Asynchronous Circuit Design

Chris J. Myers
Lecture 1: Introduction
Preface and Chapter 1

Synchronous Systems

- All events are synchronized to a single global clock.

Synchronous Advantages

- Simple way to implement sequencing.
- Widely taught and understood.
- Available components.
- Simple way to deal with noise and hazards.

Synchronous Disadvantages

- Clock distribution is difficult due to clock skew.
- Worst-case design.
- Sensitive to variations in physical parameters.
- Not modular.
- Power consumption.

Asynchronous Systems

- Synchronization is achieved without a global clock.

Asynchronous Advantages

- Elimination of clock distribution problems.
- Average-case performance.
- Adaptivity to processing and environmental variations.
- Component modularity.
- Lower system power requirements.
Asynchronous Challenges

- Lack of mature computer-aided design tools.
- Large area overhead for the removal of hazards.
- Average-case delay can be large.
- Lack of designer experience.

Asynchronous Circuit History

- Every design method traces its roots to one of two individuals:
  - Huffman - fundamental-mode circuits.
  - Muller - speed-independent circuits.

Key Asynchronous Circuit Designs

- ILLIAC (1952) and ILLAC2 (1962) - U. of Illinois
- Atlas (1962) and MU-5 (1966) - U. of Manchester
- Macromodules (60s-70s) - Washington U., St. Louis
- First commercial graphics system (70s) - Evans & Sutherland
- DDM dataflow computer (1978) - U. of Utah
- First asynchronous microprocessor (1989) - Caltech
- First code-compatible processor (1994) - U. of Manchester
- Commercial pager (80s) - Phillips
- RAPPID (1995-9) - Intel

Asynchronous Startups

- Handshake Solutions - Microcontrollers (Phillips)
- Fulcrum - Ethernet Switches (Caltech)
- Silistix - Self-timed interconnect (U. of Manchester)
- Achronix Semiconductor - Asynchronous FPGAs (Cornell)

Wine Shop Problem Specification

- Small winery and wine shop in Southern Utah.
- Only a single wine patron.
- Wine shop only has a single small shelf.
- Synchronous versus asynchronous wine shopping.

Channels of Communication

- Winery
- WineryShop
- Shop
- ShopPatron
- Patron
Channels of Communication in VHDL

Winery: process
begin
    send(WineryShop, bottle);
end process;

Shop: process
begin
    receive(WineryShop, shelf);
    send(ShopPatron, shelf);
end process;

Patron: process
begin
    receive(ShopPatron, bag);
end process;

Event Protocol

Shop: process
begin
    req_wine; - call winery
    ack_wine; - wine arrives
    req_patron; - call patron
    ack_patron; - patron buys wine
end process;

Signal Protocol

Shop: process
begin
    assign(req_wine,'1'); - call winery
    guard(ack_wine,'1'); - wine arrives
    assign(req_patron,'1'); - call patron
    guard(ack_patron,'1'); - patron buys wine
end process;

2-Phase Protocol

Shop_2Phase: process
begin
    assign(req_wine,'1'); - call winery
    guard(ack_wine,'1'); - wine arrives
    assign(req_wine,'0'); - reset req_wine
    guard(ack_wine,'0'); - ack_wine resets
    assign(req_patron,'1'); - call patron
    guard(ack_patron,'1'); - patron buys wine
    assign(req_patron,'0'); - reset req_patron
    guard(ack_patron,'0'); - ack_patron resets
end process;

4-Phase Protocol: Active/Active

Shop_4Phase: process
begin
    assign(req_wine,'1'); - call winery
    guard(ack_wine,'1'); - wine arrives
    assign(req_wine,'0'); - reset req_wine
    guard(ack_wine,'0'); - ack_wine resets
    assign(req_patron,'1'); - call patron
    guard(ack_patron,'1'); - patron buys wine
    assign(req_patron,'0'); - reset req_patron
    guard(ack_patron,'0'); - ack_patron resets
end process;
Waveform for 4-Phase Protocol

req_wine

ack_wine

req_patron

ack_patron

4-Phase Protocol: Passive/Active

Shop_PA: process
begin
  guard(req_wine,'1'); - winery calls
  assign(ack_wine,'1'); - wine is received
  guard(req_wine,'0'); - req_wine resets
  assign(ack_wine,'0'); - reset ack_wine
  guard(ack_patron,'1'); - patron buys wine
  assign(req_patron,'1'); - call patron
  guard(ack_patron,'0'); - ack_patron resets
end process;

4-Phase Protocol: Passive/Passive

Active/Active Protocol

Shop_AA: process
begin
  assign(req_wine,'1'); - call winery
  guard(ack_wine,'1'); - wine arrives
  assign(req_wine,'0'); - reset req_wine
  guard(ack_wine,'0'); - ack_wine resets
  assign(req_patron,'1'); - call patron
  guard(ack_patron,'1'); - patron buys wine
  assign(req_patron,'0'); - reset req_patron
  guard(ack_patron,'0'); - ack_patron resets
end process;

Active/Active Reshuffled

Shop_AA_reshuffled: process
begin
  assign(req_wine,'1'); - call winery
  guard(ack_wine,'1'); - wine arrives
  assign(req_patron,'1'); - call patron
  guard(ack_patron,'1'); - patron buys wine
  assign(req_wine,'0'); - reset req_wine
  guard(ack_wine,'0'); - ack_wine resets
  assign(req_patron,'0'); - reset req_patron
  guard(ack_patron,'0'); - ack_patron resets
end process;

Active/Active Circuit

req_wine → ack_patron

ack_wine → req_patron
Shop_PA_reshuffled: process
begin
  guard(req_wine,'1'); - winery calls
  assign(ack_wine,'1'); - receives wine
  guard(ack_patron,'0'); - ack_patron resets
  assign(req_wine,'1'); - call patron
  guard(req_wine,'0'); - req_wine resets
  assign(req_patron,'1'); - call patron
  assign(ack_wine,'0'); - reset ack_wine
  guard(ack_patron,'0'); - reset req_patron
  assign(ack_wine,'1'); - receives wine
end process;

AFSM and Huffman Flow Table (A/A reshuffled)

<table>
<thead>
<tr>
<th>req_wine</th>
<th>ack_wine</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>0</td>
<td>1, 10</td>
</tr>
<tr>
<td>1</td>
<td>0, 10</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Petri-net (P/A reshuffled)
TEL Structure (P/A reshuffled)

A/A Reshuffled Circuit

P/A Reshuffled Circuit

Active/Active State Variable

A/A SV Circuit

AFSM and Huffman Flow Table (A/A SV)
Reduced AFSM and Huffman Flow Table (A/A SV)

```
<table>
<thead>
<tr>
<th>ack_wine / ack_patron</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
```

Huffman's A/A SV Circuit

- **Bounded gate and wire delay model.**
- Circuit does not need to be closed.
- Single-input change fundamental mode.
- One input changes $\rightarrow$ output changes $\rightarrow$ state changes.
- May need to add delay in feedback state variables.

```
Muller's Active/Active SV Circuit
```

```
<table>
<thead>
<tr>
<th>req_wine/ack_wine</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
```

```
Huffman's Assumptions
```

```
Huffman's A/A SV Circuit
```

```
Huffman's A/A SV Circuit
```

```
Muller's Active/Active SV Circuit
```

```
Muller's Active/Active SV Circuit
```

```
Muller's Active/Active SV Circuit
```

```
Chris J. Myers (Lecture 1: Introduction) Asynchronous Circuit Design 39 / 1
```

```
Chris J. Myers (Lecture 1: Introduction) Asynchronous Circuit Design 40 / 1
```

```
Chris J. Myers (Lecture 1: Introduction) Asynchronous Circuit Design 41 / 1
```

```
Chris J. Myers (Lecture 1: Introduction) Asynchronous Circuit Design 42 / 1
```
Muller's Assumptions

- Unbounded gate delay model.
- Wire delays are assumed to be negligible.
- Forks are assumed to be isochronic.
- Model called speed-independent.

Muller's Active/Active SV Circuit

```
Muller's Active/Active SV Circuit

req_wine, ack_wine+, x+, req_wine-, ack_wine-, req_patron+, ack_patron+
ack_patron change felt at both x and req_wine gates simultaneously due to
isochronic fork assumption.
```

Timed Wine Shop

```
Shop_AA_timed: process
begin
assign(req_wine,'1',0,1); - call winery
assign(req_patron,'1',0,1); - call patron
- wine arrives and patron arrives
  guard_and(ack_wine,'1',ack_patron,'1');
  assign(req_wine,'0',0,1);
  assign(req_patron,'0',0,1);
  - wait for ack_wine and ack_patron to reset
  guard_and(ack_wine,'0',ack_patron,'0');
end process;
```

Timed Winery and Patron

```
winery: process
begin
  guard(req_wine,'1'); - wine requested
  assign(ack_wine,'1',2,3); - deliver wine
  guard(req_wine,'0');
  assign(ack_wine,'0',2,3);
end process;
patron: process
begin
  guard(req_patron,'1'); - shop called
  assign(ack_patron,'1',5,inf); - buy wine
  guard(req_patron,'0');
  assign(ack_patron,'0',5,7);
end process;
```

TEL Structure for Timed Wine Shop Example

```
TEL Structure for Timed Wine Shop Example

req_wine+
[2,3]
[req_wine]
req_wine-
ack_wine+
ack_patron+
[5,7]
[req_wine]
[req_patron]
req_patron+
[0,1]
[ack_wine & ack_patron]
req_wine-
[0,1]
[~ack_wine & ~ack_patron]
req_patron-
```

State Graph for Timed Wine Shop Example

```
State Graph for Timed Wine Shop Example

State vector: (ack_wine, ack_patron, req_wine, req_patron)
```

Chris J. Myers (Lecture 1: Introduction) Asynchronous Circuit Design
**Karnaugh Maps for Timed Circuit**

<table>
<thead>
<tr>
<th>req_wine</th>
<th>req_patron</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01 11 10</td>
</tr>
<tr>
<td>00</td>
<td>01 0 0 0</td>
</tr>
<tr>
<td>01</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>11</td>
<td>1 1 0 1</td>
</tr>
<tr>
<td>10</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>

**Timed Circuit**

```
req_wine [0.1] --<-- ack_wine [5,inf; 5,7]
```

**Performance Analysis**

- **Cycle time** is the delay from when the patron gets one bottle of wine until he can get another.
- Assuming the timed circuit delays are uniformly distributed except that the patron is extremely unlikely to take more than 10 minutes, we obtain the following cycle times:
  - Muller and Huffman’s circuits (A/A SV) - 21.5 minutes
  - Original (A/A reshuffled) - 20.6 minutes
  - Timed circuit - 15.8 minutes

**Validation versus Verification**

- **Validation** is simulation of interesting situations.
- **Verification** is exhaustive checks of all possible situations.
  - Can check that circuit conforms to the specification.
  - Can check that protocol has certain properties.

**Sample Properties**

- The wine arrives before the patron:
  - Always(ack_patron → ack_wine)
- When the wine is requested, it eventually arrives:
  - req_wine ⇒ Eventually(ack_wine)

**Summary of Course Topics**

- Communication Channels
- Communication Protocols
- Graphical Representations
- Huffman Circuits
- Muller Circuits
- Timed Circuits
- Verification
- Applications