Chapter 7

INPUT/OUTPUT ORGANIZATION

7.1 Otherwise, the same data could be read twice.

7.2 The device would appear at addresses 7CA4, 7DA4, 7EA4 and 7FA4.

7.3 First we need to filter the incoming interrupt requests by setting all except the highest-priority request to 0. This can be arranged by generating a filtered version, I_i, as follows:

$$\begin{aligned} I7 &= INTR7 \\ I6 &= INTR6 \cdot \overline{INTR7} \\ &\vdots \\ I1 &= INTR1 \cdot \overline{(INTR7 + INTR6 + INTR5 + INTR4 + INTR3 + INTR2)} \end{aligned}$$

A binary number $b_2b_1b_0$ representing the highest-priority interrupt can now be readily generated from the filtered requests, where

$$\begin{aligned} b_0 &= I1 + I3 + I5 + I7 \\ b_1 &= I2 + I3 + I6 + I7 \\ b_2 &= I4 + I5 + I6 + I7 \end{aligned}$$

7.4 *Figure 7.4*: The malfunction would not be recognized. The master that requested the data reads whatever happens to be on the data lines and uses that as data, resulting in an error. To prevent this error, the bus should include a signal such as the Salve-ready response in the protocols of Figures 7.5 and 7.6.

Figure 7.5: The master waits for the Salve-ready response for a preset period of time, then aborts the operation. Usually this results in an interrupt so that an interrupt-service routine can inform the user of the malfunction.

Figure 7.6: Same as in the case of Figure 7.5.

7.5 The address is needed only to select an I/O device register. Once the device recognizes the address of one of its registers during the first clock cycle, it can proceed with the rest of the bus transaction on its own, provided it keeps track of which register has been addressed. For example, the interface may include a flip-flop associated with each of its addressable registers to identify the register that has been addressed.

7.6 The duration of a data transfer operation increases automatically to accommodate the increased propagation delay. Each of the two parties exchanging data using the asynchronous protocol in Figure 7.6 waits for a signal to arrive from the other, so data are transferred correctly. In the case of Figure 7.4, the duration of the clock cycle must accommodate the longest possible propagation delay. A longer distance requires a longer clock cycle; otherwise errors could occur.

7.7 The clock period must accommodate the worst case delays introduced by the bus drivers, propagation on the bus, address decoding and setup times. Variability in the addressed device's response time is accommodated by using different numbers of clock cycles.

(*a*) Minimum time to send and decode address = 2 + 10 + 6 = 18 ns.
Minimum time to send data = 2 + 10 + 1.5 = 13.5
Therefore the minimum period of a clock cycle is 18 ns.

(*b*) Assume that the minimum clock period of 18 ns is used. The addressed device begins to fetch the requested data at the beginning of the second clock cycle. If the requested data are available immediately, the time needed to send them is 13.5 ns, which is less than the clock period. Hence, the device can respond in the second clock cycle. A device that requires 25 ns to fetch the data will have the data available to send 7 ns into the third clock cycle. This does not leave sufficient time to send the data to the processor in the same cycle. Hence, it will respond in the fourth clock cycle.

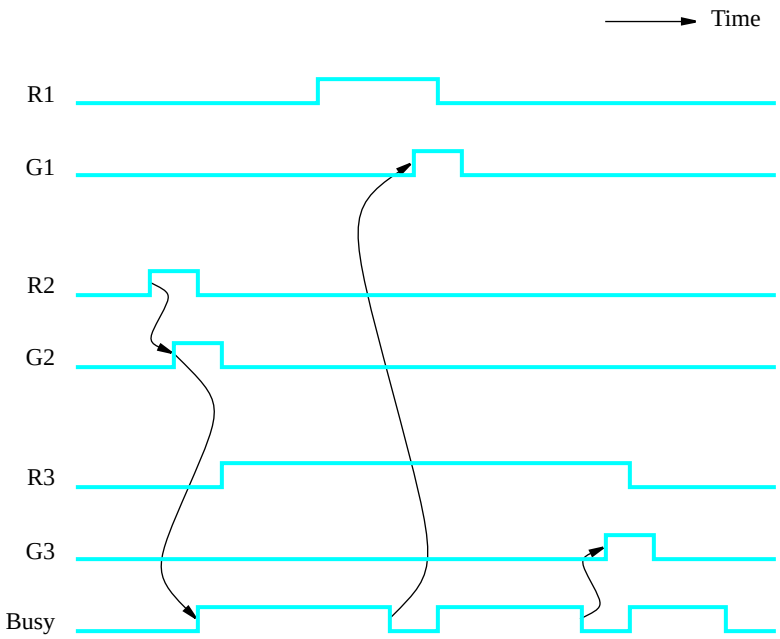

Figure PS7.1 Timing diagram for an arbiter that uses a Busy signal.

7.8 The addressed device sends the requested data at t_2. This is followed by three traversals of the bus to complete the handshake between Master-ready and Slave-ready, each taking one bus driver delay and one propagation delay. The bus cycle ends when the master sees a negated Slave-ready (t_5). Therefore:

Minimum bus cycle = 1+ 2 + 5 + 6 + 0 + 1 + 1.5 + 3(2 + 5) + 1 = 38.5 ns

Maximum bus cycle = 1+ 2 + 5 + 6 + 25 + 1 + 1.5 + 3(2 + 10) + 1 = 78.5 ns

Note that an allowance for bus skew is added when the address is transmitted and later when it is removed, because the address must always be correct when Master-ready is asserted. It is also added when the slave places its data on the data lines.

7.9 This protocol uses a partial handshake in which the master sends a pulse on Master-ready, to which the slave responds with a pulse on Slave-ready. The cycle ends when the falling edge of the Slave-ready pulse reaches the master.

Minimum bus cycle = 1 + 2+ 5 + 6 + 0 + 1 + 1.5 + 2 + 5 + 4 = 27.5 ns

Maximum bus cycle = 1+ 2 + 5 + 6 + 25 + 1 + 1.5 + 2 + 10 + 4 = 57.5 ns

7.10 A timing diagram for a protocol that uses a Busy signal is shown in Figure PS7.1.

7.11 The main change in this protocol is that the arbiter sends of a grant only when Busy is inactive. It drops the grant when it sees Busy asserted. The revised state diagram is given in Figure PS7.2.

7.12 (*a*) A possible revision to the protocol would work as follows. When the arbiter receives a request from a high-priority device while a lower-priority device is being serviced, it drops the grant. This indicates to the device that is currently using the common resource that it must suspend its operation. It does so at the earliest, but safe opportunity, saving any information that may be needed when it resumes operation later. Then it drops its request. The arbiter can then grant the use of the resource to the higher-priority device. Meanwhile, the lower priority device asserts its request again, so that it can complete its suspended operation when the resource becomes available.

(*b*) A state diagram is given in Figure PS7.3.

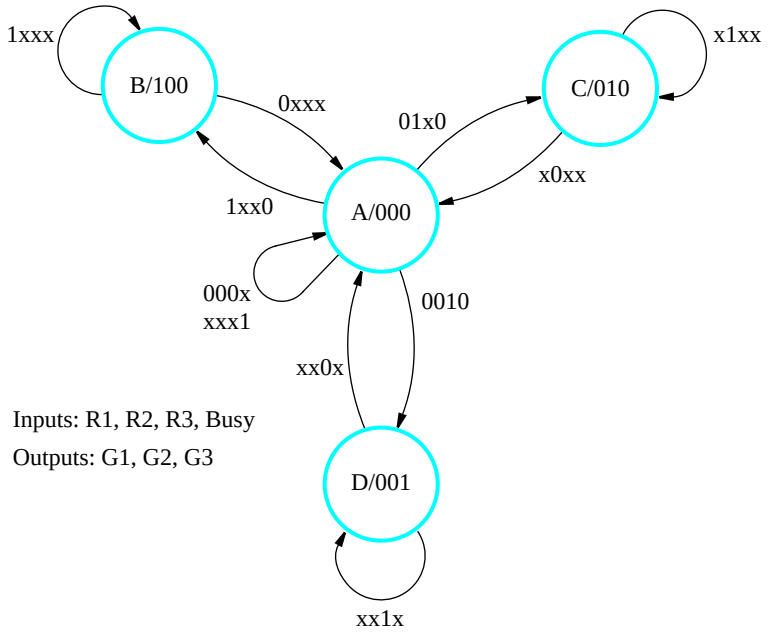

Figure PS7.2 State diagram for an arbiter that uses a Busy signal.

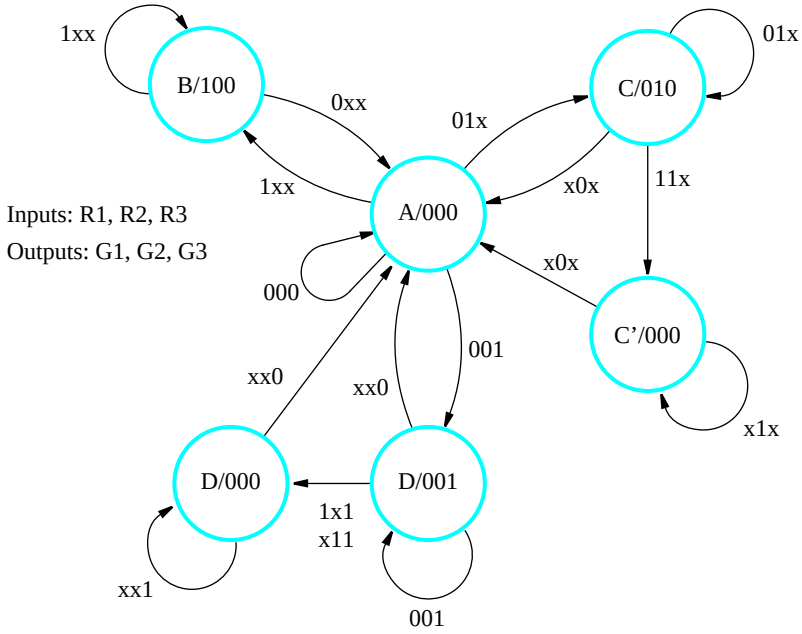

Figure PS7.3 An arbiter that allows preemption.

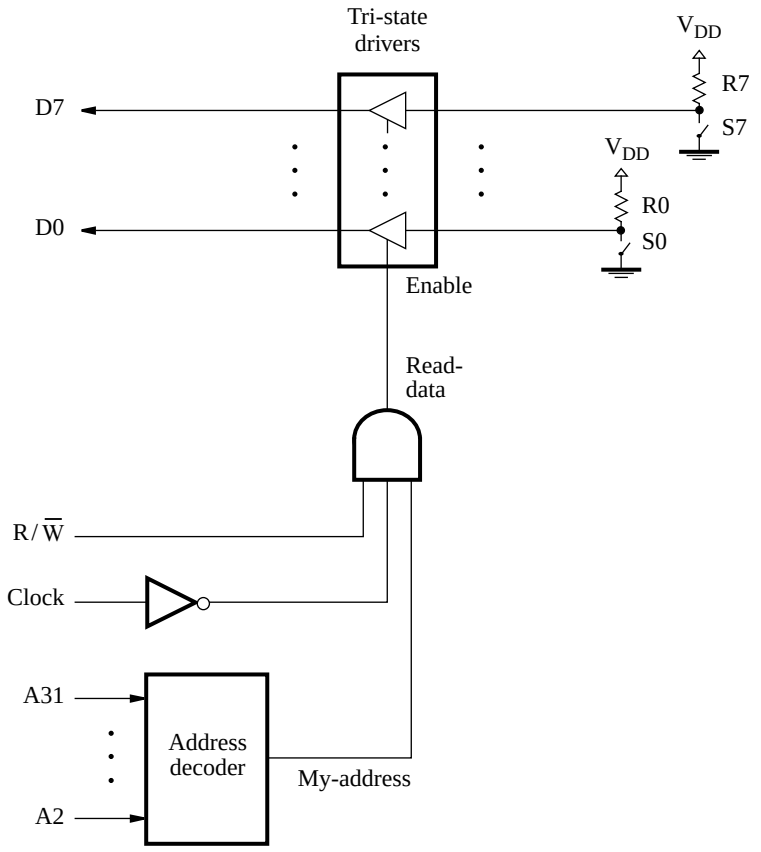

Figure PS7.4 Interface circuit for 8 sensors.

7.13 After R3 is serviced, the priority order is R4, R1, R2. Service will proceed in this order, regardless of the time of arrival of the requests.

7.14 If a device sends a second request after being served, the rotating-priority regime ensures that any outstanding requests are served before that device is served a second time. The fixed-priority regime, on the other hand, allows a high-priority device to be served multiple times before other requests are served, thus leaving open the possibility that some lower-priority device may be denied service for a long period of time.

7.15

$$\begin{aligned}\text{My-address} = \quad & A_{15} \cdot A_{14} \cdot A_{13} \cdot A_{12} \cdot \\ & A_{11} \cdot \overline{A}_{10} \cdot A_9 \cdot \overline{A}_8 \cdot \\ & \overline{A}_7 \cdot A_6 \cdot A_5 \cdot \overline{A}_4 \cdot \\ & \overline{A}_3 \cdot A_2 \cdot \overline{A}_1 \cdot \overline{A}_0\end{aligned}$$

7.16 A possible interface circuit for connecting the sensors is shown in Figure PS7.4. The switch and resistor arrangement for each sensor produces a high or a low logic signal when the switch is open or closed, respectively. The resulting 8 bits of data are placed on the processor's bus data lines D0 to D7 using tri-state gates. In accordance with the bus protocol of Figure 7.4, the tri-state gates are enabled during the low phase of the clock cycle.

7.17 (*a*) The processor does not load the input data into its input register until the end of the clock cycle. It makes no difference if the device sends its data early. The data will still be available at the end of the cycle and the processor will receive them correctly.

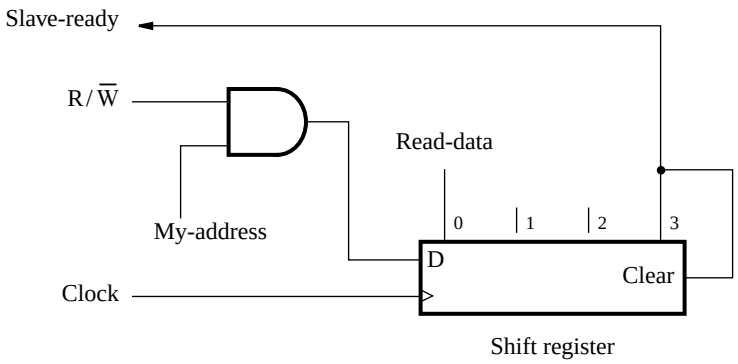

Figure PS7.5 Timing signal generation.

(*b*) The output of an address decoder may change several times before it settles to the final correct value. Before reaching this final value, it may erroneously indicate that a device is being addressed. Requiring devices to wait until the second phase of the clock before taking any action ensures that only the device being addressed will respond. Otherwise, a device may take erroneous action, such as reseting its status bit. Also, The outputs of all I/O devices are connected together via the bus lines. If two or more devices enable their tri-state drivers at the same time, large electrical currents could flow, thus consuming power unnecessarily and possibly causing damage to the gates.

7.18 A shift register is a convenient structure to generate the timing signals for this interface, as shown in Figure PS7.5. The register contains zeros when the interface is idle. A Read command causes a 1 to be shifted into bit 0 of the shift register, initiating the Read operation. Shifting continues until cycle 4, in which bit 3 becomes equal to 1 and Slave-ready is asserted. At the end of that cycle, the shift register is cleared, returning the interface to the idle state. (We have assumed that the shift register has a synchronous Clear input. When this input is activated, the counter is cleared on the next active edge of the clock.)

7.19 The DEVSEL# signal is a response from the target device to indicate to the initiator that the device has recognized its address and is ready to participate in a data transfer operation. However, data are not transmitted in every clock cycle. The TRDY# signal is asserted only during clock cycles in which data transfer actually takes place.

7.20 The target uses TRDY# to identify clock cycles in which it sends data to the initiator. It inserts the two cycles of delay needed by deactivating TRDY# in clock cycle 5 of Figure 7.19 and asserting it again in clock cycle 7.

7.21 The timing diagram for an output operation is similar to that in Figure 7.19, with one important difference. Clock cycle 2 in Figure 7.19 is used to turn the AD lines around. The initiator turns off its bus drivers, and the target turns on its bus drivers. This action is not needed in the case of an output operation, because both the address and the data are sent by the initiator. Hence, the initiator may start sending data in cycle 2, as shown in Figure PS7.6.

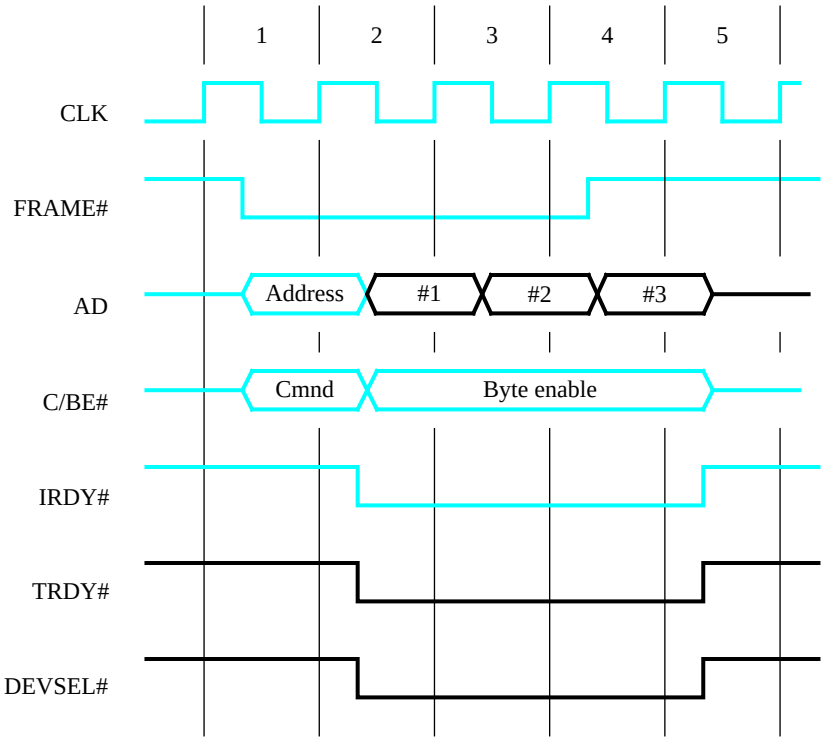

Figure PS7.6 An output operation on a PCI bus.