COMMUNICATION and INPUT/OUTPUT:

Interacting Modules

COMMUNICATION and INPUT/OUTPUT:

Lecture Notes # 11

CAD Based Logic Design

ECE 368
Why do modules need to communicate?

1. To transfer data and correctly signal the presence of data
2. To access shared resources (e.g., buses, memory units) safely and correctly in a mutually exclusive manner

Methods of inter-module communication/synchronization:

1. Using semaphores, which are shared storage locations in which the modules write various values to signal various events. Some form of mutual exclusion is required on the access of these storage locations (generally, registers or main-memory) by different modules. A location needs to be "locked" by a module before writing to it, and then "unlocked" or "released" after writing.

2. Using direct signaling. We will concentrate on this in this class.

2. Using direct signaling. We will concentrate on this in this class.
A system is said to be synchronous when:

1. Each event in it happens only as a result of a clock transition after a fixed number of clock's from a start clock. Example: A simple non-pipelined CPU.
2. The propagation delay of any signal is < 1 clock period.

Synchronous, Asynchronous and Unsynchronized Systems

Interacting Modules (contd.)
Synchronous, Asynchronous and Unsynchronized Systems (contd.)

Asynchronous: A system is asynchronous if:

1. All events in it take place as a result of a clock transition and
   and

   2. The propagation delay of any signal > 1 clock period.

   There is no clock synchronization if the CPU always waits for these
   propagation can take place synchronously if the CPU always waits for these
   notes: The max. cc's needed for any multiplication is known, and multi-
   cycle.s
   Example: S& A or Booth's multiplication: The number of add/subtracts
   # of clock cycles (cc's) needed.

Asynchronous: A system is asynchronous if:

Synchronous, Asynchronous and Unsynchronized Systems (contd.)
A system is unsynchronized if any 2 modules in it are unsynchronized.

- (a) One is a clock-mode system and the other is not. Example: A carry-completion adder, which detects when the carry has fully propagated, and a slower clock than the CPU.
  (b) They run on different clocks that have no relationship to each other. Example: Dynamic memory unit and the CPU. The memory unit generally has a slower clock than the CPU.

Asynchronous, Asynchronous, and Unsynchronized Systems (contd.)
NOTE: In the following VHDL code snippets for various types of signaling,
for reliable simulation, replace all statements of the type:
wait until signal name = value
with:
if signal name \= value then
  wait until signal name = value
end if;

Signaling Mechanisms
Signaling Mechanisms

Unresponsive Signaling:

Producer Consumer

\[
\text{wait until clock's event and clock = 1;}
\]

\[
\text{prod state := next state;}
\]

\[
\text{case prod state is}
\]

\[
\text{case cons state is}
\]

\[
\text{next state = next state;}
\]

\[
\text{next state = next state;}
\]

\[
\text{wait until clock's event and clock = 1;}
\]

Additionally, it is required that both modules should be waiting for the other's signal prior to the signaling itself.

Will only work in a synchronous or asynchronous system, i.e., if the producer and consumer modules are synchronous or asynchronous w.r.t. each other.

Additionally, it is required that both modules should be waiting for the other's signal prior to the signaling itself.

Will only work in a synchronous or asynchronous system, i.e., if the producer and consumer modules are synchronous or asynchronous w.r.t. each other.
Partially Responsive Signaling:
(a) Needed in asynchronous systems when the two modules will not necessarily be waiting for each other's signals. In other words, it cannot be predetermined when either module will send a signal. Thus, after the initiating module sends a signal, it should wait for the receiving module's acknowledgment before lowering the signal. Another requirement is that both modules be busy waiting for the signal's acknowledgment.

(b) Another requirement is that the initiating module busy waits for the responding module's signal. This is because the initiating module's acknowledgment signal does not arrive until the responding module has processed the signal. Therefore, the initiating module should not lower its signal until it receives the responding module's acknowledgment.

(2) Partially Responsive Signaling: (a) Needed in asynchronous systems when the two modules will not necessarily be waiting for each other's signals. In other words, it cannot be predetermined when either module will send a signal. Thus, after the initiating module sends a signal, it should wait for the receiving module's acknowledgment before lowering the signal. Another requirement is that both modules be busy waiting for the signal's acknowledgment.

(b) Another requirement is that the initiating module busy waits for the responding module's signal. This is because the initiating module's acknowledgment signal does not arrive until the responding module has processed the signal. Therefore, the initiating module should not lower its signal until it receives the responding module's acknowledgment.
Partially Responsive Signaling (contd.)

In the previous example, the producer can also initiate signaling. Alternatively, the consumer can also initiate signaling. Initially, the consumer can also initiate signaling.

(2) Partially Responsive Signaling (contd.): In the previous example, the
Signaling (that mean completion of the round).

Thus not only is it necessary to signal and detect the start of a round, but also signal and detect the end of the current round, before starting the next round. Essentially, the modules need to detect the lowering of each other's signals (that mean completion of the round). If partially responsive signaling is used, as in the previous example, it is possible that the producer enter its next round of data transfer before the consumer is finished with the previous round. Thus the producer will cause a "stale" high on the ready line (high on it from the prev. round), and the consumer will detect this high and enter its next round of data transfer. This is necessary in unsynchronized systems.

(3) Fully Responsive/Interlocked Signaling

Data ready

Clock

50 cc

75 cc
(3) Fully Responsive/Interlocked Signaling (contd.):

Producer

\[ \text{\textbf{Data Reg}} := \text{New Data}; \]

\[ \text{if ready} = \text{`0'} \text{then} \]
\[ \text{wait until ready = `0'}; \]
\[ \text{wait until clock'event and clock = `0'}; \]
\[ \text{prod state} := s9 \]

\[ \text{if data ready} = \text{`0'} \text{then} \]
\[ \text{wait until data ready = `0'}; \]
\[ \text{wait until clock'event and clock = `0'}; \]
\[ \text{cons state} := s7 \]

\[ \text{when s9 =} \]
\[ \text{data ready = `0'}; \]
\[ \text{wait until ready = `1'}; \]
\[ \text{prod state} := s10 \]

\[ \text{when s8 =} \]
\[ \text{clock'event and clock = `0'}; \]
\[ \text{cons state} := s9 \]

\[ \text{when s10 =} \]
\[ \text{data ready = `1'}; \]
\[ \text{DATA = Data Reg;} \]
\[ \text{wait until data ready = `1'}; \]
\[ \text{wait until clock'event and clock = `0'}; \]
\[ \text{prod state} := s11 \]

\[ \text{when s11 =} \]
\[ \text{clock'event and clock = `0'}; \]

\[ \text{Producer} \]

\[ \text{Consumer} \]