Week #8 Topics

- Definition of Register and Counter
- Registers, Shift Registers
- Ripple Counters
- Synchronous Binary Counters
- BCD Counters

Resources

Chapter #7, Mano Sections
- 7.1 Registers and Load Enable
- 7.6 Shift Registers
- 7.6 Ripple Counters
- 7.6 Synchronous Binary Counters

Registers: Definition

Register – a set of flip-flops
- May include extensive logic to control state transition
- Registers also refer to fast memory for storing data in a computer

Counters: Definition

Counter
- Register that goes through sequence of states as it is clocked
- We designed a simple counter in the previous Lecture using T Flip Flops!

Simple Register (No External Gates)

- Functionality
  - Store D (D₀, D₁, D₂, D₃)
  - On pos-edge of Clock
- Clear signal normally high
  - Power-up reset
**Disabling the Clocking**

- The transfer of new info into a register is referred to as **loading** the register.
- Typically we **don't want** to load every clock.
- We **gate the clock!! (Disable the clock)!!**
- We try to avoid gating!! (Timing issues)

**Shift Registers**

- A shift register is a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop.
- It is a register capable of shifting its stored bits laterally in one or both directions.
- All flip-flops receive a common clock pulse, which activates the shift from each stage to the next.

**Clocking**

- The transfer of new info into a register is referred to as **loading** the register.
- Typically we **don't want** to load every clock.
- We **gate the clock!! (Disable the clock)!!**
- We try to avoid gating!! (Timing issues)

**Alternative**

- If load 'H', then D goes through
- Otherwise, Q is fed back  - Keep same value
- No clock gating

- Why add all this logic?
- Because D FF doesn't have "no change" behavior

**Simple 4-bit Shift Register**

- Every bit shifts to the right at every clock edge.
- New information will enter via $S_i$ and leave from $S_0$. 

**Shift Registers**

- A shift register is a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop.
- It is a register capable of shifting its stored bits laterally in one or both directions.
- All flip-flops receive a common clock pulse, which activates the shift from each stage to the next.
**Simple 4-Bit Shift Register: Diagram**

- Clocked in common
- Just serial in and serial out

![Diagram of a 4-bit shift register](image)

---

**VHDL for Shift Registers**

```vhdl
-- 4-bit Shift Register with Reset
library ieee;
use ieee.std_logic_1164.all;

entity srg_4_r is
  port (CLK, RESET, SI : in std_logic;
        Q                      : out std_logic_vector(3 downto 0);
        SO                    : out std_logic);
end srg_4_r;

architecture behavioral of srg_4_r is
  Signal shift : std_logic_vector (3 downto 0);
begin
  process (CLK, RESET)
  begin
    if (RESET = '1') then
      shift <= '0000';
    elsif (CLK'event and CLK = '1')) then
      shift<= shift(2 downto 0) & SI;
    end if;
  end process;

  Q <= shift;
  S0 <= shift(3);
end behavioral;
```

---

**Bidirectional Shift Register**

- Shift either way
- Now we have following possible inputs
  - Parallel load
  - Shift from left
  - Shift from right
  - Also "no change"

---

**Shift Register with Parallel Load**

- **Load any value** to the shift register
- The shift register can then **shift information** one bit at a time
Shift Registers (Summary)

- Capability to shift bits
  - In one or both directions
- Usage?
  - Part of standard CPU instruction set
  - Cheap Multiplication/Division
  - Serial communications

Serial Addition

Hardware Comparison

Serial vs. parallel adder
- One full adder vs. n adders
- Serial takes n clock cycles, parallel only one clock cycle

Serial Addition

Synchronous Counters

Serial Addition

Counters

- Counter is a register – but has states
- Also goes through sequence of states – counts – on clock or other pulses
- Examples:
  - Binary Counter
    - Counts through binary sequence
    - n bit counter counts from 0 to \( 2^n - 1 \)
  - BCD Counter
  - Any Sequence Counter
Synthesis Using T Flip Flops

- Design a counter that counts from “000” to “111” and then back to “000” again.
- Use T Flip-Flops

A Counter using T Flip Flops “Synchronous Counters”

Example: T Flip Flop Synthesis

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Flip Flop Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Q_1)</td>
<td>(Q_0)</td>
<td>(T)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</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>1</td>
</tr>
</tbody>
</table>

Asynchronous Counters

- Asynchronous counters are yet another type of counters where not all flip flops are driven by the global clock.
- Ripple counters are asynchronous counters that are easy to design.
Counters with Parallel Load

Counters employed in digital systems quite often require a parallel-load capability for transferring an initial binary number into the counter prior to the counter operation.

- When "load" is equal to 1, the input load control disables the count operation and causes a transfer of data from the four parallel inputs to the four outputs.
- The carry output C0 becomes a 1 if all flip-flops are equal to 1 while the count input is enabled.
- This feature is useful for expanding the counter to more stages

BCD Counters

BCD Counters can also be designed using individual flip-flops and gates
The "Binary Counter" with parallel load can be converted into a synchronous BCD counter (How?) by connecting an external AND gate to the load control (as shown in the Figure).

By connecting an external AND gate to the load control (as shown in the Figure).

**Arbitrary Counters**

- One more type of counter is useful
- Count an arbitrary sequence
  - Maybe you need a sequence of states

**VHDL for Counters**

```vhdl
-- 4 bit Binary Counter with Reset
library ieee; use ieee.std_logic_1164.all;
entity count_4_r is
port (CLK, RESET, EN : in std_logic;
   Q        : out std_logic_vector(3 downto 0);
   CO       : out std_logic);
end count_4_r;
architecture behavioral of count_4_r is
signal count : std_logic_vector(3 downto 0);
beg
process (CLK, RESET)
begin
if (RESET = '1') then
   count <= '0000';
elsif (CLK'event and CLK = '1') and (EN = '1') then
   count <= count + "0001";
end if;
end process;
q <= count;
C0 <= '1' when count = "1111" and EN = '1' else '0';
end behavioral;
```
Serial Transfer (8-bit Shift Register)

Could shift data in

(a) Block diagram

What's on wire at each clock?

Clocked 4 times

Example II

Table Showing Shift

**Table of Serial Transfer**

<table>
<thead>
<tr>
<th>Timing pulse</th>
<th>Shift Register A</th>
<th>Shift Register B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial value</td>
<td>1 0 1 0 0 0 1 0</td>
<td>0 0 1 0 1 1 0 0</td>
</tr>
<tr>
<td>After T1</td>
<td>0 1 0 1 0 0 1 0</td>
<td>1 1 0 0 0 1 0 1</td>
</tr>
<tr>
<td>After T2</td>
<td>0 0 1 0 1 1 0 0</td>
<td>0 0 0 0 1 0 1 1</td>
</tr>
<tr>
<td>After T3</td>
<td>0 0 0 1 0 1 1 0</td>
<td>0 0 0 0 1 0 1 1</td>
</tr>
<tr>
<td>After T4</td>
<td>0 0 0 0 1 0 1 1</td>
<td>0 0 0 0 1 0 1 1</td>
</tr>
</tbody>
</table>

**Example II .. Continue**