of cycle:
  a: Which chip select (address decode)
  b: Read or write
  c: Single word or burst, write near or non-page write
3: Count out cycles until memory is ready and assert R3051 handshaking signals
4: Acknowledge the end the cycle

Thus, the basic control strategy is to use a counter which is held at zero until a cycle is started, and which then increments every clock cycle until the transfer is completed. This master counter then provides the reference by which control outputs to the memory, data path, and CPU are provided.

R3051's use of both Clock Edges

The R3051 uses both edges of the clock to assert and de-assert its control signals. This is to ameliorate the fixup time between memory cycles, which for most processors, takes 1 full clock cycle. The R3051 is able to do the fixup in 1/2 clock cycle. This would seem to complicate the design of state machines which must latch these signals synchronously to one edge or the other. However, as will be shown in the following sections, a traditional state machine that follows a small number of simple design rules can still use a single edge clock.

The R3051 uses an input clock, Clk2xIn, that runs at twice the frequency of the processor. The R3051 provides an output clock, SysClk, that runs at the same frequency as the processor and can be used to clock external state machines. The polarity of SysClk was chosen intentionally so that either an unbuffered SysClk or an inverted version of SysClk, (referred to here as SysClk) can be used. Because all the R3051 control outputs have very short propagation delays (less than 1/2 clock), a state machine can use either edge of SysClk.

In developing the set of constraints brought on by the use of both the rising and falling clock edges, some observations can be made:

1: All clockable control line outputs, except DataEn assert off the rising edge of SysClk.
2: All clockable control line outputs de-assert off the falling edge of SysClk.
3: All control line inputs required by the R3051 are sampled on the rising edge of SysClk.

Observations 1 and 2 can be specifically applied to two of the primary control signals, Rd and Wr.

1: Rd and Wr both assert off the rising edge of SysClk.
2: Rd and Wr both de-assert off the falling edge of SysClk.

The similarity of edge assertions for Rd and Wr can be used to simplify the wait-state controller.

Detecting the Beginning of a Memory Cycle

State machines looking for the beginning of a memory cycle can look for one of two things:

1: Rd or Wr asserting
2: ALE asserting

In general, state machines have to choose between using SysClk and SysClk. State machines such as those implemented in ASICs can use both clock edges, however, to simplify the discussion it will be assumed that only one or the other clocks is being used. If SysClk is used, certain registers must use SysClk directly from the processor to provide sufficient hold time from the processor. Only a negative edge clocked register can synchronously clock ALE under worst case timing, since ALE is only high surrounding the falling SysClk edge which requires a negative edge triggered flip-flop. SysClk cannot be used because its inverter delay will put it past when ALE could fall.

Machines which use SysClk (the inverted SysClk) will have a delay from inverting SysClk. All state machines can use Rd and Wr to determine the beginning of a cycle. SysClk machines are able to do this easily with wide margins on setup and hold times to its registers. SysClk machines must use SysClk directly from the processor and use registers with 0 hold time and also have a guaranteed minimum clock to output delay to meet the R3051's input hold time.

Determining the type of Memory Cycle

The type of memory cycle usually depends on the following variables:

1: Type of memory
2: Read or write cycle
3: Burst or non-burst, write near or non-page write

These three variables are usually logically AND'ed together to form equations for determining the number of wait-states before asserting RdCEn, Ack, or BusError as well as any transceiver controls. The chip selects from the memory decoder can be used to determine the type of memory to count

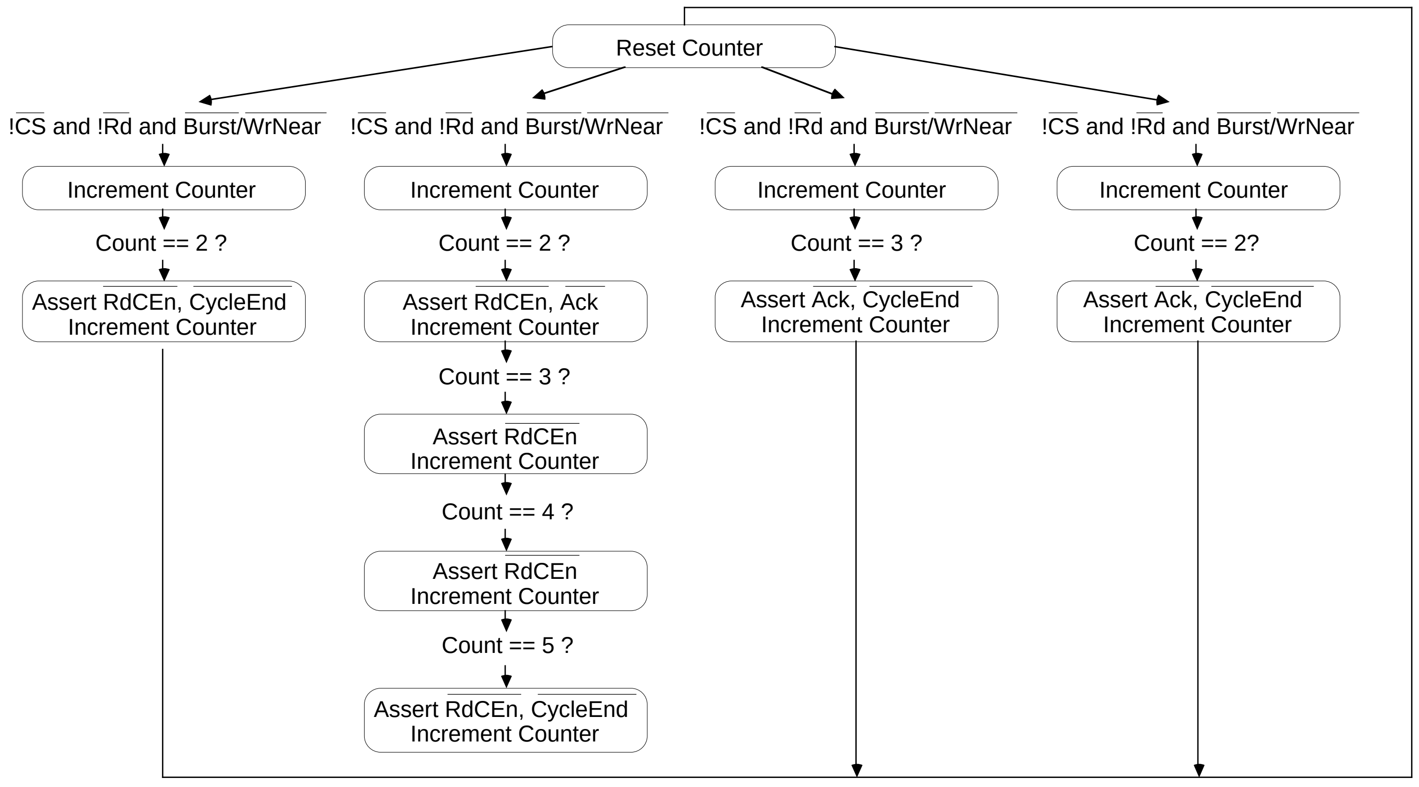

Figure 11. State Diagram of an Example Wait-State Controller for a Single Memory Type

the correct number of wait-states. By using the R3051's Rd and Wr lines, the transceiver controls can be defined. On read cycles, the R3051's Burst/WrNear line determines if 1 word or 4 words are to be returned. On write cycles, Burst/WrNear determines if a consecutive write is on the same 256 word page as its predecessor. An example of a state transition diagram that uses the read/write and burst/non-burst variables for one memory type is shown in Figure 11. Each memory type in the system also has a state diagram.

Further variables that affect the type of memory cycle are implied by the mode initialization vector which is supplied during processor reset initialization. The variables determine whether the data byte ordering is Big or Little Endian and whether data cache miss refills are handled one word at a time or as 4 word block refill reads. BigEndian and DBRefill are set by multiplexing the interrupt lines on the de-assertion of reset, an example of which is shown in Figure 12.

The mode vector of the R3051 was chosen to allow it to be supplied by just using pull-up resistors on the appropriate interrupt inputs. For example, the multiplexer shown in Figure 12 could be eliminated, and the pull-up resistors tied directly to the SInt(2:0) pins.

Note that to maintain compatibility with future versions of the R3051 family, Int(5:3) should be high when Reset is de-asserted. This also can be performed using pull-up resistors.

Memory Interface Handshaking

The R3051 uses two inputs, RdCEn and Ack,to indicate that the memory system is ready to receive or return data. On read cycles, RdCEn is sampled on the rising edge of SysClk by the R3051 so that it can enable its internal read buffer clock on the next falling edge of SysClk. Thus on single word reads, a single RdCEn is asserted as the memory becomes ready as shown in Figures 2 and 11. On 4 word burst reads, RdCEn is asserted for each of the 4 words. Thus on burst reads, the wait-state controller can optionally "throttle" each word into the R3051 by delaying the return of each word by a varying number of clocks. RdCEn can be generated by gating the memory type and the count:

```
RdCEn not := Reset and CycleEnd and BusError and (
                    (!RamCS and !Rd
                         and ( (Counter == 02H)
                              or (!Burst/WrNear and (Counter == 03H))
                              or (!Burst/WrNear and (Counter == 04H))
                              or (!Burst/WrNear and (Counter == 05H))
                         )
                    )
);
```

The acknowledge input, Ack, has two uses. On burst reads, Ack can be used to optimize the processor execution engine restart. On writes, Ack is used to signal the end of the cycle, as will be explained later. The R3051 throttles burst reads into its internal read buffer at the rate of the memory system; however, it reads data from the read buffer on every clock cycle. Therefore, the R3051 will either wait until the 4th RdCEn has occurred to begin reading the internal read buffer, or until the memory system signals Ack to the processor. Asserting Ack on a burst read cycle causes the R3051 to start reading words from the read buffer in the next cycle; thus, the memory

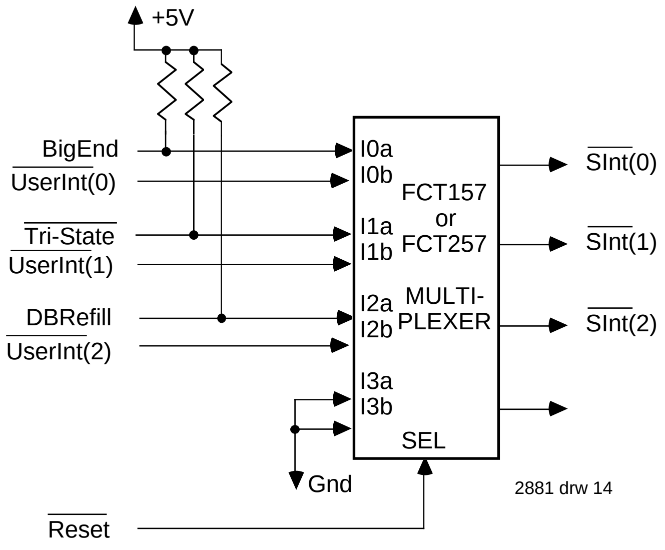

Figure 12. Reset Vector Circuit

system times the assertion of Ack so that the 4th word can be presented by the memory system just before it is read from the read buffer. Thus for optimal speed burst reads, Ack should be asserted 3 clocks before the last RdCEn occurs, as shown in Figure 3.

On write cycles, Ack is sampled on the rising edge of SysClk by the R3051 so that the cycle ends on the next falling edge of SysClk as shown in Figure 4. Ack is used by the wait-state controller on write cycles to acknowledge that data is being strobed into memory. Ack can be generated by gating the memory type and count.

Note that in writes, the WrNear output from the processor may also affect the write timing. For example, when writing to Page Mode DRAMs, it will be possible to retire near writes faster than non-near writes.

An example of generating Ack from gating the memory type and count is:

```
Ack not := Reset and CycleEnd and BusError and (
                (!RamCS and !Wr
                        and ( ( Burst/WrNear and (Counter == 03H))
                              or (!Burst/WrNear and (Counter == 02H))
                        )
                )
        or      (!RamCS and !Rd
                        and ( (!Burst/WrNear and (Counter == 02H))
                )

);
```

Stopping the Counting

Four common ways to end the memory cycle and stop the counter include:

1: Use a SysClk state machine and look for the de-asserting edge of Rd or Wr

2: Use a SysClk state machine and gate the type of cycle into the counter to reset it independently of the de-asserting edge of Rd and Wr (predict the end of the cycle)

3: Use registers with asynchronous resets and gate Rd and Wr into the reset

4: Interlock a SysClk register looking for the asserting edge of Rd or Wr with a SysClk register looking for the de-asserting edge of Rd or Wr

In method 1, the SysClk registering of Rd or Wr is straightforward. However, if the counting is based on SysClk, the state machine will not be able to bring Ack or RdCEn low during the first possible clock cycle that they are sampled for by the R3051. This is, because the state machine will not detect the assertion of Rd or Wr in time. This implies that a SysClk based state machine will have a minimum of one or more wait-states.

In method 2, SysClk based state machines must determine when to stop counting independent of the de-assertion of Rd or Wr. In general they cannot use Rd or Wr to terminate the cycle because Rd or Wr may de-assert within the buffered (inverter delayed) SysClk register's setup or hold time. Thus SysClk based state machines should use its counter to determine when the cycle will end, e.g., with CycleEnd. CycleEnd or a similar signal uses the chip selects and a counter to determine the end of the memory cycle, without using the de-asserting edges of Rd and Wr. Logic equations for CycleEnd and the LSB of an N-bit binary up counter look like:

```
CycleEnd not := Reset and CycleEnd and (
                        (!RamCS and (Counter == 02H) and !Rd and Burst)
                        (!RamCS and (Counter == 05H) and !Rd and !Burst)
                        (!RamCS and (Counter == 03H) and !Wr and Burst)
                        (!RamCS and (Counter == 02H) and !Wr and !Burst)
                        ({Bus Error Timeout} (Counter == 0FH))
);

Counter(0) := Reset and CycleEnd and BusError and (!Rd or !Wr)
                                                  and (Counter(0) xor 1)
;
```

A Timing Diagram of CycleEnd showing how CycleEnd asserting at the end of the memory cycle will reset the wait-state counter independently of Rd and Wr is shown in Figure 13.

Counters using CycleEnd use the type of cycle to determine when the wait-state counter should stop and reset independent of the de-asserting edge of Rd or Wr.

Wait-state machines implemented in ASICs can consider using method 4 which involves interlocking SysClk and SysClk based registers as shown in Figure 15. ASICs can also selectively combine two independent SysClk and SysClk state machines to avoid 1/2 cycle interlock timing constraints.

Bus Errors

Bus errors can be handled by timing out with the wait-state controller counter as it is about to overflow. For all types of memory cycles, the R3051 de-asserts its control edges, e.g.,

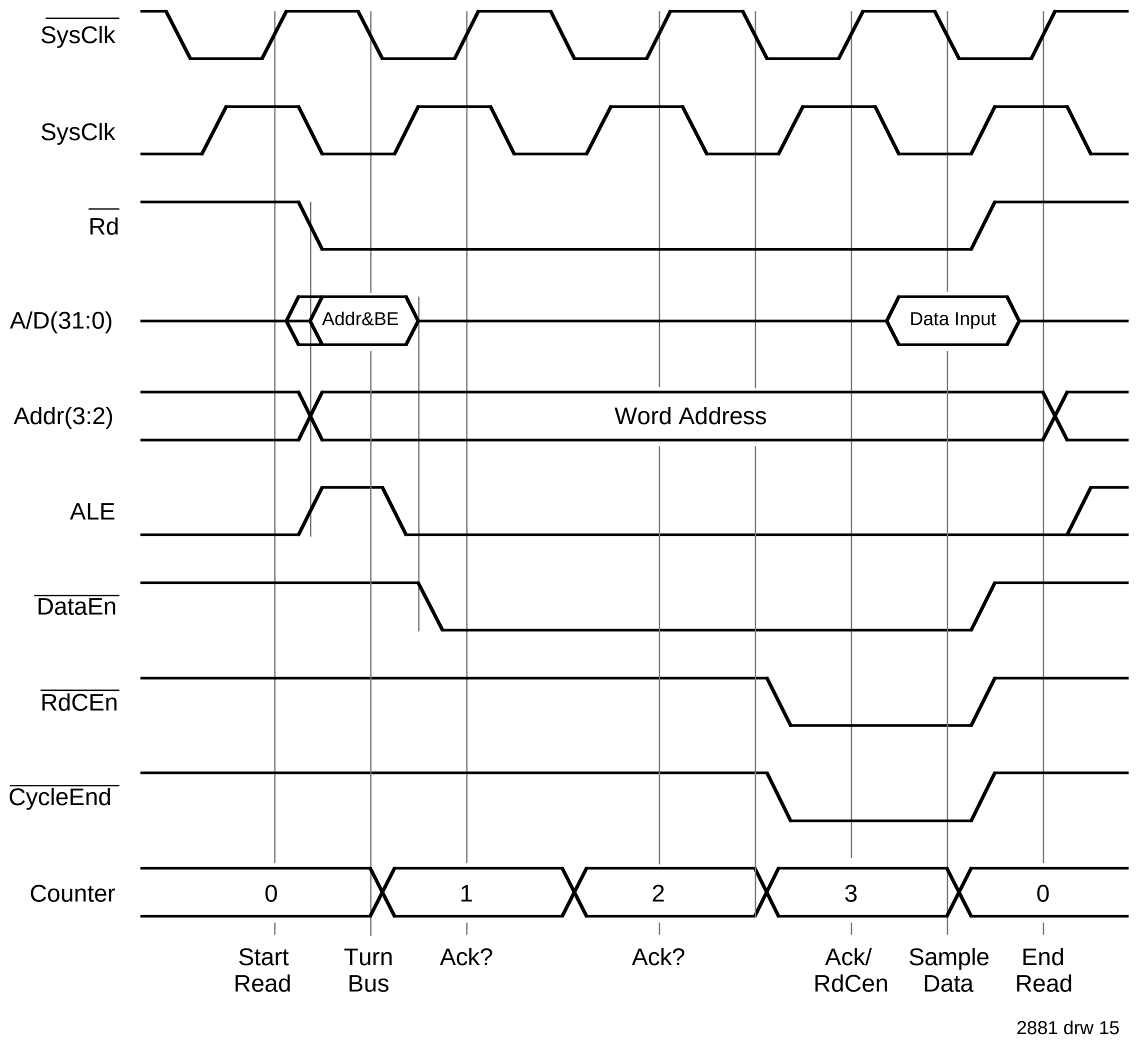

Figure 13. Timing Diagram of $\overline{\text{CycleEnd}}$

$\overline{\text{Rd}}$ or $\overline{\text{Wr}}$, on the clock following the assertion of $\overline{\text{BusError}}$. $\overline{\text{SysClk}}$ based state machines can look for the de-asserting edge of $\overline{\text{Rd}}$ or $\overline{\text{Wr}}$ in order to reset the wait-state machine's counter. In SysClk based state machines, $\overline{\text{BusError}}$ can directly reset the wait-state machine's counter or the overflow count can be used to assert $\overline{\text{CycleEnd}}$ which will then reset the counter.

Bus errors signal an exception to the R3051 only if it is a read cycle. If exceptions need to be noted for write or DMA cycles, $\overline{\text{BusError}}$ should be gated into an interrupt line. The interrupt must be held until the R3051 can acknowledge it, since the R3051 re-registers its interrupt inputs on each clock cycle in which it is executing instructions in its run or fixup state.

READ ENABLES AND WRITE ENABLES

Memories and I/O devices have a combination of chip selects, read enables, and write enables to drive data out of the device and to strobe data into the device. Because the exact timing and functions of the selects, enables, and strobes differ for DRAM, SRAM, and I/O, this section discusses read and write enables and their relationship to the byte enables.

Read Enables

In general, a memory or I/O device has an output enable pin to enable its data outputs on a read cycle. Typical designs will address all 8-bit and 16-bit I/O devices using 32-bit word addressed, (i.e., use Addr(3:2) as their LSBs). Even though the R3051 produces byte enables on read cycles, it is rare to require use of the byte enables for reads as the R3051 will internally mask the bytes not being used. The output enable for the device can be derived from $\overline{\text{Rd}}$ or from $\overline{\text{DataEn}}$.

If more than one memory device uses a single transceiver, it may be necessary to generate device Output Enables using a delayed version of $\overline{\text{DataEn}}$. If one of the memory or I/O devices has a long output disable to tri-state time, then extra time must be allowed for that device to tri-state before another device is enabled. An equation determining if the read enables should be delayed on a back to back read cycle is:

$$t_{SysClk} >= t_{DisableControl} + t_{OldMemoryDisable} - t_{NewMemoryData} + t_{Cap}$$

The output enable control should be asserted at least until the clock cycle that $\overline{\text{Rd}}$ and $\overline{\text{DataEn}}$ de-assert to provide sufficient data hold time to the R3051.

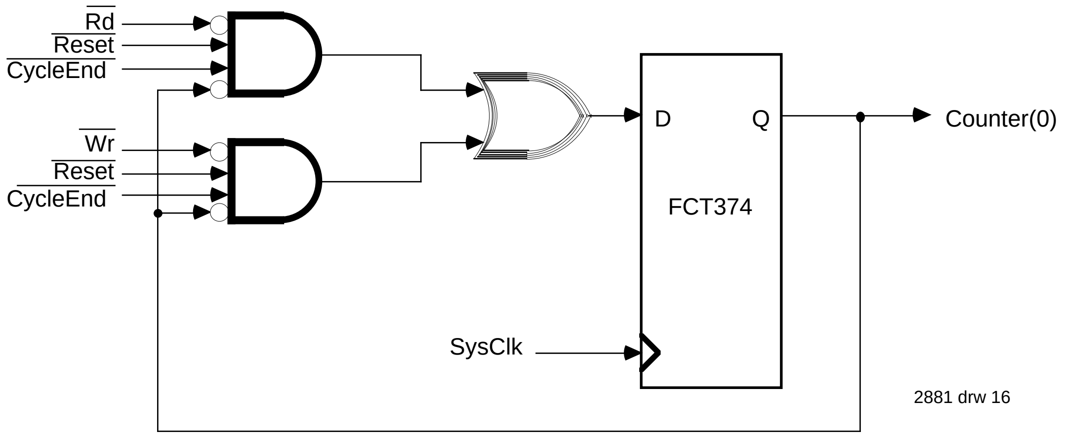

Figure 14. Using $\overline{\text{CycleEnd}}$ in a SysClk Based Counter

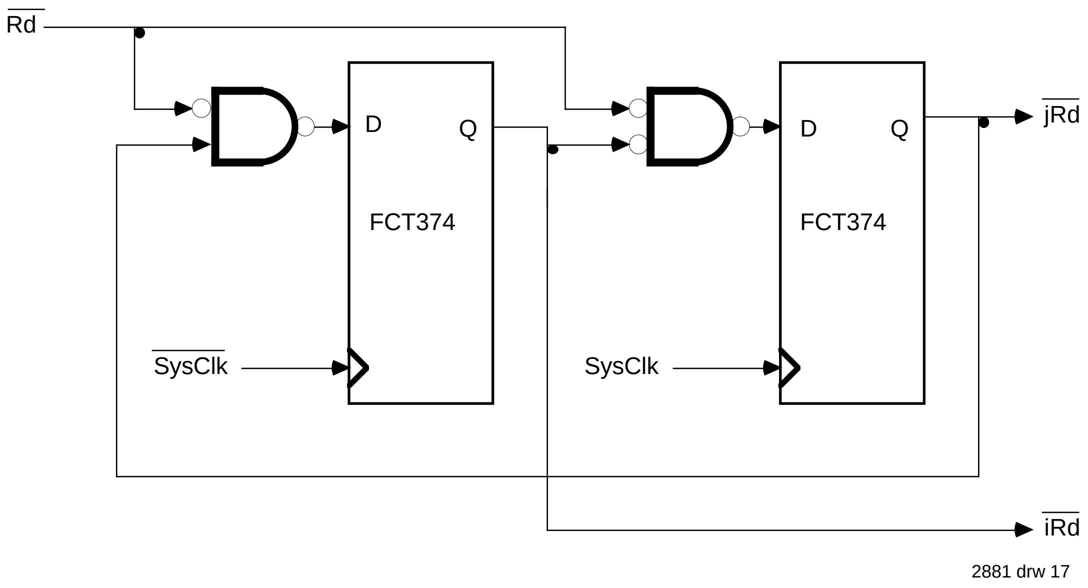

Figure 15. Using Interlocked Registers

Gating Write Enables and Byte Enables

Memory and I/O devices have a write enable pin or a similar protocol to strobe data into the device. A special case occurs for partial word stores, where only the pertinent bytes of a word have their byte enables asserted. Partial word stores occur when a store byte, store half-word, or store tri-byte instruction is executed. Because of the efficiency and optimization capabilities of modern compilers, such as the MIPS™ and IDT Compilers for the R3000™ family, the hardware must always assume that the software will make use of the partial word store instructions. Thus the write enables (or as shown earlier the chip selects) of each byte of a word must be gated with their respective byte enables. Gating the byte enables into the write enables can be done with an FCT157/257 multiplexer by configuring it as a set of four OR gates with a common input term as shown in Figure 16. The write enable signal can be derived from $\overline{\text{Wr}}$.

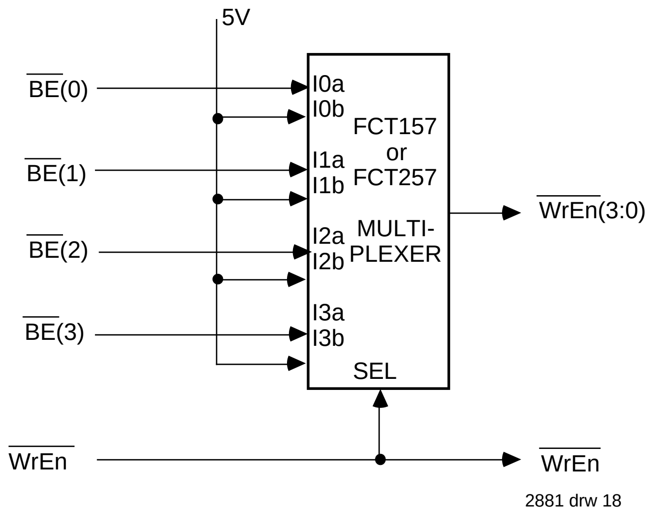
5V
$\overline{\text{BE}}$(0)
$\overline{\text{BE}}$(1)
$\overline{\text{BE}}$(2)
$\overline{\text{BE}}$(3)
I0a
I0b
I1a
I1b
I2a
I2b
I3a
I3b
FCT157 or FCT257
MULTI-PLEXER
SEL
$\overline{\text{WrEn}}$(3:0)
$\overline{\text{WrEn}}$
$\overline{\text{WrEn}}$
2881 drw 18

Figure 16. Gating Byte Enables into the Write Enables

SUMMARY

The main memory interface of the R3051 is conventional and simple. Basic blocks include address de-multiplexing, address decoding, data transceivers, wait-state controller, as well as the memory and I/O modules themselves. The R3051's uses both edges of the clock for control signals to reduce inter-cycle latency. Thus conventional wait-state controller algorithms can be used if the following guidelines are followed:

1: In $\overline{\text{SysClk}}$ based wait-state controllers, the input clock should be unbuffered from the processor's $\overline{\text{SysClk}}$ output. $\overline{\text{SysClk}}$ controllers will have a minimum of 1 or more wait-states. $\overline{\text{SysClk}}$ registers require small hold time and a minimum clock to output propagation delay to meet the R3051 input hold time.

2: In SysClk (inverted version of processor $\overline{\text{SysClk}}$ output) based wait-state controllers, the master reference counter must be reset independently of the de-asserting edges of $\overline{\text{Rd}}$ or $\overline{\text{Wr}}$. This can be done by gating the memory type and cycle type into a $\overline{\text{CycleEnd}}$ output which deterministically resets the counter.

The R3051's integration of an instruction cache, a data cache, read buffers, and write buffers allows simple main memory interfacing which can be implemented using a small amount of external logic. Thus the R3051 reduces the cost and board size of RISC processing, while maintaining very high throughput.