Asynchronous Design

Peter A. Beerel
University of Southern California

Asynchronous Circuits

- Functional blocks communicating via handshaking on channels
  - No global clock
- An Example: Sequential Channel Decoder (Fano)
Asynchronous Channels

- Channel: A bundle of wires and a protocol for communicating data/control called a token
  - Data encoding: How to encode control/data: number of bits and semantics of bits
  - Communication protocol: Specific form of handshaking over request and acknowledgement wires

---

Single-rail encoding

- Features
  - 1 wire per bit
  - Request indicates validity of data

- Advantages
  - Same as synchronous
  - Minimal # of wires

- Disadvantages
  - Requires timing assumptions
Dual-rail Encoding

- **Dual-rail (1-of-2)**
  - 2 wires per bit
  - Generalizes to 1-of-N

- **Advantages**
  - Request encoded in data
  - Less timing assumptions

- **Disadvantages**
  - More wires

- Generalizes to 1-of-N

---

Four-phase Protocol

- Two-phases “active” communication
- Two-phases “Return to zero phases”
Two-phase Protocol

- Request and acknowledge represented by a single transition

![Diagram showing two-phase protocol]

Two-Phase Protocol (Con’t)

- Communication interpreted through transition on signal, not level of signal
- Request can be initiated by request signal going from 0 to 1 or 1 to 0
- No recovery period
- Often called *Transition Signalling*
Asynchronous Functional Blocks

- **Functionality**
  - Read a subset of input channels
    - Waits for input channels to have data tokens
  - Compute F and write to a subset of output channels
    - Waits for output channels to be reset
    - Acknowledges input channels
  - Resets output channels
    - Upon being acknowledged

Function Block Example: One-bit Adder

- **Functionality**
  - Wait for new tokens on A and B and Cin
  - Wait for Cout, Sum to have reset;
  - Compute and Send Cout = Maj(A,B,Cin)
  - Compute and Send Sum = XOR(A,B,Cin);
  - Repeat
Function Block Example: The Split

- **Functionality**
  - Wait for tokens on In and C and for Out0 and Out1 to be reset
  - Tokens on In routed to Out0 or Out1 depending on value of token on C
    - $C = 1$: Next data token on In routed to Out0
    - $C = 0$: Next data token on In routed to Out1
  - Yields architectures with choice

Achieving High Speed: Pipelining

- **Coarse-Grain Pipeline**
  - **The Basic Idea**
    - F1 can reset and re-compute as soon as F2 is done with data
  - **Performance: cycle time**
    - Often involves delay through multiple blocks due to handshaking
    - Often involves completion detection on channel data
  - **Comparison w/ comparably-grained synchronous pipeline**
    - Slow!
Typical Asynchronous Pipeline

- PS0 pipeline [Williams 91]
  - Functional Units: Dual-rail dynamic logic
  - Channels: Dual-rail w/ no explicit request wire

<table>
<thead>
<tr>
<th>Data(^1)</th>
<th>Data(^2)</th>
<th>Logical Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Reset</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Invalid</td>
</tr>
</tbody>
</table>

PS0 Handshaking Operation

- Precharge stage\(_N\): When stage\(_{N+1}\) completes evaluation
- Evaluate stage\(_N\): When stage\(_{N+1}\) completes precharging