Chapter 7 problems 7.4, 7.10, 7.16, 7.18.

Matt Krueger

7.4 [M] Figures 7.4, 7.5, and 7.6 show three protocols for transferring data between a master and a slave. What happens in each case if the addressed device does not respond due to a malfunction during a Read operation? What problems would this cause and what remedies are possible?

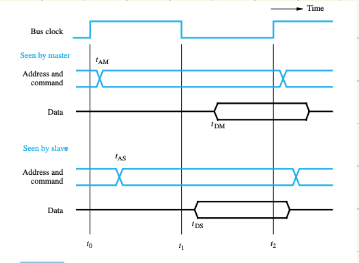


Figure 7.4 A detailed timing diagram for the input transfer of Figure 7.3.

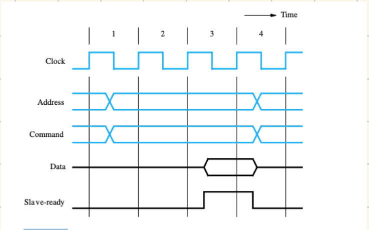


Figure 7.5 An input transfer using multiple clock cycles.

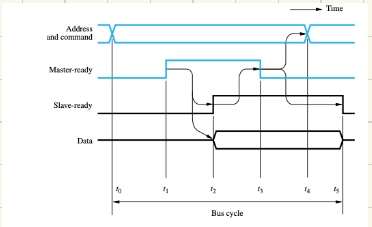


Figure 7.6 Handshake control of data transfer during an input operation.

Protocol	Problem	Solution(s)
Synchronous Bus (7.4) master / slave synchro on same bus clock cycle must be long enough to tolerate worst case delays	Data read is assumed to be valid. If slave didn't respond due to malfunction, stale/bad data is latched to masters register. BAD DATA LATCHED!	1. Add error detection circuitry such as a bus timeout to detect slave malfunction 2. Add status (ACK/NACK) flags or exception flags to make system robust
Multiple cycle (7.5) uses 'slave ready' signal extending function of synchronous bus to multiple cycles slave ready asserted. This solves issue of synchronous bus worst case	If slave never asserts signal then master may indefinitely be listening for a call that will not come. This wastes cpu cycles & reduces performance. No Action; Performance reduction	1. Add timeout same reasoning ↑ 2. watchdog timer for resetting slave
Asynchronous Bus (7.6) no clock; only a hand shake immune to skew, propagation delays & device latency	Master sends request by raising "master ready" but "slave ready" never comes. This is like previous cases where there is a bus lock/thread starvation waiting... performance reduction	1. as you probably guessed: Another timeout signal. if "slave ready" not received then drop "master ready". This can be done via an exception

Chapter 7 problems 7.4, 7.10, 7.16, 7.18.

Matt Kruger

7.10 [M] In the arbiter protocol example depicted in Figure 7.9, the master that receives a bus grant maintains its request line in the asserted state until it is ready to relinquish bus mastership. Assume that a common line called Busy is available, which is asserted by the master that is currently using the bus. The arbiter grants the bus only when Busy is inactive. Once a master receives a grant, it asserts Busy and drops its request, and in response the arbiter drops the grant. The master deactivates Busy when it is finished using the bus. Draw a timing diagram equivalent to Figure 7.9 for this mode of operation.

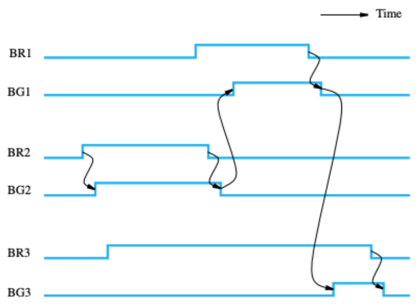


Figure 7.9 Granting use of the bus based on priorities.

Arbitration protocol:

- Busy line is added & asserted by current master
- arbiter grants only when not busy (Busy:0)

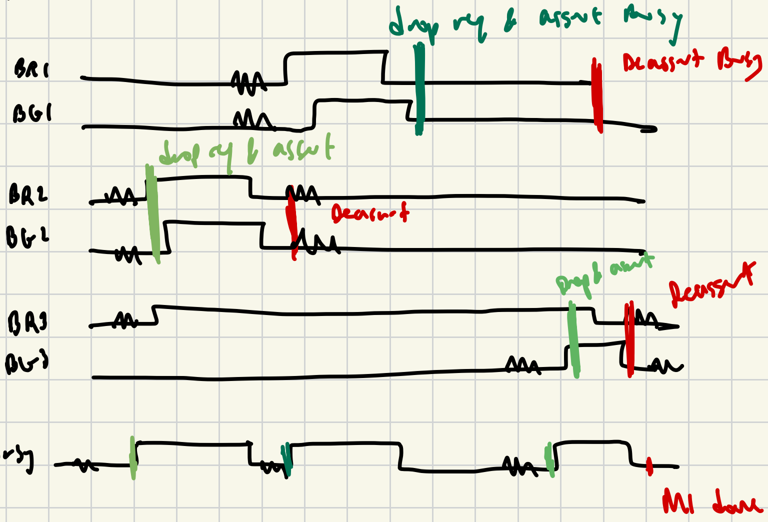
if master:

1. Assert Busy
2. Drop bus request
3. uses bus
4. De-asserts after done

Priority & not round Robin
↓
introduces starvation risk

BRx: Bus Grant request from Master x
BGx: Bus Granted to Master x
Busy: bus is in use

priority: M1 > M2 > M3



** ∿ used as this is not accurate depiction of data length. **

Chapter 7 problems 7.4, 7.10, 7.16, 7.18.

[illegible]

7.16 [M] An industrial plant uses several sensors to monitor temperature, pressure, and other factors. Each sensor includes a switch that moves to the ON position when the corresponding parameter exceeds a preset limit. Eight such sensors need to be connected to the bus of a 16-bit computer. Design an appropriate interface to enable the state of all eight switches to be read simultaneously as a single byte. Assume the bus is synchronous and that it uses the timing sequence of Figure 7.4.

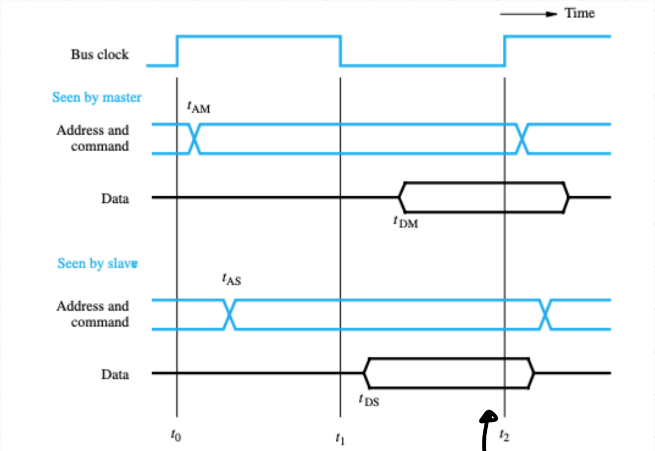


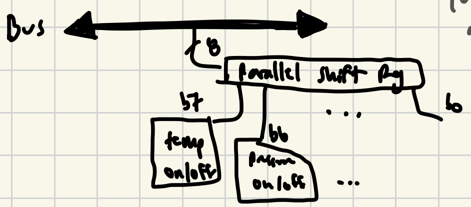
Figure 7.4 A detailed timing diagram for the input transfer of Figure 7.3.

All Flags 1 bit. A 16 bit message allows for 16 unique devices tracked for the industrial plant status interface

Data (2 bytes): [temp][pressure]...[x]
b16 b15 b0

As there are only 8 connected to the bus $\rightarrow$ only use 8 bits and have all others 0.

To decode & read simultaneously, a shift register can be used to output parallel to each connected device. This is a latch that holds each state during simultaneous output.



Question "simultaneous" constraint & bus requires a parallel solution?

sequential queries by assigning addresses & days something similar to I^2C

Chapter 7 problems 7.4, 7.10, 7.16, 7.18.

Matt [illegible]

7.18 [M] Data are stored in a small memory in an input interface connected to a synchronous bus that uses the protocol of Figure 7.5. Read and Write operations on the bus are indicated by a Command line called R/$\overline{W}$. The speed of the memory is such that two clock cycles are required to read data from the memory. Design a circuit to generate the Slave-ready response of this interface.

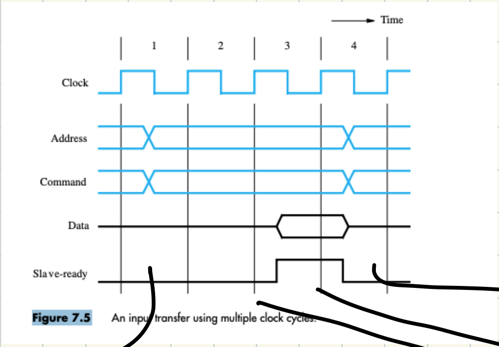


Figure 7.5 An input transfer using multiple clock cycles.

cycle

1: CPU sends address, command

2: slave accesses small memory connected

3: Data valid; "slave ready" asserted

4: Master reads data; slave ready deasserted

circuit design:

- clock to synchronize
- address decoder / command decoder to detect valid read access to this slave
- 2 bit counter to track cycles after read begins
- "slave ready" ff to store

cycle 1:
- R/$\overline{W}$ = 1, if address matches → decoder asserts read request; counter started

cycle 2:
- counter increments to 1

cycle 3:
- counter increments to 2
- assert "slave ready", data is valid on bus

cycle 4:
- "slave ready" deasserted
- counter reset.