Clock and Synchronization I
Outline

1. Why synchronous?
2. Clock distribution network and skew
3. Multiple-clock system
4. Meta-stability and synchronization failure
5. Synchronizer
Outline

1. Why synchronous?
2. Clock distribution network and skew
3. Multiple-clock system
4. Meta-stability and synchronization failure
5. Synchronizer
Timing of a combinational digital system

• Steady state
  – Signal reaches a stable value
  – Modeled by Boolean algebra

• Transient period
  – Signal may fluctuate
  – No simple model

• Propagation delay: time to reach the steady state
Timing Hazards

• Hazards: the fluctuation occurring during the transient period
  – Static hazard: glitch when the signal should be stable
  – Dynamic hazard: a glitch in transition
• Due to the multiple converging paths of an output port
- E.g., static-hazard \( sh=ab'+bc; \ a=c=1 \)
• E.g., dynamic hazard \((a=c=d=1)\)
E.g., Hazard of circuit with closed feedback loop (async seq circuit)

<table>
<thead>
<tr>
<th></th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>q</td>
</tr>
<tr>
<td>1</td>
<td>d</td>
</tr>
</tbody>
</table>

(a) D latch

```vhdl
library ieee;
use ieee.std_logic_1164.all;
entity dlatch is
  port(
    c: in std_logic;
    d: in std_logic;
    q: out std_logic
  );
end dlatch;

architecture demo_arch of dlatch is
  signal q_latch: std_logic;
begin
  process (c, d, q_latch)
  begin
    if (c='1') then
      q_latch <= d;
    else
      q_latch <= q_latch;
    end if;
  end process;
  q <= q_latch;
end demo_arch;
```
Dealing with hazards

- In a small number of cases, additional logic can be added to eliminate race (and hazards).

(c) Revised Karnaugh map and schematic to eliminate hazards
• This is not feasible for synthesis
• What’s can go wrong:
  – During logic synthesis, the logic expressions will be rearranged and optimized.
  – During technology mapping, generic gates will be re-mapped
  – During placement & routing, wire delays may change
  – It is bad for testing verification
• Better way to handle hazards
  – Ignore glitches in the transient period and retrieve the data after the signal is stabilized

• In a sequential circuit
  – Use a clock signal to sample the signal and store the stable value in a register.
  – But register introduces new timing constraint (setup time and hold time)
• Synchronous system:
  – group registers into a single group and drive them with the same clock
  – Timing analysis for a single feedback loop
Synchronous circuit and EDA

- Synthesis: reduce to combinational circuit synthesis
- Timing analysis: involve only a single closed feedback loop (others reduce to combinational circuit analysis)
- Simulation: support “cycle-based simulation”
- Testing: can facilitate scan-chain
2. Clock distribution network and skew
Outline

1. Why synchronous?
2. Clock distribution network and skew
3. Multiple-clock system
4. Meta-stability and synchronization failure
5. Synchronizer
Clock distribution network

• Ideal clock: clock’s rising edges arrive at FFs at the same time
• Real implementation:
  – Driving capability of each cell is limited
  – Need a network of buffers to drive all FFs
  – In ASIC: done by clock synthesis (a step in physical synthesis)
  – In FPGA: pre-fabricated clock distribution network
• Block diagram

• Ideal H-routing
Clock skew

- Skew: time difference between two arriving clock edges
Timing analysis

- Setup time constraint (impact on max clock rate)
- Hold time constraint
\[ t_3 < t_4 \]
\[ t_3 = t_0 + T_{cq} + T_{next}(\text{max}) \]
\[ t_4 = t_5 - T_{setup} = (t_0 + T_c + T_{skew}) - T_{setup} \]
\[ T_{cq} + T_{next}(\text{max}) + T_{setup} - T_{skew} < T_c \]
\[ T_{c}(\text{min}) = T_{cq} + T_{next}(\text{max}) + T_{setup} - T_{skew} \]

- Clock skew actually helps increasing clock rate in this particular case
• If the clock signal travels from the opposite direction

\[ T_{c(min)} = T_{cq} + T_{nnext(max)} + T_{setup} + T_{skew} \]

• Normally we have to consider the worst case since
  – No control on clock routing during synthesis
  – Multiple feedback paths
• Hold time constraint

\[
t_h < t_2
\]

\[
t_2 = t_0 + T_{cq} + T_{next(min)}
\]

\[
t_h = t_0 + T_{hold} + T_{skew}
\]

\[
T_{hold} < T_{ca} + T_{next(min)} - T_{skew}
\]

\[
T_{hold} < T_{cq} - T_{skew}
\]

• Skew may reduce hold time margin
• Hold time violation cannot be corrected in RT level
• Summary
  – Clock skew normally has negative impact on synchronous sequential circuit
  – Effect on setup time constraint: require to increase clock period (i.e., reduce clock rate)
  – Effect on hold time constraint: may introduce hold time violation
    • Can only be fixed during physical synthesis: re-route clock; re-place register and comb logic; add artificial delay logic
  – Skew within 10% of clock period tolerable
Outline

1. Why synchronous?
2. Clock distribution network and skew
3. Multiple-clock system
4. Meta-stability and synchronization failure
5. Synchronizer
Why multiple clocks

• Inherent multiple clock sources
  – E.g., external communication link

• Circuit size
  – Clock skew increases with the # FFs in a system
  – Current technology can support around 10k FFs

• Design complexity
  – E.g., as system w/ 16-bit 20 MHz processor, 1-bit 100 MHz serial interface, 1 MHz I/O controller

• Power consideration
  – Dynamic power proportional to switching freq
Derived vs Independent clocks

- Independent clocks:
  - Relationship between the clocks is unknown

- Derived clocks:
  - A clock is derived from another clock signal (e.g., different clock rate or phase)
  - Relationship is known
  - Logic for the derived clock should be separated from regular logic and manually synthesized (e.g., special delay line or PLL)
  - A system with derived clock can still be treated and analyzed as a synchronous system
GALS

• Globally asynchronous locally synchronous system
  – Partition a system into multiple clock domains
  – Design and verify subsystem in same clock domain as a synchronous system
  – Design special interface between clock domains
Outline

1. Why synchronous?
2. Clock distribution network and skew
3. Multiple-clock system
4. Meta-stability and synchronization failure
5. Synchronizer
Timing analysis of a synchronous system

• To satisfy setup time constraint:
  – Signal from the state register
    • Controlled by clock
    • Adjust clock period to avoid setup time violation
  – Signal from external input
    • Same if the external input comes from another synchronous subsystem
    • Otherwise, have to deal with the occurrence of setup time violation.
Metastability

- What happens after timing violation?

\[ T_{\text{setup}} + T_{\text{hold}} \]

\[ T_{\text{cq}} \]
– Output of FF becomes 1 (sampled old input value)
– Output of FF becomes 0 (sampled new input value)
– FF enters metastable state, the output exhibits an “in-between” value
  • FF eventually “resolves” to one of stable states
  • The resolution time is a random variable with distribution function (τ is decay constant)
    \[ P(T_r) = e^{-\frac{T_r}{\tau}} \]
  • The probability that metastability persists beyond Tr (i.e., cannot be resolved within Tr)
MTBF(Tr)

• Synchronization failure
  – an FF cannot resolve the metastable condition within the given time

• MTBF
  – Mean Time Between synchronization Failures
  – Basic criterion for metastability analysis
  – Frequently expressed as a function of Tr (resolution time provided)
• **MTBF computation**

- $R_{\text{meta}}$: The average rate at which an FF enters the metastable state.
- $P(T_r)$: The probability that an FF cannot resolve the metastable condition within $T_r$.

$$R_{\text{meta}} = w \times f_{\text{clk}} \times f_d$$

$w$ is the **susceptible time window**

$$P(T_r) = e^{-\frac{T_r}{\tau}}$$

$$AF(T_r) = R_{\text{meta}} \times P(T_r) = w \times f_{\text{clk}} \times f_d \times e^{-\frac{T_r}{\tau}}$$

$$\text{MTBF}(T_r) = \frac{1}{AF(T_r)} = \frac{e^{\frac{T_r}{\tau}}}{w \times f_{\text{clk}} \times f_d}$$
• E.g., $w=0.1\text{ns}$, $\tau=0.5\text{ns}$, $f_{\text{clk}}=50\text{MHz}$, $f_d=0.1f_{\text{clk}}$

<table>
<thead>
<tr>
<th>$T_r$</th>
<th>MTBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0 ns</td>
<td>$4.00 \times 10^{-05}$ sec (0.04 msec)</td>
</tr>
<tr>
<td>2.5 ns</td>
<td>$5.94 \times 10^{-03}$ sec (5.94 msec)</td>
</tr>
<tr>
<td>5.0 ns</td>
<td>$8.81 \times 10^{-01}$ sec (0.88 sec)</td>
</tr>
<tr>
<td>7.5 ns</td>
<td>$1.31 \times 10^{+02}$ sec (131 sec)</td>
</tr>
<tr>
<td>10.0 ns</td>
<td>$1.94 \times 10^{+04}$ sec (5.39 hours)</td>
</tr>
<tr>
<td>12.5 ns</td>
<td>$2.88 \times 10^{+06}$ sec (3.33 days)</td>
</tr>
<tr>
<td>15.0 ns</td>
<td>$4.27 \times 10^{+08}$ sec (1.36 years)</td>
</tr>
<tr>
<td>17.5 ns</td>
<td>$6.34 \times 10^{+10}$ sec (2.01 $\times$ $10^3$ years)</td>
</tr>
<tr>
<td>20.0 ns</td>
<td>$9.42 \times 10^{+12}$ sec (2.99 $\times$ $10^5$ years)</td>
</tr>
<tr>
<td>22.5 ns</td>
<td>$1.40 \times 10^{+15}$ sec (4.43 $\times$ $10^7$ years)</td>
</tr>
<tr>
<td>25.0 ns</td>
<td>$2.07 \times 10^{+17}$ sec (6.58 $\times$ $10^9$ years)</td>
</tr>
<tr>
<td>27.5 ns</td>
<td>$3.08 \times 10^{+19}$ sec (9.76 $\times$ $10^{11}$ years)</td>
</tr>
<tr>
<td>30.0 ns</td>
<td>$4.57 \times 10^{+21}$ sec (1.45 $\times$ $10^{14}$ years)</td>
</tr>
<tr>
<td>32.5 ns</td>
<td>$6.78 \times 10^{+23}$ sec (2.15 $\times$ $10^{16}$ years)</td>
</tr>
<tr>
<td>35.0 ns</td>
<td>$1.01 \times 10^{+26}$ sec (3.19 $\times$ $10^{18}$ years)</td>
</tr>
</tbody>
</table>
• Observations
  – MTBF is statistical average
  – Only Tr can be adjusted in practical design
  – MTBF is extremely sensitive to Tr
  • Good: synchronization failure can be easily avoided by providing additional resolution time
  • Bad: minor modification can introduce synchronization failure
Outline

1. Why synchronous?
2. Clock distribution network and skew
3. Multiple-clock system
4. Meta-stability and synchronization failure
5. Synchronizer
• Synchronization circuit:
  – Synchronize an asynchronous input with system clock
  – No physical circuit can prevent metastability
  – Synchronizer just provides enough time for the metastable condition to be “resolved”

• E.g.,
  – \( w = 0.1 \text{ns}, \ \tau = 0.5 \text{ns}, \ \frac{f_{\text{clk}}}{f_{\text{clk}}} = 50 \text{MHz}, \ \frac{f_{d}}{f_{\text{clk}}} = 0.1 \frac{f_{\text{clk}}}{f_{\text{clk}}}, \)
  – \( T_{\text{setup}} = 2.5 \text{ns} \)
(a) No synchronizer

(b) One-FF synchronizer

(c) Two-FF synchronizer

(d) Three-FF synchronizer
No synchronizer

- $T_r = 0$
- $MTBF(0) = 0.04$ ms
One-FF synchronizer

- $T_r = T_c - (T_{comb} + T_{setup})$
- $T_r$ depends on $T_c$, $T_{setup}$ and $T_{comb}$
  - $T_c$: vary with system specification
  - $T_{comb}$: vary with circuit, synthesis (gate delay), placement & routing (wire delay)
- E.g.,
  - $T_r = 20\text{ns} - (T_{comb} + 2.5\text{ns}) = 17.5\text{ns} - T_{comb}$
  - $T_{comb} = 1\text{ns}, T_r = 16.5\text{ns}; \text{MTBF}(16.5) = 272\text{yr}$
  - $T_{comb} = 12.5\text{ns}, T_r = 5\text{ns}; \text{MTBF}(5) = 0.88\text{s}$
- Not a reliable design
Two-FF synchronizer

• Add an extra FF to eliminate $T_{comb}$
  – $T_r = T_c - T_{setup}$
  – $T_r$ depends on $T_c$ only
  – Async input delayed by two clock cycles

• E.g.,
  – $T_r = 20 - 2.5 = 17.5$; MTBF(17.5) = 2000 yr

• Most commonly used synchronizer

• In ASIC technology
  – May have “metastability-hardened” D FF cell (large area)
Three-FF synchronizer

• Add an extra stage to increase resolution time
  – $T_r = 2(T_c - T_{\text{setup}})$
  – Async input delayed by three clock cycles

• E.g.,
  – $T_r = 2 \times (20 - 2.5); \quad \text{MTBF}(35) = 3 \times 10^{18} \text{ yr}$

• Hardly needed
Observation

• $T_r$ is in the exponent of MTBF equation
• Small variation in $T_r$ can lead to large swing in MTBF
Proper use of synchronizer

- Use a glitch-free signal for synchronization
- Synchronize a signal in a single place
- Avoid synchronization multiple “related” signals.
- Reanalyze the synchronizer after each design change
(a) Synchronizing a signal in two places

(b) Synchronizing a signal in one place
Why synchronization is a “tricky” issue

• Metastability is basically an “analog” phenomena
• Metastability behavior is described by random variable
• Metastability cannot be easily modeled or simulated at the gate level (only ‘X’)
• Metastability cannot be easily observed or measured in a physical circuit (e.g., MTBF = 3 months)
• MTBF is very sensitive to circuit revision
Summary

Synchronous methodology simplifies life
Clock distribution / clock skew
  Can tolerate around skew around $T_c/10$
  Must partition larger designs

Multiple-clock system
  Asynchronous signals $\Rightarrow$ metastability

Characterization of Metastability
  Probabilistic: $MTBF(T_r)$

Synchronizers
  Confine metastability, increase $T_r$
Synthesis Guidelines

Using clocks
- Don’t manipulate in RT level
- Minimize number of clocks
- Use derived clocks if possible (sep. from RT)

Synchronizers
- Synchronize in a single place
- Avoid synchronizing related signals
- Use glitch-free signal for sync.
- Reanalyze after changes (clock rate, devices)