EE 330
Lecture 43
Digital Building Blocks
Digital Building Blocks

- Shift Registers
- Sequential Logic
- Shift Registers (stack)
- Array Logic
- Memory Arrays
Ring Oscillators

- Widely used for clock generation when frequency of clock is not critical
- Jitter is not all that good either
- For lower frequencies, follow Ring Oscillator with a Boolean Divide by N
Ring Oscillators

- Odd number of stages will oscillate (even will not oscillate)
- Waveform nearly a square wave if n (number of stages) is large
- Output will slightly imbalance ring and device sizes can be compensated if desired
- Usually use a prime number (e.g. 31)
- Number of stages usually less than 50 (follow by dividers)
- Frequency highly sensitive to process variations and temperature

\[ f_{\text{OSC}} \approx \frac{1}{nt_{\text{PROP}}} \]

- \( f_{\text{OSC}} \) is the oscillation frequency
- \( n \) is the number of stages
- \( t_{\text{PROP}} \) is the propagation delay of a single stage (all assumed identical)
Sequential Logic Elements

Latch

\[ \text{D is held on } Q \text{ when } C_{LK} \text{ is low} \]
\[ \bar{Q} \text{ may also be present} \]

Pulsed Latch

\[ \text{D is held on } Q \text{ when } C_{LK} \text{ is low} \]
\[ C_{LK} \text{ is a pulse} \]
\[ \bar{Q} \text{ may also be present} \]

Special case of Latch

Flip Flop

May have one or two inputs
Inputs transferred to output on \( C_{LK} \)
Four basic types
\( J-K, S-R, D, T \)
Any type can be obtained from other
with simple combinational logic ckts
Often edge triggered and Master-Slave
Most widely used sequential logic element
Timing Comparison of Sequential Logic Elements

- **Latch**
  - Input: D
  - Output: Q
  - Clock: $C_{LK}$

- **Pulsed Latch**
  - Input: D
  - Output: Q
  - Clock: $C_{LK}$

- **Flip Flop**
  - Input: X
  - Output: Q
  - Clock: $C_{LK}$

(edge triggered MS D flip flop)
Latch Flip-Flop Terminology

• **Transparent:**
  – Input to Latch Appears on the Latch Output immediately while in transparent state

• **Opaque**
  – Input to the Latch does not appear at the output while in the opaque state

• **Edge Triggered**
  – Input to latch at a clock transition determines when the input is transferred to the latch output

• **Master-Slave**
  – Two-stage cascaded structure where output from one serves as input to the second stage
Latches

- Very simple
- When $X_C$ is high, in transparent state (tracks input)
- Negative edge triggered

Limitations:
- $Q$ can get to only $V_{DD} - V_T$
- Leakage of charge will occur when $X_C$ is low
- Any loads placed on $Q$ will further degrade held signal
Latches

- Provides rail-rail output
- Modestly faster
- Other limitations
Latches

- Buffering eliminates loading
- Provides rail-rail output
- Provides signal inversion
- Second inverter will eliminate signal inversion
- Output transparent when $X_C$ is high

- $C$ is input capacitance to inverter
- Need not show load since buffered
Latches

- Becomes nonvolatile
- Provides complimentary output
- Output transparent when $X_C$ is high
- Contention on output when loading
  - (must size devices so latch will load)
Latches

- Becomes nonvolatile
- Provides complimentary output
- Output transparent when $X_C$ is high
- Contention on output removed
- 6-transistor Cell
- Actually the basic memory element used in many SRAM arrays
Pulsed Latch

- Pulse generator drives clock input of Latch
- Many different pulse generators can be used
Pulsed Latch

Pulse Generator

Simple Pulse Generator

timing diagram
Pulsed Latch

Pulse Generator

Simple Pulsed Latch

\( C_{LK} \)

\( \bar{C}_P \)

\( X_C \)

\( \bar{Q} \)

\( Q \)

\( Q \)
Pulsed Latch

Pulse Generator

Another Simple Pulsed Latch
Flip Flops

Four Basic Flip Flops

S-R Flip Flop

J-K Flip Flop

D Flip Flop

T Flip Flop
Flip Flops

Implementation of the S-R Flip Flops

- Many different flip flops exist
- Extremely high number of flip flops warrants design of a good structure
- Not clocked or Master Slave

8 transistor implementations
Flip Flops

Implementation of the S-R Flip Flops

Clocked S-R flip flop

20 transistor implementation
Output transparent when clock is H
Flip Flops

Implementation of the S-R Flip Flops

Clocked S-R flip flop

Clocking Edge-Triggered S-R Master Slave Flip Flop
Flip Flops

Implementation of the S-R Flip Flops

Clocked Edge-Triggered S-R Master Slave Flip Flop

Master-Slave Edge-triggered Clocked S-R Flip Flop

Large Device Count: 40 transistors
Flip Flops

D Flip Flop

Output transparent when in clock is high
Flip Flops

Master-Slave Edge-triggered D Flip Flop

Timing Diagram

- 12 transistors (but will work with 10)
- Many other simple D Flip-flops exist as well
Shift Registers

- Basic 1-bit dynamic shift register
- Data is stored on parasitic capacitor $C_P$
- $C_P$ is generally input capacitance to inverter and omitted from diagram
Shift Registers

Dynamic Shift Register

- 6 transistor cell
- Must be clocked to retain data

Timing Diagram:
Shift Registers

Dynamic Shift Register
6 transistor cell

Dynamic Shift Register with Static Hold
Shift Registers

Dynamic Shift Register

\[ \text{D} \rightarrow Q_1 \rightarrow Q \]

Simple redrawing
Shift Registers

Dynamic Shift Register

\[ \text{n}=\text{bit Dynamic Shift Register} \]

- FIFO Operation
- Layout so stages can be simply abutted
Shift Registers

Dynamic Shift Register

Bi-directional Dynamic Shift Register

If $T_L$ and $T_R$ replaced with $HCT_L$ and $HCT_L$, have static hold operation
Shift Registers

Dynamic Shift Register

n-bit Bi-directional Dynamic Shift Register

• Can serve as a Stack
• FIFO and FILO operation
Shift Registers

Dynamic Shift Register

n-bit Parallel-Load, Parallel-Read Bidirectional Dynamic Shift Register

- Useful for Parallel to Serial and Serial to Parallel Conversion
- Can be put in static hold state if $T_L$ and $T_R$ replaced with HCTL and HCTL
Array Logic

• Array logic is often used for sections of logic that may change later in the design or that will be changed for different variants of a product
• FPGA are a special case of array logic
• Can personalize array logic with only one layer of metal
  – Very quick turn-around and low incremental costs (as few as one additional mask)
Array Logic

Will consider only two types

– Gate Array
– Sea of Gates

Variants of the following approach are possible depending upon process but this will convey the basic concepts
Array Logic

Gate Array

- Can add M1 (blue), M2 (purple), contact (M1 to Poly), via (M1 to M2)
- Upper and lower metal shown actually lie above poly and are already present
- Assume upper M1 is $V_{DD}$ and lower M1 is $V_{SS}$
- Array can be very large
- Routing channels between segments of array
Array Logic

Sea of Gates

- Can add M1 (blue), M2 (purple), contact (M1 to Poly), via (M1 to M2)
- Upper and lower metal shown actually lie above poly and are already present
- Assume upper M1 is $V_{DD}$ and lower M1 is $V_{SS}$
- Array can be very large
- Routing channels between segments of array
Array Logic

Gate Array

Example:

Via (M1 to M2)
Contact (M1 to diff, Poly)
Array Logic

Gate Array

V_{DD}

A

B

V_{SS}

F

G

Example:

Diagram showing components and connections:

- **Via (M1 to M2)**
- **Contact (M1 to diff,Poly)**
Array Logic

Gate Array

Example:

\[ A \rightarrow F \rightarrow G \]

- \( V_{DD} \)
- \( V_{SS} \)
- \( A, B, F, G \)

- Via (M1 to M2)
- Contact (M1 to diff,Poly)
Array Logic

Sea of Gates

\[ V_{DD} \]

\[ V_{SS} \]

Example:

- **Via (M1 to M2)**
- **Contact (M1 to diff, Poly)**
Array Logic

Sea of Gates

$V_{DD}$

$V_{SS}$

Example:

- Via (M1 to M2)
- Contact (M1 to diff, Poly)
Array Logic

Sea of Gates

Example:

Via (M1 to M2)
Contact (M1 to diff, Poly)
Array Logic

Example:

Via (M1 to M2)
Contact (M1 to diff,Poly)
End of Lecture 43