Other Logic Styles
Ring Oscillators
Sequencial Circuits
Array Logic
Memory Structures
Logic Styles

- Static CMOS
- Complex Logic Gates
- Pass Transistor Logic (PTL)
- Pseudo NMOS
- Dynamic Logic
  - Domino
  - Zipper
Static CMOS

- Widely used
- Attractive static power dissipation and signal swing
- Dynamic Power Dissipation can become large
- Device count can get large
Static CMOS

Example: \( F = A \oplus B \)

\[ F = A \overline{B} + \overline{A} B \]

Straightforward Static CMOS implementation

22 transistors, 5 levels of logic
Static CMOS

Example: \( F = A \oplus B \)  \( F = AB + \overline{AB} \)

Recall “Bubble Pushing”
Static CMOS

Example: \( F = A \oplus B \)  \( F = A \bar{B} + \bar{A} B \)

18 transistors, 4 levels of logic
Static CMOS

Example: \( F = A \oplus B \)  
\[ F = A \overline{B} + \overline{A} B \]

Static CMOS implementation
Static CMOS

Example: \[ F = A \oplus B \]

\[ F = A \overline{B} + \overline{A} B \]

16 transistors, 3 levels of logic

Number of devices is unacceptably large in some applications
Logic Styles

- Static CMOS
- Complex Logic Gates
- Pass Transistor Logic (PTL)
- Pseudo NMOS
- Dynamic Logic
  - Domino
  - Zipper
Complex Logic Gates

Recall: Basic gates can be decomposed into n-channel PD network and p-channel PU network.

Concept can be extended to more general PD and PU networks and signal swing and static power properties will be retained if PD of n-channel and PU of p-channel.
Complex Logic Gates

- Implement B in PDN
- Implement B in PUN with complimented input variables
- Zero static power dissipation
- $V_H = V_{DD}$, $V_L = 0V$ (or $V_{SS}$)
- Complimented input variables often required
Complex Logic Gates

Example: \( F = A \oplus B \)

\[
F = AB + \overline{AB}
\]

\[
\overline{F} = \overline{A} + B \cdot A + \overline{B}
\]

\( \overline{A} \) and \( \overline{B} \) need to be generated

12 transistors, 2 levels of logic

Sizing can be:
- Minimum Size
- Equal worst-case rise/fall
- Equal worst-case rise/fall with OD
- Arbitrary
Complex Logic Gates were discussed at the beginning of the course.
Logic Styles

- Static CMOS
- Complex Logic Gates
- Pass Transistor Logic (PTL)
- Pseudo NMOS
- Dynamic Logic
  - Domino
  - Zipper
Pass Transistor Logic

Consider:

\[ V_{\text{DD}} \quad A \quad F \quad R_x \quad F = A \]

- Requires only two components
- Implements noninverting function!
Pass Transistor Logic

Consider:

\[ F = \overline{A} \]

Inverter
Pass Transistor Logic

F = AB

AND Gate

Requires only 3 components

Even simpler AND gate, requires only 2 components
Pass Transistor Logic

If complimented input variables available, may be useful
Pass Transistor Logic

\[ F = A \land B \]

\[ F = A \land B \land C \]

\( V_{DD} \) can be replaced with a Boolean variable
Pass Transistor Logic Gates

F = A

F = A
Pass Transistor Logic Gates

\[ F = A \cdot B \]

\[ F = A \cdot B \]
Pass Transistor Logic Gates

\[ F = A + B \]

\[ F = A + B \]
Pass Transistor Logic Gates

\[ F = A \cdot B \]

\[ F = A \cdot B \]
Pass Transistor Logic Gates

\[ F = A \oplus B \]
Pass Transistor Logic

- Low device count implementation of non inverting function (can be dramatic)
- Logic Swing not rail to rail 😞
- Static power dissipation not 0 when F high 😞
- $R_{LG}$ may be unacceptably large 😞
- Slow $t_{LH}$ 😞
- Signal degradation can occur when multiple levels of logic are used 😞

\[ F = A \cdot B \]
Pass Transistor Logic

Methods of implementing $R_{LG}$

Physical Size of $R_{LG}$ too large
Pass Transistor Logic

Signal degradation

No inherent signal degradation
(if A, B and D to $V_{DD}$)

Signal degradation may occur with PTL

- Can intermix n-channel and p-channel devices to reduce/eliminate the signal degradation problem
- Can add static CMOS buffers to restore signals provided too much signal degradation has not occurred
Pass Transistor Logic
- signal restoration
Pass Transistor Logic
- signal restoration

Implement $\overline{F}$ in PTL and use single Static CMOS inverter
Observe all PTL gates discussed so far were of this form.
- PU network can contain a mixture of n-channel and p-channel devices.
- Any of the PU networks used for complex logic gates could also be used in PTL.
• PTL gates could also be designed with logic in PD network
• PD network can contain a mixture of n-channel and p-channel devices
• Any of the PD networks used in complex logic gates could be used in PTL
Pass Transistor Logic Gates

As an example:

\[ F = \overline{A} \]
Pseudo NMOS Logic

- May be viewed as a special case of PTL
- Ratioed Logic
- Static power dissipation not 0 (in PD state)
- Often used for really large number of inputs – e.g. NOR
- Would be particularly useful for identifying one (or more) of many events that occur very infrequently
Pseudo NMOS Logic

\[ A_1, A_2, \ldots, A_n \]

\[ V_{DD} \]

\[ F \]

\[ n \] could be several hundred or even several thousand
Dynamic Logic

- PTL reduced complexity of either PUN or PDN to single “resistor"
- PTL relaxed requirement of all n-channel or all p-channel devices in PUN/PDN

What is the biggest contributor to area?

What is biggest contributor to dynamic power dissipation?

PUN and is responsible for approximately 75% of the dynamic power dissipation in inverter, more in NOR gates, and Well!

Can the PUN be eliminated W/O compromising signal levels and power dissipation?
Can the PUN be eliminated W/O compromising signal levels and power dissipation?

Benefits could be most significant!
Consider:

Precharges F to “1” when φ is low
F either stays high if output is to be high or changes to low on evaluation
Dynamic Logic

Consider:

- Termed Dynamic Logic Gates
- Parasitic capacitors actually replace $C_D$
- If Logic Block is $n$-channel, will have rail to rail swings
- Logic Block is simply a PDN that implements $F$

---

**Figure:**

- $F = \overline{A}$
- $F = A \cdot B$
- $F = A + B$

**Time Period:**

$: T_{CLK}$
Dynamic Logic

Basic Dynamic Logic Gate

Any of the PDNs used in complex logic gates would work here!

- Have eliminate the PUN!
- Ideally will have a factor of 4 or more reduction in $C_{IN}$
- Ideally will have a factor of 4 or more reduction in dynamic power dissipation relative to that of equal rise/fall!
- Ideally will have a factor of 2 reduction in dynamic power dissipation relative to that of minimum size!
Dynamic Logic

Basic Dynamic Logic Gate

What about the speed?
Consider the inverter
\[ C_{IN} = C_{OX} W_{MIN} L_{MIN} = \frac{C_{REF}}{4} \]
\[ R_{PD} = R_{PDREF} \]
Assume \( \phi \) is nominally a 50% duty cycle
Recall \[ t_{REF} = 2 R_{PDREF} C_{REF} \]
\[ \frac{T_{CLK}}{2} = R_{PDREF} \frac{C_{REF}}{4} \]
\[ T_{CLK} = R_{PDREF} \frac{C_{REF}}{2} = \frac{t_{REF}}{4} \]
What about the speed?

\[ T_{CLK} = R_{PDREF} \frac{C_{REF}}{2} = \frac{t_{REF}}{4} \]

\[ t_{HL} = \frac{t_{REF}}{4} \]

\[ t_{LH} = 0 \]

Ideally, dynamic logic is 4 times as fast as equal rise/fall CMOS.
Dynamic Logic

Dynamic XOR Gate
Dynamic logic (properly designed) is over twice as fast as normal logic. It uses only fast N transistors, and is amenable to transistor sizing optimizations. Static logic is slower because it has twice the loading, higher thresholds, and actually uses slow P transistors to compute things. Domino logic may be harder to work with, but if you need the speed, there is no other choice. Anything you buy that runs over 2GHz in 2007 uses dynamic logic. Another advantage is low power. A dynamic logic circuit running at 1/2 voltage will consume 1/4 the power of normal logic. Also each rail can convey an arbitrary number of bits, and there are no power-wasting glitches. Also power-saving clock gating and asynchronous techniques are much more natural in dynamic logic.
Dynamic Logic

Basic Dynamic Logic Gate

Advantages:

• Lower dynamic power dissipation (Ideally 4X)
• Improved speed (ideally 4X)

Limitations:

• Output only valid during evaluate state
• Need to route a clock
  (and this dissipates some power)
• Premature Discharge!
• More complicated
• Charge storage on internal nodes of PDN
• No Static hold if output H
Dynamic Logic

Premature Discharge Problem

If $A$ is high, then $F$ may go low at the start of the evaluate cycle and there is no way to recover a high output later in the evaluate phase - i.e. there may be a boolean error!

Can not reliably cascade dynamic logic gates!
Dynamic Logic

Premature Discharge Problem

This problem occurs when any inputs to an arbitrary dynamic logic gate create an $R_{PD}$ path in the PDN during at the start of the evaluate phase that is not to pull down later in that evaluate phase.

How can this problem be fixed?

Precharging to the low level all inputs to a PDN that may change to the high state later in the evaluate cycle (called domino).

Alternating gates with n-channel and p-channel pull networks (Zipper Logic)
Adding an inverter at the output will cause $F$ to precharge low so it can serve as input to subsequent gate w/o causing premature discharge.

Implement $F$ instead of $\overline{F}$ in the PDN.

Termed **Domino Logic**

Some additional dynamic power dissipation in the inverter.

Some additional delay during the evaluate state in inverter.
Domino Logic

![Diagram of a domino logic circuit with inputs A, n, and φ, output F, and power supply V_{DD}.]
Dynamic Logic

- p-channel logic gate will pre-charge low
- Phasing of PUN and PDN networks is reversed
- Some performance loss with p-channel logic devices
- Direct coupling between alternate type dynamic gates is possible without causing a premature discharge problem
Dynamic Logic

Direct coupling between alternate type dynamic gates
Zipper Logic

Map gates to appropriate precharge type
Zipper Logic

Acceptable Implementation in Zipper
Unacceptable Implementation in Zipper
- Premature discharge at output of 2-input NAND
Static Hold Option

If not clocked, charge on upper node of PDN will drain off causing H output to degrade
Static Hold Option

- Weak p will hold charge
- Size may be big (long L)
- Some static power dissipation
- Can use small current source
- Sometimes termed “keeper”

- Weak p will hold charge
- Size may be big (long L)
- Can eliminate static power with domino
- Sometimes termed “keeper”
Charge stored on internal nodes of PDN

If voltage on $C_{P1}$ and $C_{P2}$ was 0V on last evaluation, these may drain charge (charge redistribution) on $C_P$ if output is to evaluate high (e.g. On last evaluation $A_1=A_2=A_3=H$, on next evaluation $A_3=L$, $A_1=A_2=H$.)
Charge stored on internal nodes of PDN

Can precharge internal nodes to eliminate undesired charge redistribution
Dynamic Logic

Many variants of dynamic logic are around

- Domino
- Zipper
- Ratio-less 2-phase
- Ratio-less 4-phase
- Output Prediction Logic
- Fully differential
- ...

Benefits disappear, however, when interconnect (and diffusion) capacitances dominate gate capacitances
Dynamic logic will likely disappear in deep sub-micron processes because interconnect parasitics will dominate gate parasitics.
Other types of Logic (list is not complete and some have many sub-types)

<table>
<thead>
<tr>
<th>From Wikipedia:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>B</strong> BiCMOS</td>
</tr>
<tr>
<td><strong>C</strong> CMOS</td>
</tr>
<tr>
<td>Cascode Voltage Switch Logic</td>
</tr>
<tr>
<td>Clocked logic</td>
</tr>
<tr>
<td>Complementary Pass-transistor Logic</td>
</tr>
<tr>
<td>Current mode logic</td>
</tr>
<tr>
<td>Current steering logic</td>
</tr>
<tr>
<td><strong>D</strong> Differential TTL</td>
</tr>
<tr>
<td>Diode logic</td>
</tr>
<tr>
<td>Diode–transistor logic</td>
</tr>
<tr>
<td>Domino logic</td>
</tr>
<tr>
<td>Dynamic logic (digital logic)</td>
</tr>
<tr>
<td><strong>E</strong> Emitter-coupled logic</td>
</tr>
<tr>
<td><strong>F</strong> Four-phase logic</td>
</tr>
<tr>
<td><strong>G</strong> Gunning Transceiver Logic</td>
</tr>
<tr>
<td><strong>H</strong> HMOS</td>
</tr>
<tr>
<td>HVDS</td>
</tr>
<tr>
<td>High-voltage differential signaling</td>
</tr>
<tr>
<td><strong>I</strong> Integrated injection logic</td>
</tr>
<tr>
<td><strong>L</strong> LVDS</td>
</tr>
<tr>
<td>Low-voltage differential signaling</td>
</tr>
<tr>
<td>Low-voltage positive emitter-coupled logic</td>
</tr>
<tr>
<td><strong>M</strong> Multi-threshold CMOS</td>
</tr>
<tr>
<td><strong>N</strong> NMOS logic</td>
</tr>
<tr>
<td><strong>P</strong> PMOS logic</td>
</tr>
<tr>
<td>Philips NORbits</td>
</tr>
<tr>
<td>Positive emitter-coupled logic</td>
</tr>
<tr>
<td><strong>R</strong> Resistor-transistor logic</td>
</tr>
<tr>
<td><strong>S</strong> Static logic (digital logic)</td>
</tr>
<tr>
<td><strong>T</strong> Transistor–transistor logic</td>
</tr>
</tbody>
</table>
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

$$f_{\text{OSC}} \approx \frac{1}{nt_{\text{PROP}}}$$

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

- **Latch**
  - D is held on Q when $C_{LK}$ is low
  - Q may also be present

- **Pulsed Latch**
  - D is held on Q when $C_{LK}$ is low
  - $C_{LK}$ is a pulse
  - Q 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

- **C_LK**
- **D**
- **Q**

- **C_P**
- **D**
- **Q**

- **C_{\phi_1}**
- **\overline{C}_{\phi_1}**
- **D**
- **Q**

- **Latch**
- **Pulsed Latch**
- **Flip Flop**

(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

\( C_{\text{LK}} \)

\( \overline{C}_P \)

\( D \)

\( X_C \)

\( \bar{Q} \)

\( Q \)
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

Q

S

R

K

J

D

D
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
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

Clock ed 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

5 transistors

Output transparent when in clock is high

With static hold
6 transistors
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:

```
D  Q
```

```
SR    TR  Q_1
```

```
D  Q  Q_1
```

```
TR  Q_1
```

```
SR  Q_1
```

```
D
```

```
Q_1
```

```
Q
```

---

**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

Simple redrawing
Shift Registers

Dynamic Shift Register

$D \rightarrow SR \rightarrow TR \rightarrow D \rightarrow SR \rightarrow TR \rightarrow Q$

$n$-bit Dynamic Shift Register

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

**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

- Can serve as a Stack
- FIFO and FILO operation

n-bit Bi-directional Dynamic Shift Register
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:

Diagram:

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

Gate Array

Example:

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

Gate Array

Example:

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

Sea of Gates

$V_{DD}$

$V_{SS}$

Example:

\[
\begin{array}{c}
A \\
B
\end{array}
\quad \rightarrow \quad
\begin{array}{c}
F \\
G
\end{array}
\]

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

Sea of Gates

$V_{DD}$

$V_{SS}$

Diffusion Partition

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)
Typical Memory Structure

- **Row Decoder**
- **Column Decoder**
- **Memory Array**
- **Sense Amplifier**
- **V_{DD}**
- **ADR**
- **DATA**
- **n = n_1 + n_2**

Mathematical expression:

\[ n = n_1 + n_2 \]
Row Decoder Architectures

Row decoder is Pseudo n-MOS NOR Gate

Typically n/2 inputs where n is the address length

\[ R_k = \overline{A_1} \cdot \overline{A_2} \cdot \overline{A_3} \]

\[ R_k = A_1 + A_2 + A_3 \]
Row Decoder Architectures

Transistor sites typically reserved in the layout for efficient, compact layout
Row Decoder Architectures

Pull-up resistor implemented with either weak p or with dynamic precharge by taking clock $\phi$ low to precharge to high (thus dynamic NOR gate)

$V_{DD}$

$V_{DD}$

$V_{DD}$

$C_P$
Mem Cells

Static RAM (SRAM)

- Uses PTL and cross-coupled inverters
- Sizing of “switches” must be strong enough to write to cell
- No static power dissipation in this PTL implementation
Mem Cells

Static ROM (Mask programmable ROM)

- Site reserved for possible transistor
- Actually programmed with contact to gate and diffusion
- Can personalize with one or two masks
- Single transistor per bit
- Uses only one column line
Mem Cells

EPROM or EEPROM

Control Gate

Source

Drain

Bulk

n-channel MOSFET

Floating Gate

Very Thin Tunneling Oxide

Floating Gate Transistor

- Very thin floating gate
- Charge tunnels onto gate during programming to change $V_T$ a lot
- Conceptual diagram only
- Somewhat specialized processing for reliable floating gate devices
Mem Cells

- Floating Gate Transistor
- Programmed by Changing the Threshold Voltage
- Nonvolatile Memory
- Can be electrically programmed with EEPROM
- Limited number of read/write cycles (but enough for most applications)
- Uses only one column line
Mem Cells

- Charge stored in small parasitic capacitor
- Very small cells
- Volatile and dynamic
- Special processes to make $C_P$ large in very small area
- $C_P$ is actually a part of the transistor
- Somewhat tedious architecture (details not shown) needed to sense very small charge