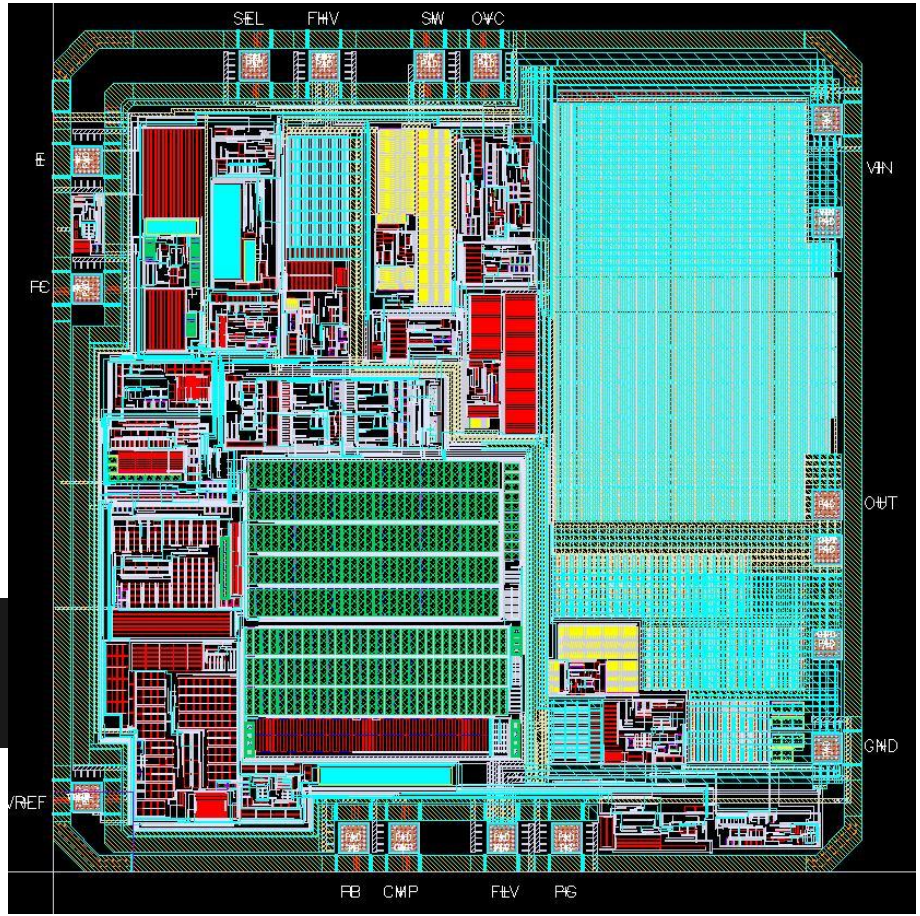
CHAPTER 3
LOW-POWER DESIGN

ECE4514

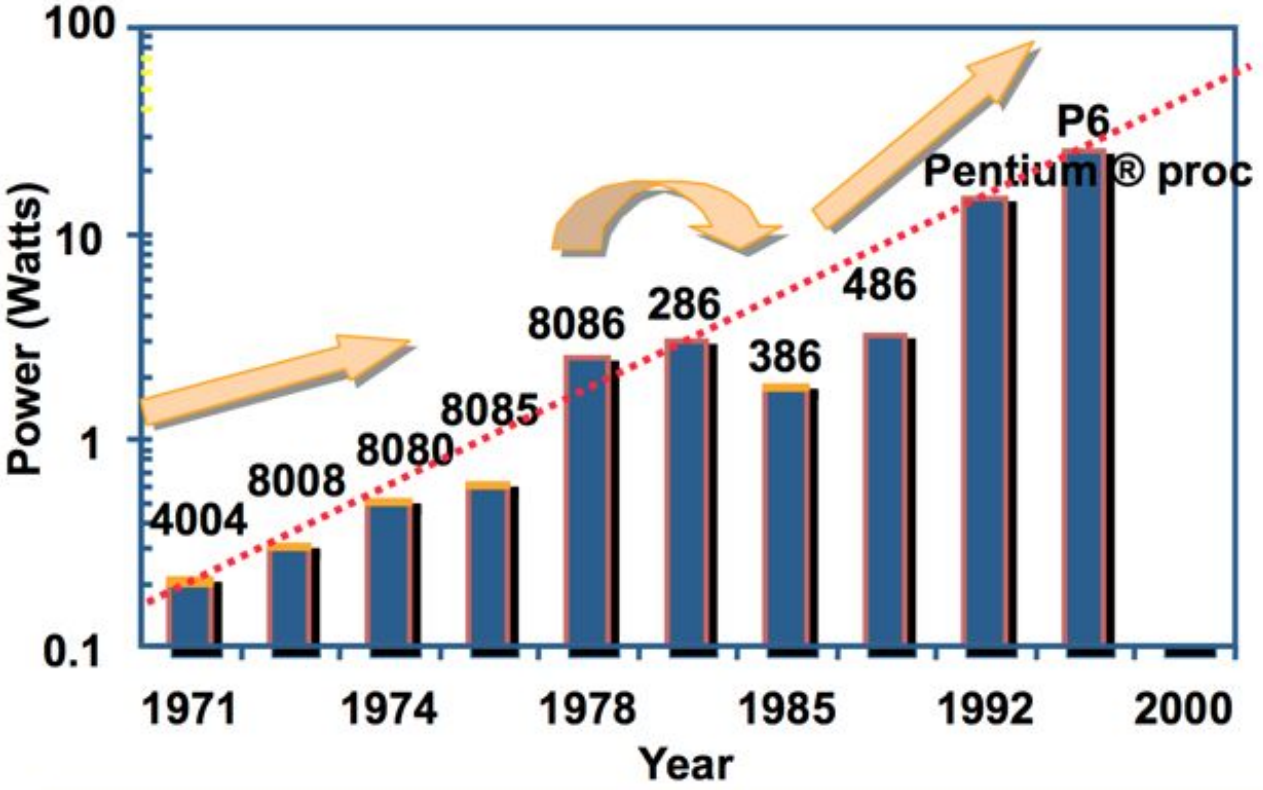
DIGITAL DESIGN 2



Topics

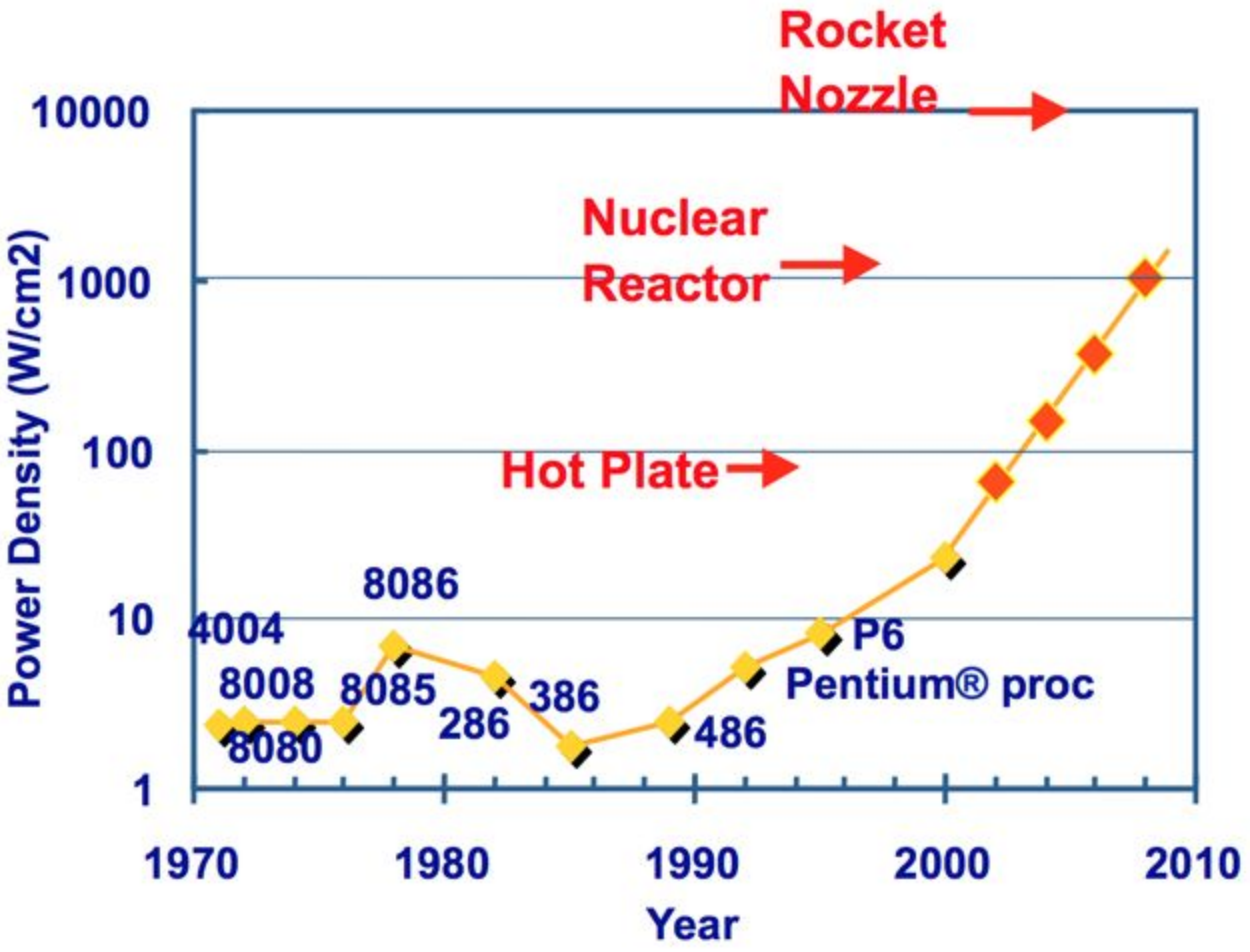
- Sources of power consumption
- The impact of clock control on dynamic power consumption
- Problems with clock gating
- Managing clock skew on gated clocks
- Input control for power minimization
- Impact of the core voltage supply
- Guidelines for dual-edge triggered flip-flops

Power in the 20th Century



Power increases despite Vdd decrease

Power Density

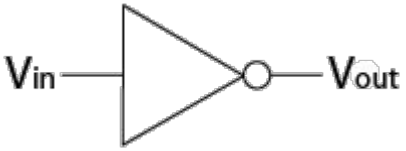


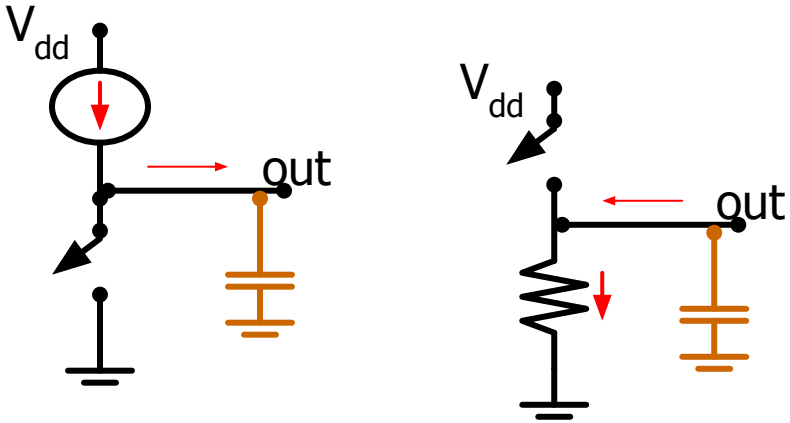
Power Dissipation

- Dynamic power
- Leakage power
- Glitches
- Short circuit power

Dynamic Power

- Occurs at each switching
- $P_d = C_L \bullet V_{dd}^2 \bullet f_p / 2$
- f_p switching frequency





AN INVERTER SWITCHING

Dynamic Power

When signals wiggle
power is consumed

Reduce Dynamic Power

$$P_d = \frac{1}{2}[\alpha \bullet C_L \bullet V_{dd}^2 \bullet f_p]$$

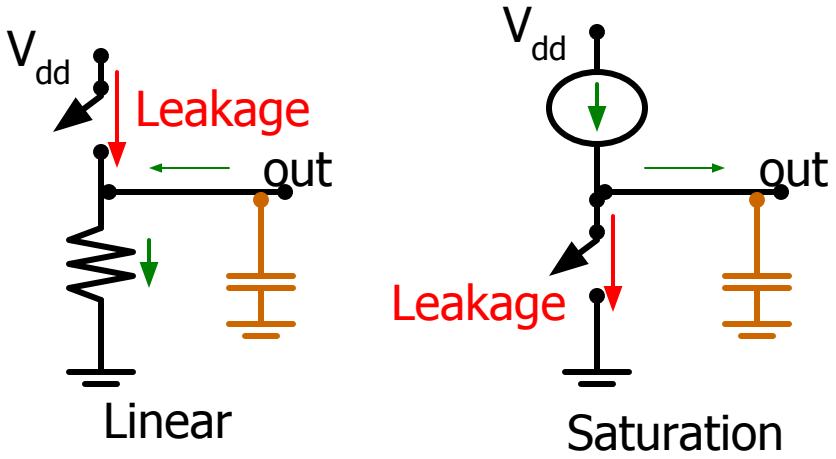
- α: *Activity Factor*
  clock gating, sleep mode
- C: use small transistors (esp. on clock), short wires
- V_{DD}: lowest suitable voltage
- f: lowest suitable frequency

Leakage Current

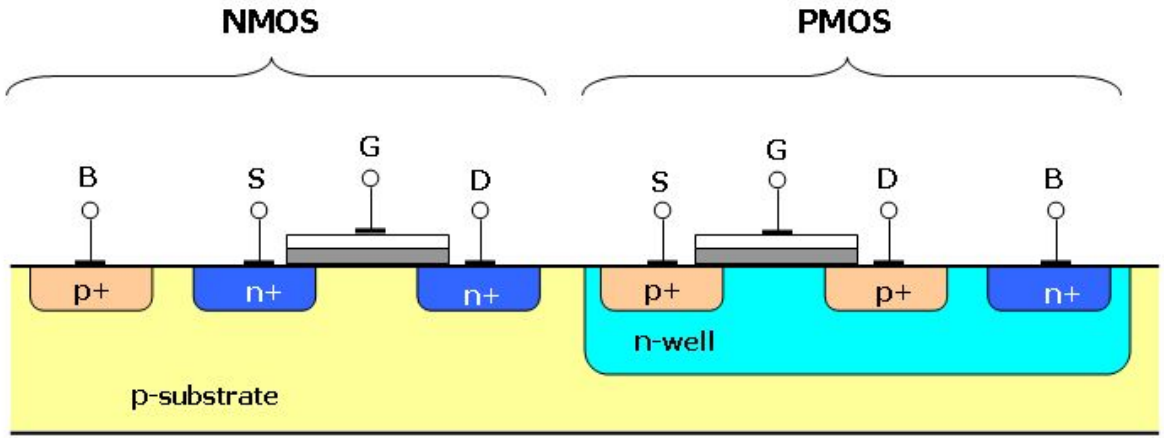
Main Sources of Leakage	Impact	Mitigation Techniques
Subthreshold leakage (I_{sub})	Dominant	■ Lower voltage ■ Higher voltage threshold ■ Longer gate length ■ Dopant profile optimization
Gate direct-tunneling leakage (I_G)	Dominant	High-k metal gate (HKMG)
Gate-induced gate leakage (I_{GIDL})	Small	Dopant profile optimization
Reverse-biased junction leakage current (I_{REV})	Negligible	Dopant profile optimization

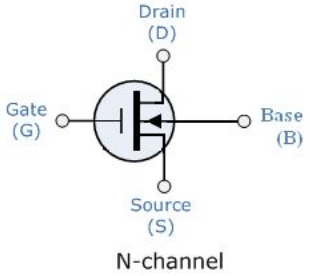
Subthreshold Leakage Current

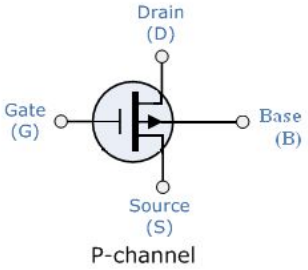
- Killer to CMOS technology

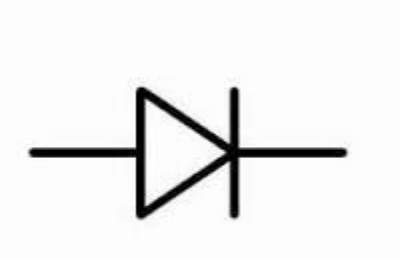


Leakage Current



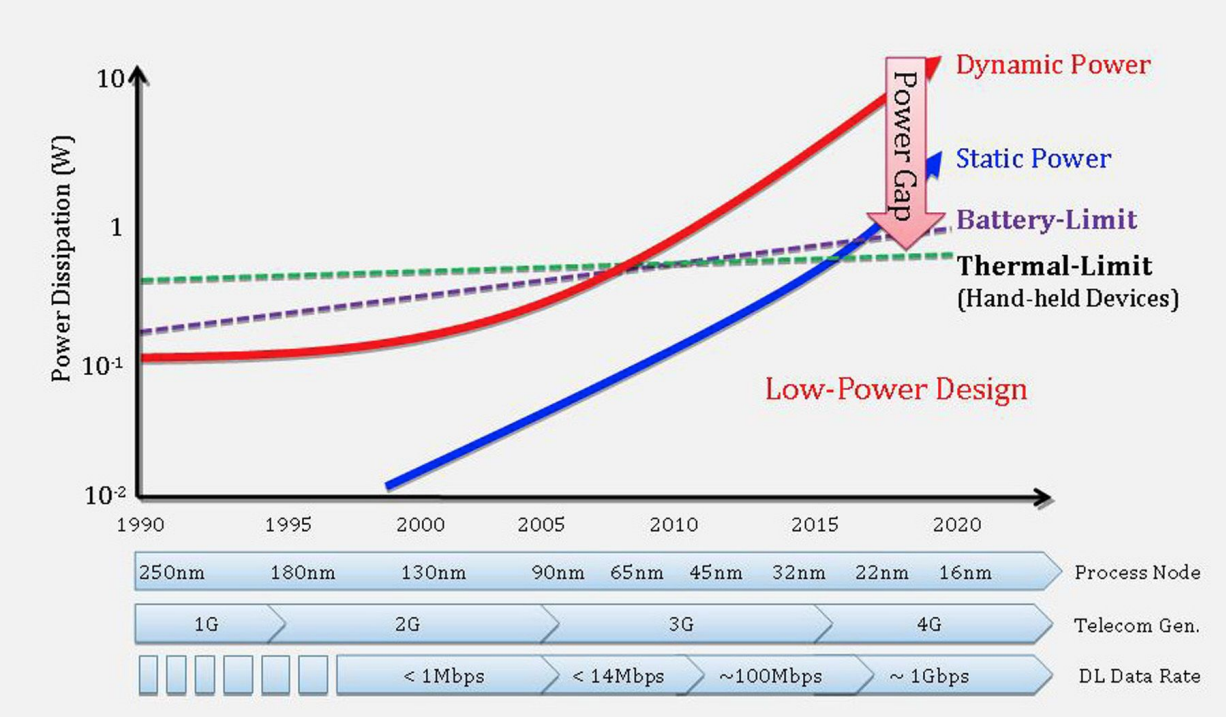




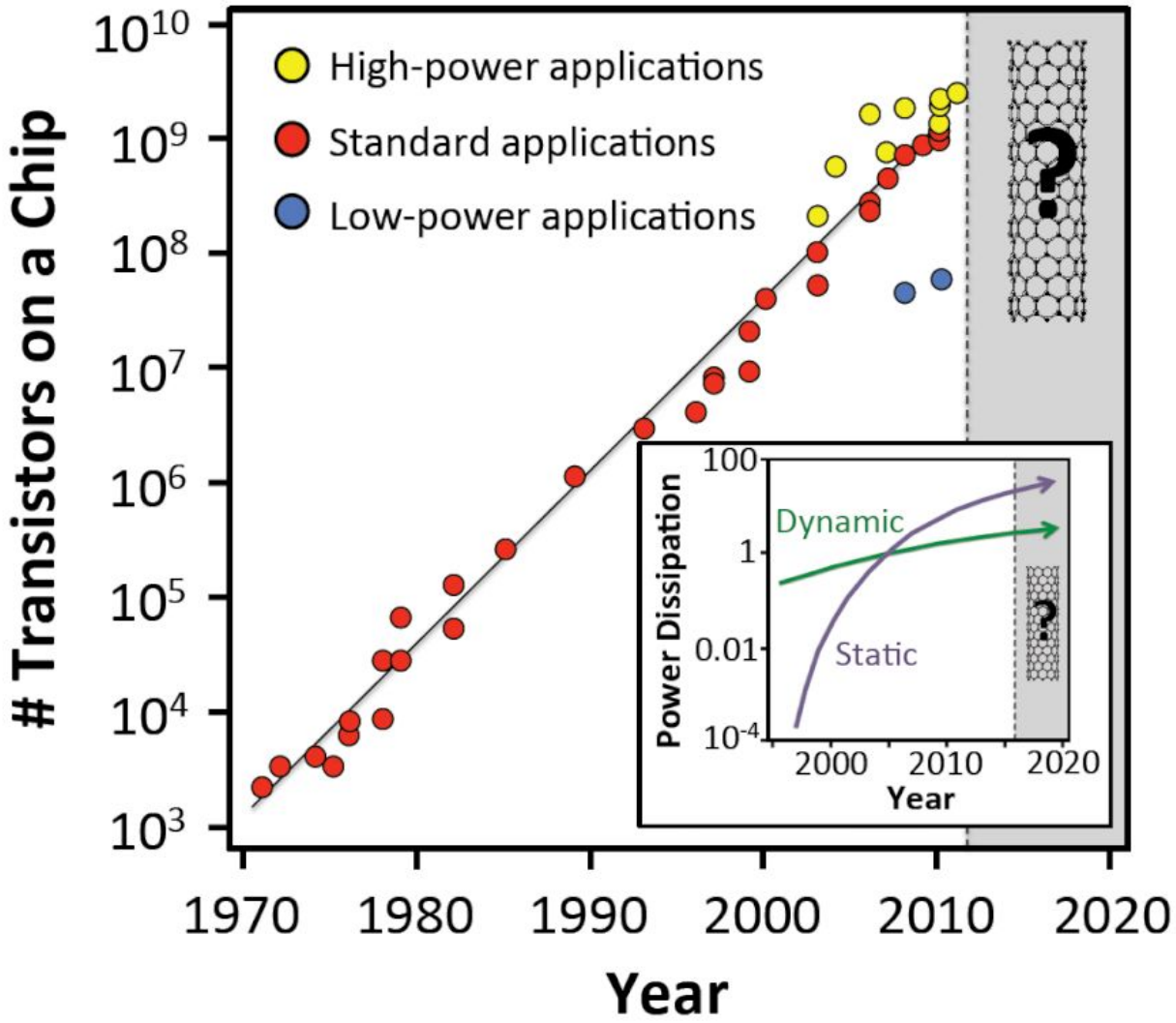


Where are the diodes?

Trends in Mobile Devices

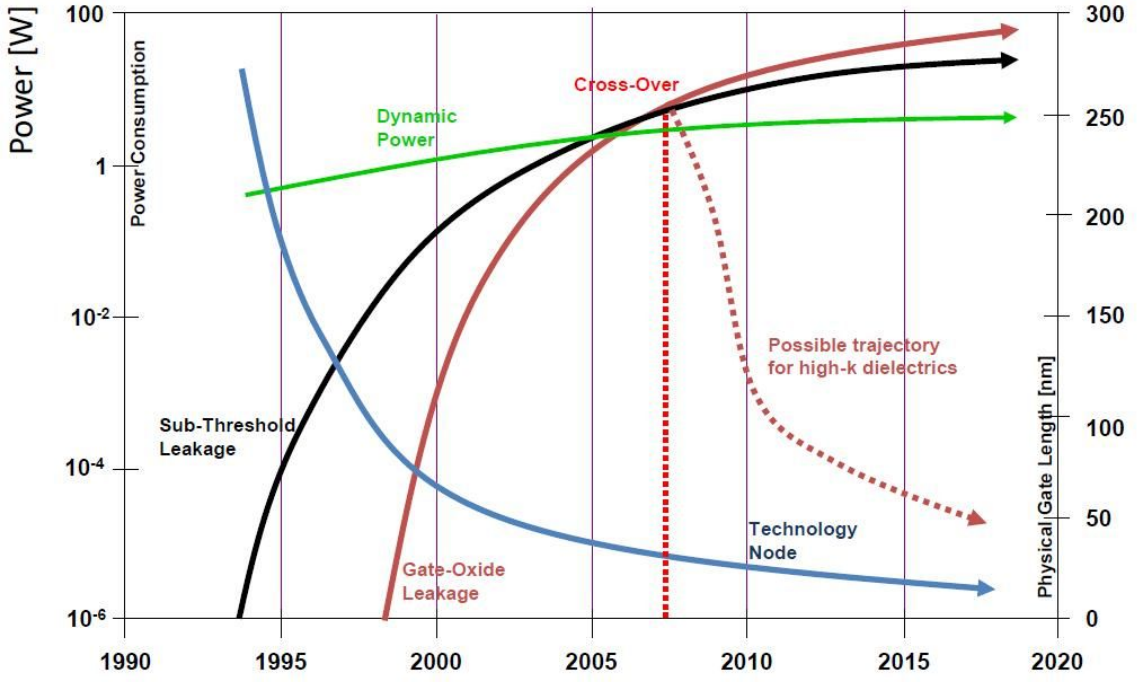


Moore and Moore Transistors

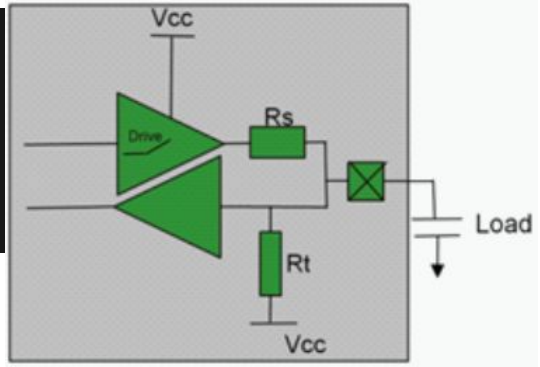


Leakage Trends

Source: ITRS Roadmap 2007

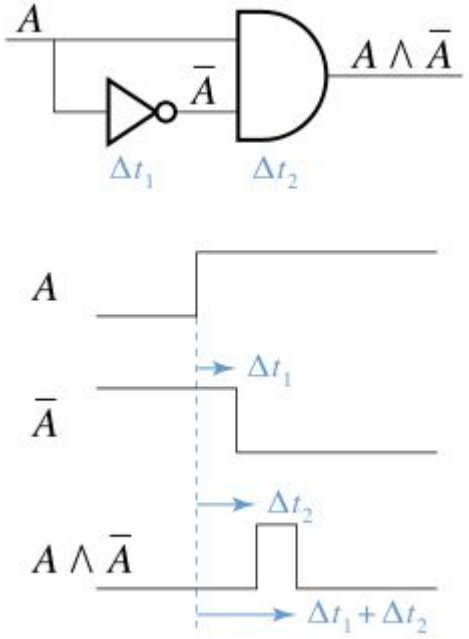


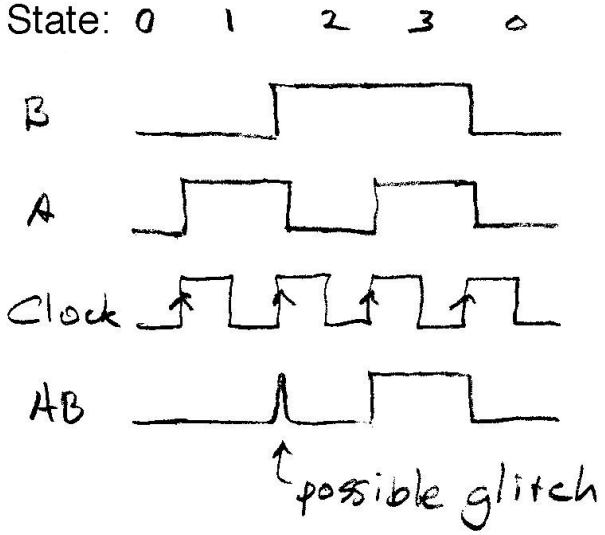
I/O Power



Main Factors Impacting I/O Power	Mitigation Techniques
Termination resistors (on-chip series termination (R_S OCT) and on-chip parallel termination (R_T OCT))	Dynamic on-chip termination (DOCT)
Output buffer drive strength	Programmable drive strength
Output buffer slew rate	Programmable slew rate
I/O standard (single ended, voltage referenced, or differential)	Support for multiple I/O standards
Voltage supply	Support for various voltage rails
Capacitive load (charging/discharging)	Interface dependent

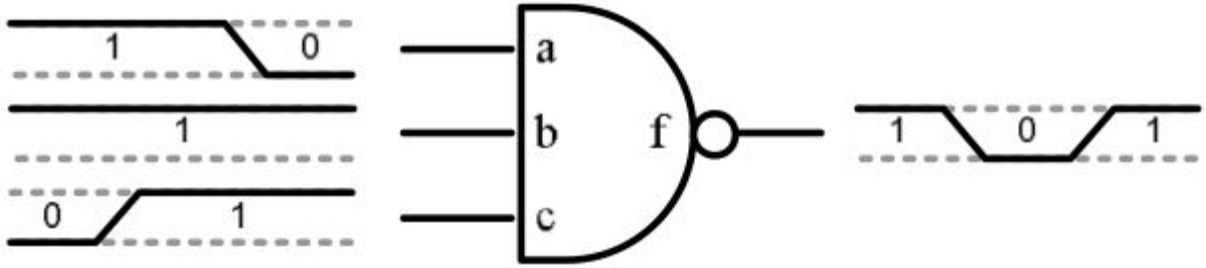
Glitches



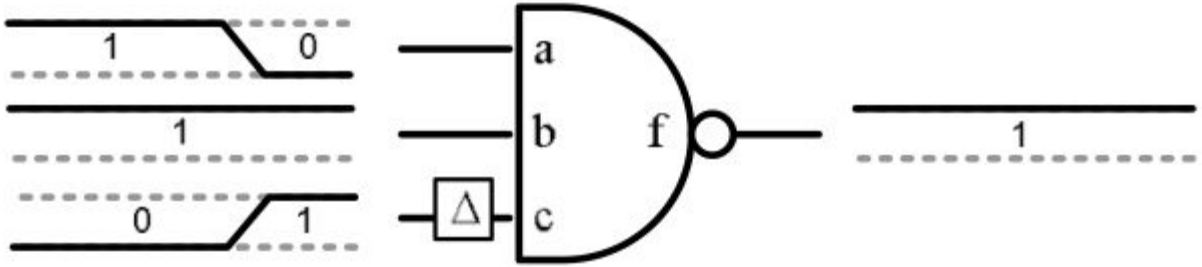


Decoding Glitch Example

Glitch Removal



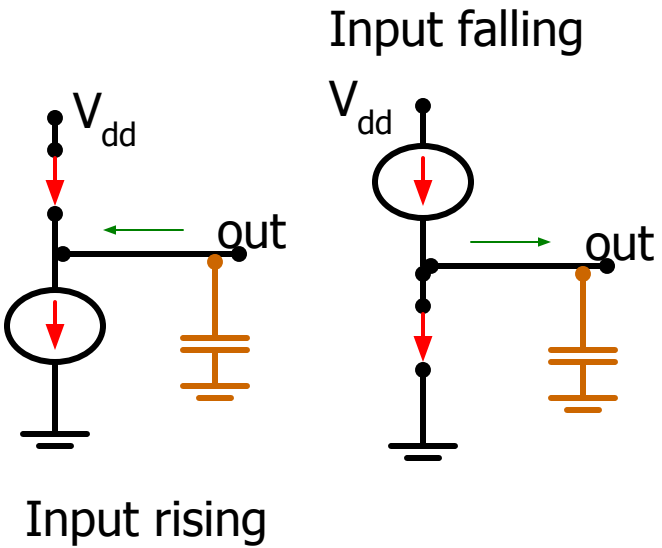
(a) Original circuit with glitch



(b) Glitch removed by delaying input c

Short Circuit Power

- During switching, there is a short moment when both PMOS and CMOS are partially on
- $P_s = Q \bullet (V_{dd}-V_t)^3 \bullet t_r \bullet f_p$
- t_r rising time



Formula not on the exam

Power Mitigation

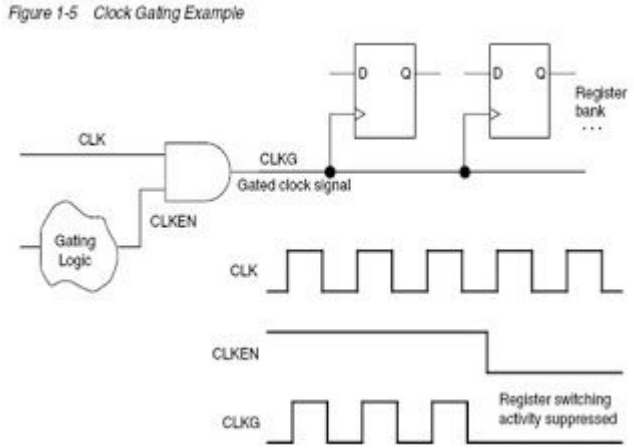
Things we can do as logic designers

Clock Gating

- Gate off clock to idle functional units
  - e.g., floating point units
  - Need logic to generate disable signal
    - increases complexity of control logic
    - timing critical to avoid clock glitches
  - Additional gate delay on clock signal
    - gating gate can replace a buffer in the clock distribution tree

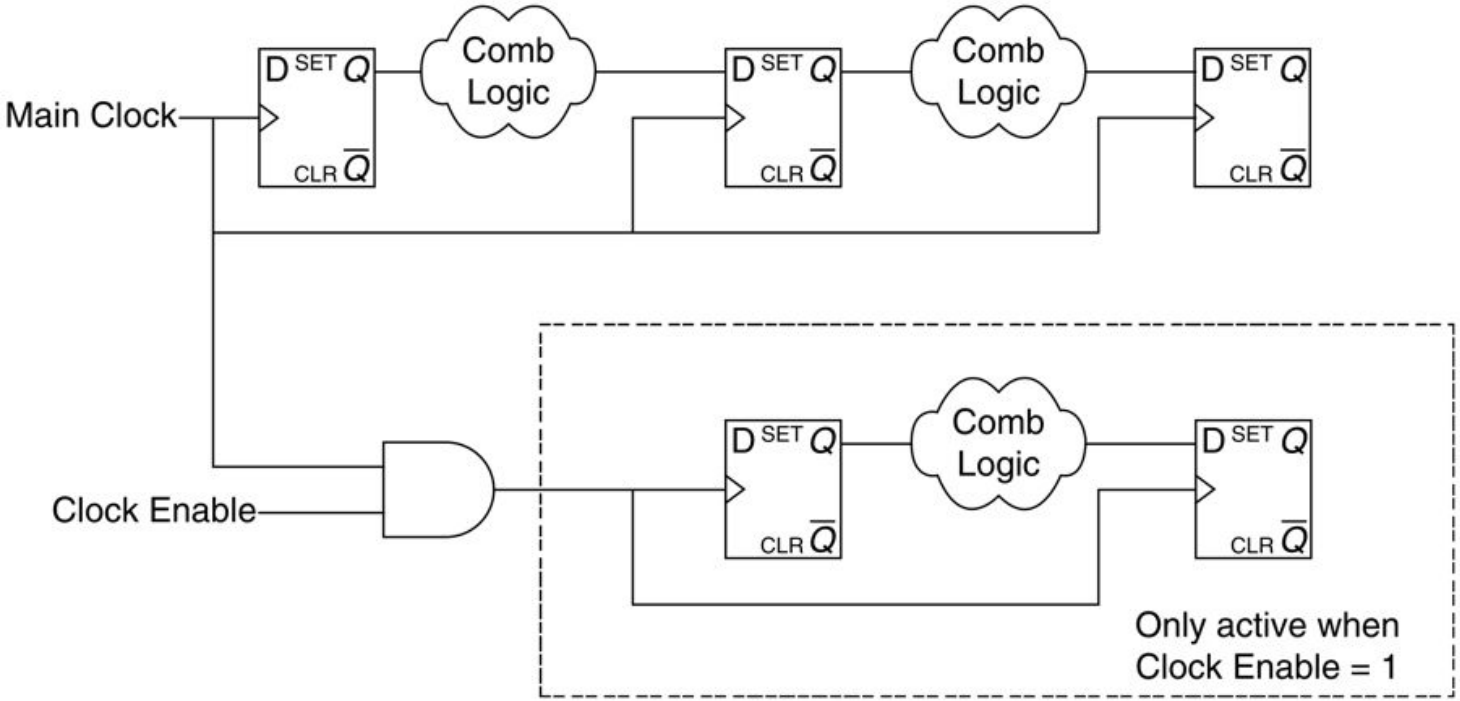
Clock Gating

A gated clock introduces a new clock domain and will create difficulties for the designer.

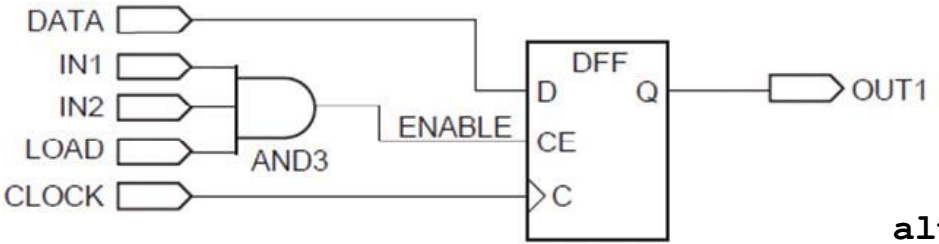


Generally viewed as a bad thing to do.

Clock Gating



Clock Gating vs Clock Enable




```
always@(posedge clock) begin
    if (in1 & in2 & load) begin
        out1 <= data;
    end
end
```

No impact on timing, yet, clock distribution network still wiggling.

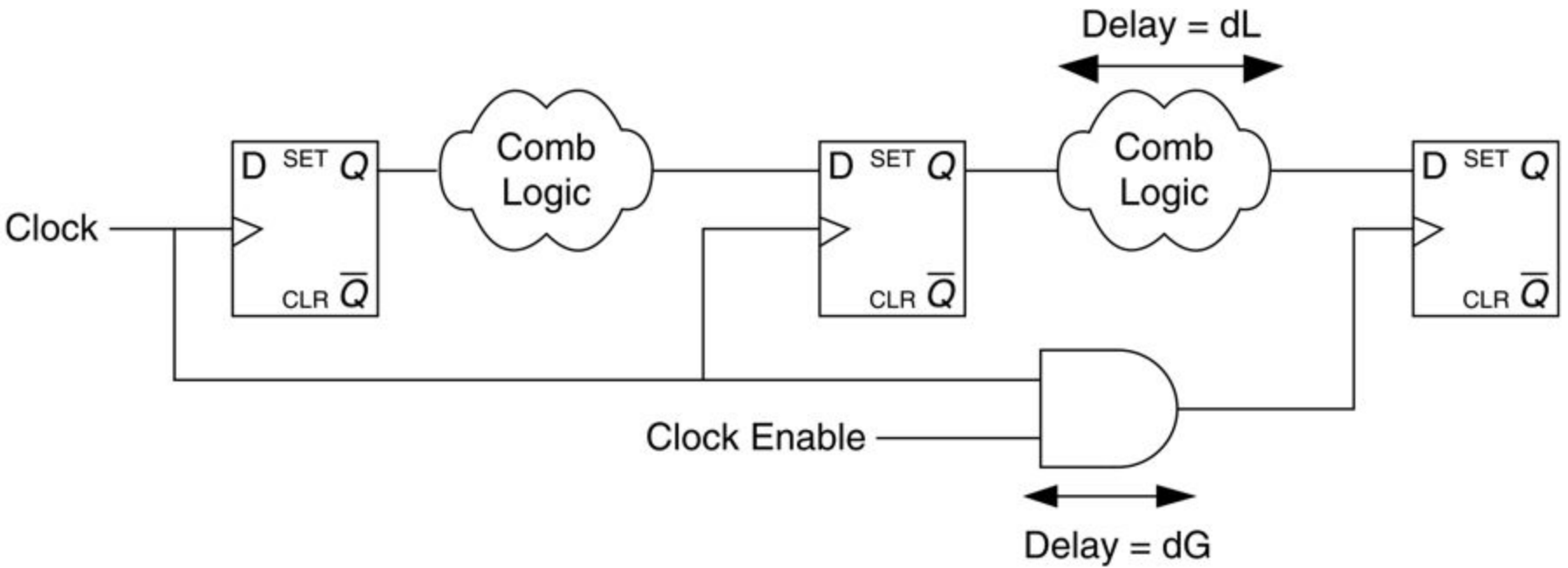
Clock Skew Danger

Mishandling clock skew can cause “catastrophic failures” in the FPGA.

Simplistic Clock Gating

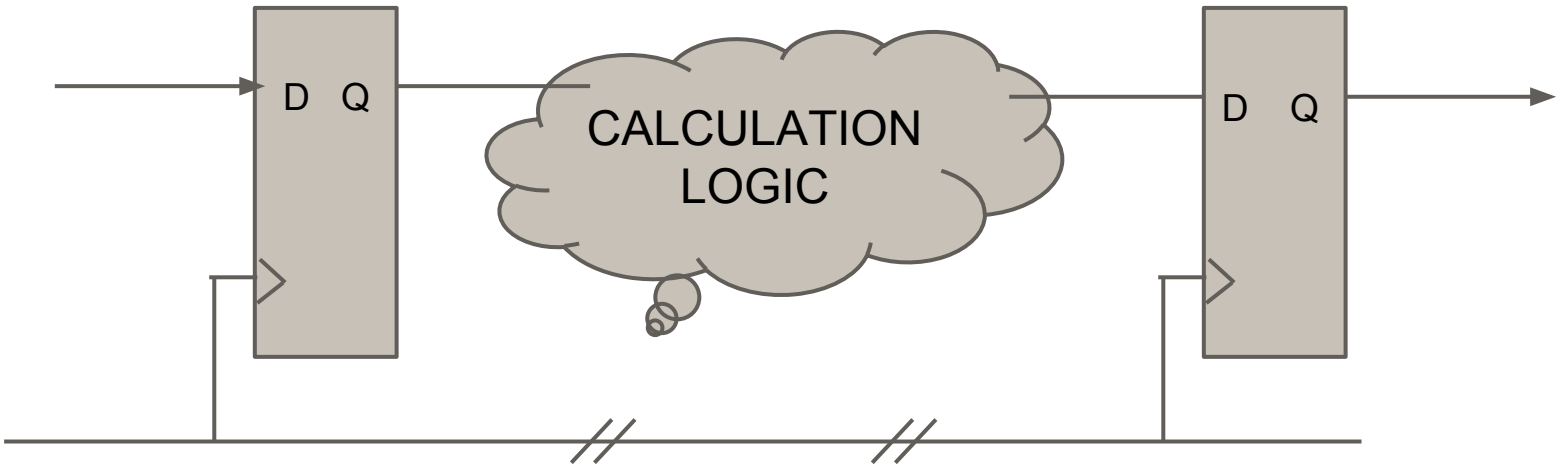
```
// Poor design practice
module clockgating(
  output dataout,
  input  clk, datain,
  input  clockgate1);
  reg    ff0, ff1, ff2;
  wire   clk1;

  // clocks are disabled when gate is low
  assign clk1    = clk & clockgate1;
  assign dataout = ff2;

  always @(posedge clk)
    ff0 <= datain;

  always @(posedge clk)
    ff1 <= ff0;

  always @(posedge clk1)
    ff2 <= ff1;
endmodule
```

Poor Design Practice

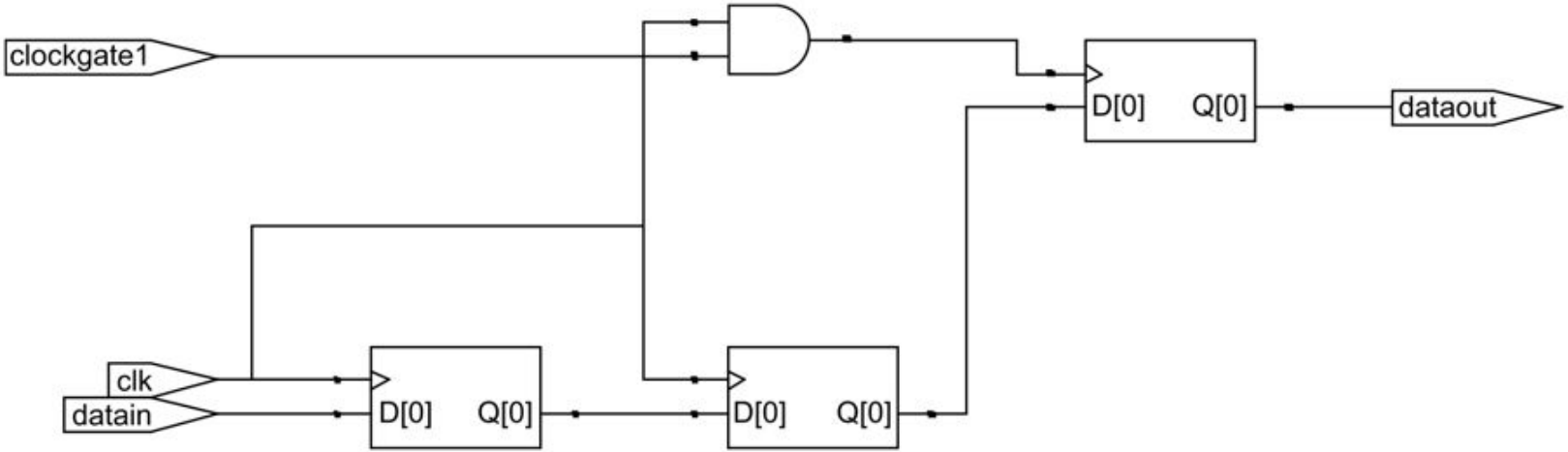


Timing

$$F_{max} = \frac{1}{T_{clk-q} + T_{logic} + T_{routing} + T_{setup} - T_{skew}}$$



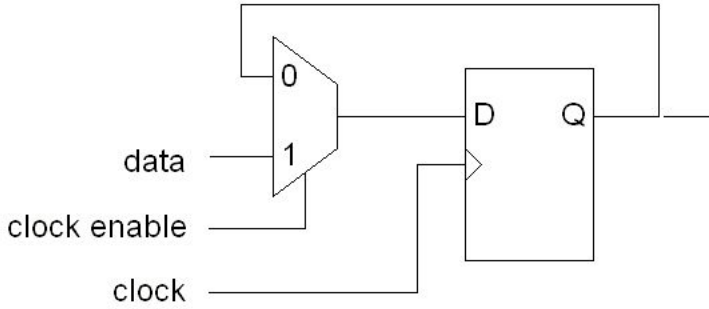
Implemented Design



Alternative Design

```
module noclockgating(
output dataout,
input clk, datain, input clockgate1);
reg ff0, ff1, ff2; wire clk1;
     // clocks are disabled when gate is
low
     assign dataout = ff2;
     always @(posedge clk)
       ff0 <= datain;
     always @(posedge clk)
       ff1 <= ff0;
     always @(posedge clk)
       if (clockgate1) ff2 <= ff1;
endmodule
```

Redo the design so that it uses a clock enable instead



Poor Design Practice

```
wire [8:0] count, cntout;

prescale pre( .clk(clk), .rst(rst),
.count(count) );

maincount main (.clk(count[8]), .ce(1),
.mainout( cntout ));
```

Better Design Practice

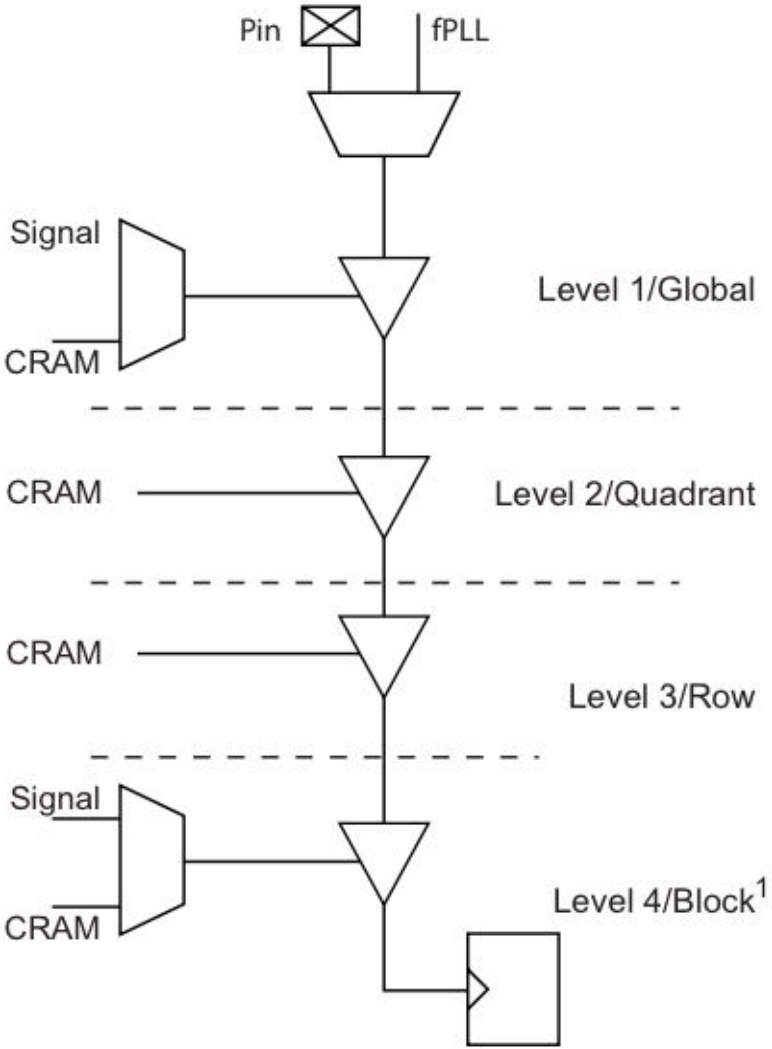
```
wire [8:0] count, cntout;

prescale pre( .clk(clk), .rst(rst),
.count(count) );
assign ce  = (count = 9'd38);
maincount main (.clk(clk), .ce(ce),
.mainout( cntout ));
```

Stratix Clock Gating

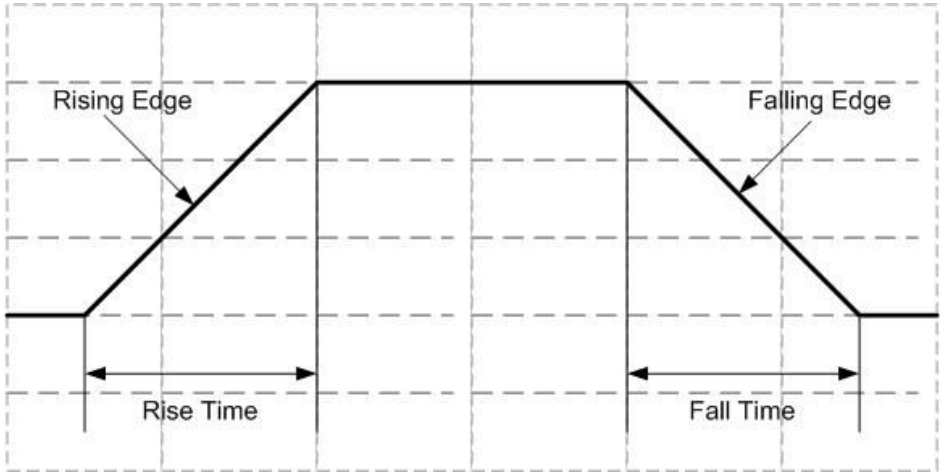
- **Static clock gating**—The clock is enabled or disabled using a configuration RAM bit at programming time. The Quartus II software leverages static clock gating on clocks that are either always used or never used in the design.

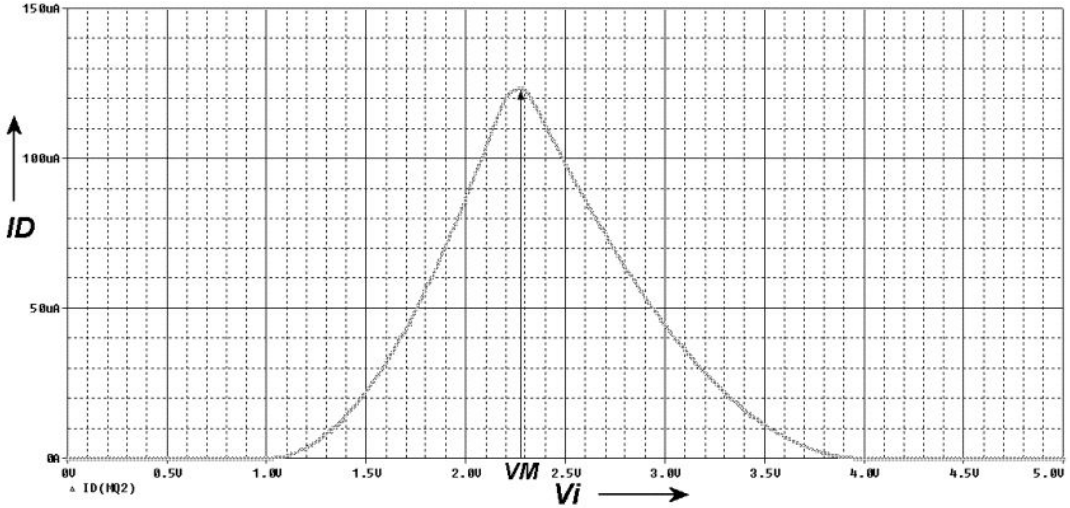
- **Dynamic clock gating**—The clock is enabled or disabled using a signal generated by designers or by the Quartus II software. Quartus II software synthesizes designs and generates signals at various clock stages to dynamically shut off clocks when the downstream logic does not need to toggle.

Stratix Clock Gating



Controlling Rise/Fall Time





Controlling Rise/Fall Time

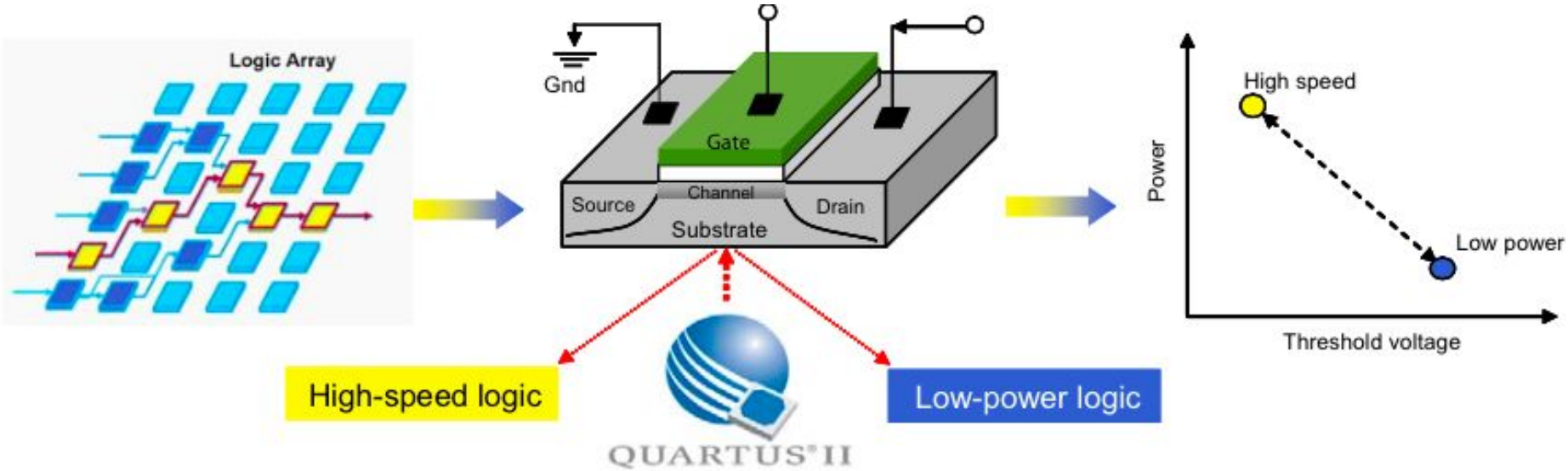
To minimize the power dissipation of input devices, minimize the rise and fall times of the signals that drive the input.

What about outputs?

Supply Voltage

Dynamic power dissipation drops off with the square of the core voltage, but reducing voltage will have a negative impact on performance.

Altera Programmable Power



Altera Programmable Power

- **Low-power mode**—Quartus II software reduces the back bias voltage (making it more negative), which increases the threshold voltage of transistors. This reduction minimizes sub-threshold leakage currents and unwanted static power in non-timing-critical circuit paths.

- **High-performance mode**—Quartus II software increases the back bias voltage (making it less negative), which reduces the threshold voltage of transistors for faster switching. This performance increase is required on logic in the few timing- critical paths to help meet the design's specified timing constraints and deliver maximum performance.

Dynamic Voltage and Frequency Scaling

- Run at the lowest supply voltage that meets the timing constraints
  - DFS (dynamic frequency scaling)
  - DVS (dynamic voltage scaling)
- A DVS+DFS system requires the following
  - A programmable clock generator (PLL)
    - PLL from 200MHz → 700MHz in increments of 33MHz
  - A supply regulation loop that sets the minimum V_{DD} necessary for operation at the desired frequency
  - An operating system that sets the required frequency + supply voltage to meet the task completion deadlines

Dual Edge Flip-Flops

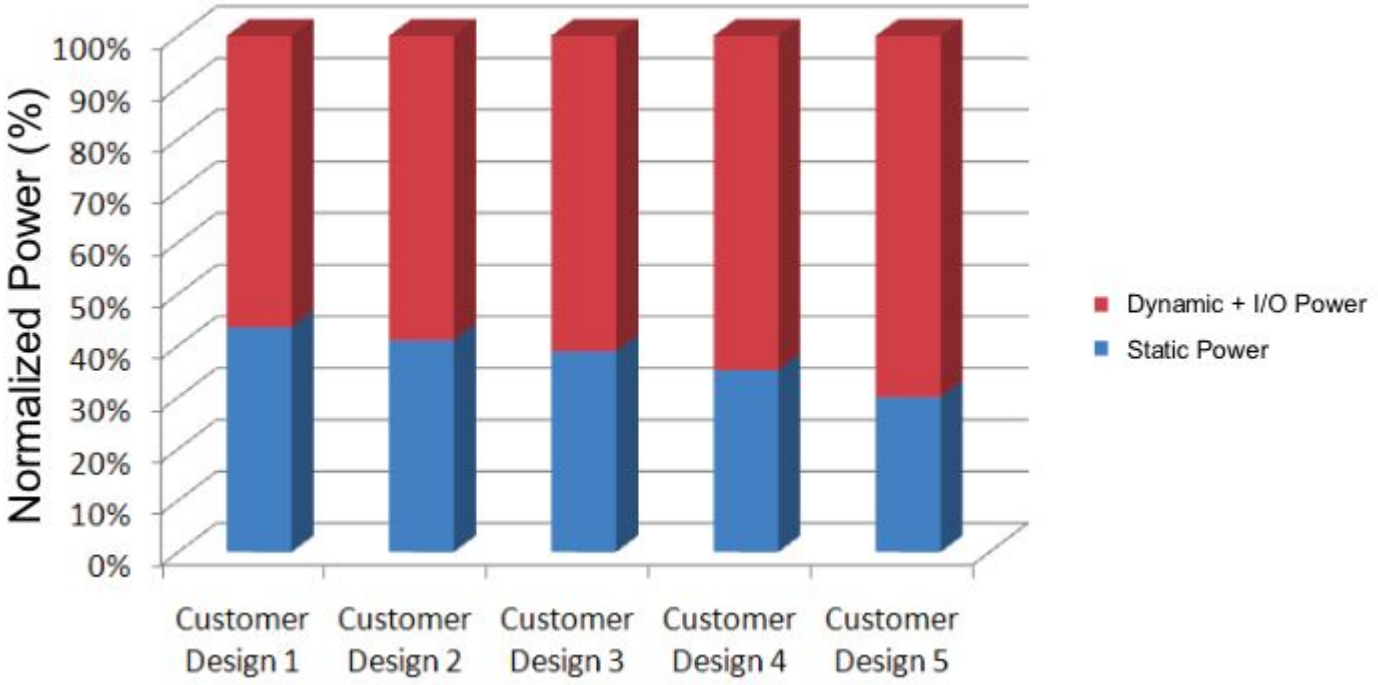
```
module dualedge(
output reg dataout, input clk, datain);
reg ff0, ff1;
    always @(posedge clk)
      ff0     <= datain;
    always @(posedge clk or negedge clk) begin
      ff1 <= ff0;
      dataout <= ff1;
    end
endmodule
```

Dual Edge Flip-Flops

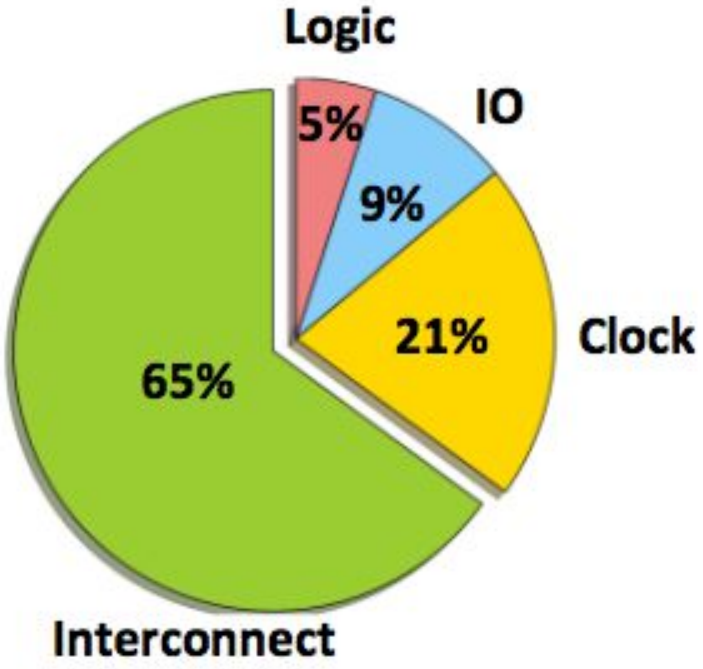
Dual-edge triggered flip-flops reduce the energy in the clock distribution network by 50%.

Dual-edge triggered flip-flops should only be used if they are provided as primitive elements (and they are NOT in Altera devices)

Power Consumed



FPGA Power Consumption



Most FPGA power consumption is attributed to losses in the programmable interconnect

Altera Gated Clocks

Entering the Danger Zone

FPGA Gated Clocks

In a design, you can use combinational logic as a clock signal that can prevent certain logic in a circuit from being activated by the clock signal and can therefore reduce the total power consumption of a device. The combinational logic that is used as a clock signal should follow the following guidelines

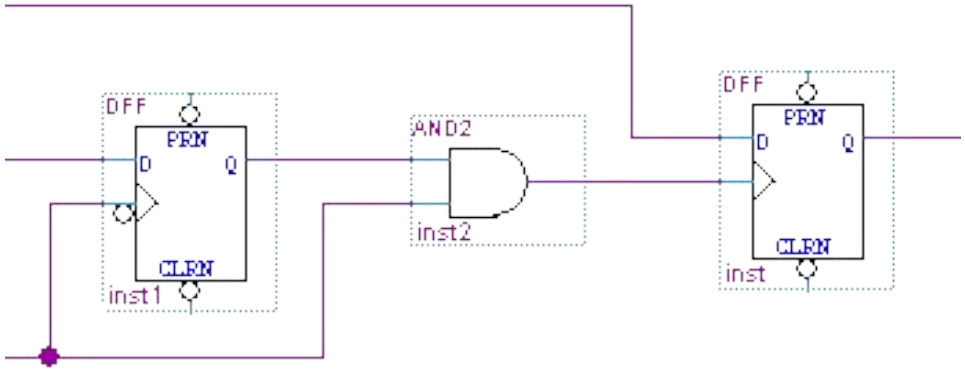
FPGA Gated Clocks

1. The gating logic should be a two-input AND gate or a two-input OR gate.
2. There should be only one input pin that acts as a primary input clock signal to the AND or gate.
3. The non-clock input clock enable signal to the AND or OR gate should be synchronized with a register. The register should be clocked by the same input pin that acts as the primary input clock signal to the AND or OR gate.
4. If the combinational logic uses an AND gate, the clock port of the register that drives the AND gate should be active on the falling edge and the clock port of the register driven by the AND gate should be active on the rising edge.
5. *or*

FPGA Gated Clocks

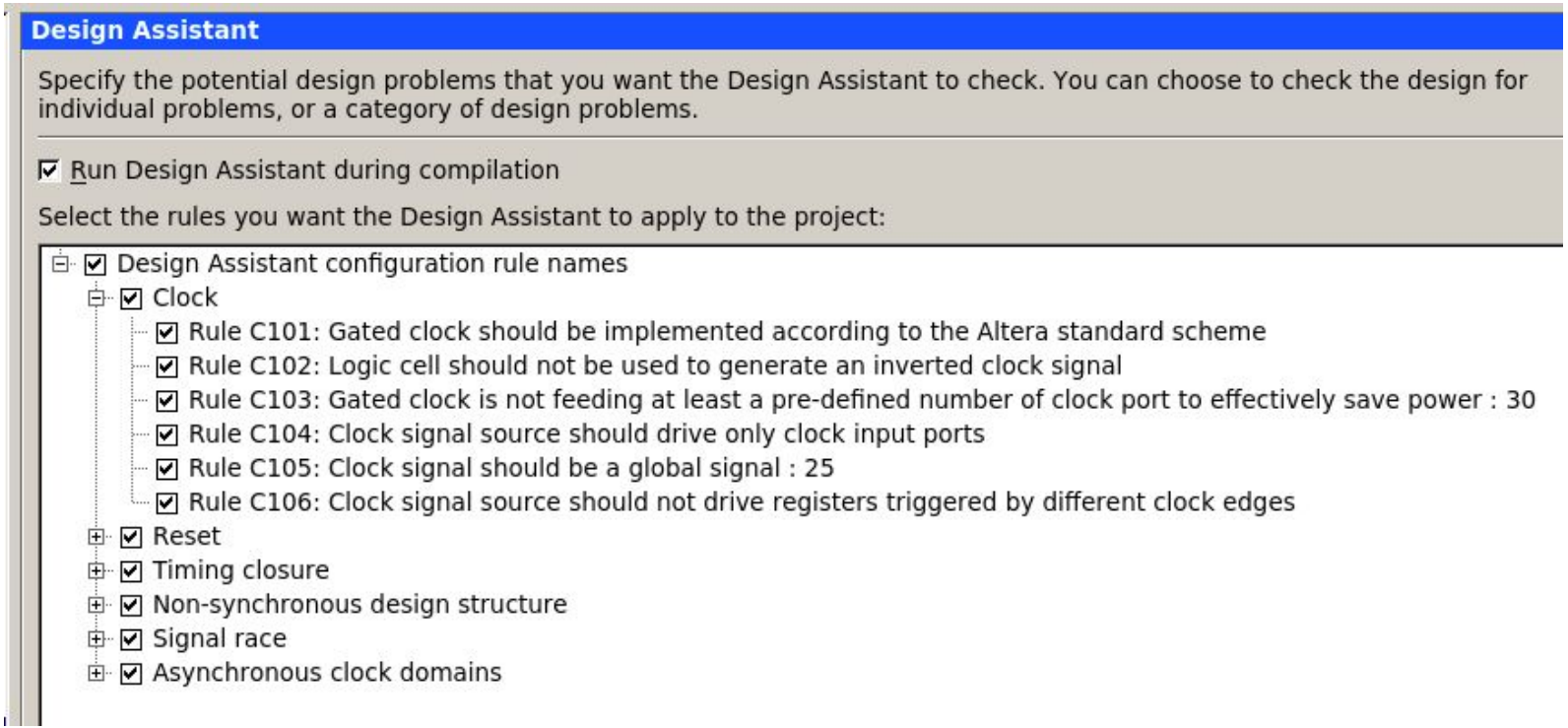
6. If the combinational logic uses an OR gate, the clock port of the register that drives the OR gate should be active on the rising edge and the clock port of the register driven by the OR gate should be active on the falling edge.
7. To reduce the probability of glitches, the `AND` gate should be placed close to the flipflop that generates the gating signal.
8. When specifying timing constraints, Altera recommends enclosing the gating circuitry (`AND` gate and flipflop) in a module. This allows you to specify timing constraints at the module output.

FPGA Gated Clock



NOTE: TWO CLOCK DOMAINS!!!!

Quartus Gated Clocks



Design Techniques to Reduce Power from Glitches: Case Study

Grey Encoding for Counters

Study by Grant Lewis

Binary Encoding

- 3-bit Counter has 14 transitions in 8 cycles
  - 14/8 = 1.75 transitions per cycle
- N-bit Counter has $2(2^N-1)$ transitions
  - Transitions per cycle $\rightarrow$ 2 as N$\rightarrow \infty$

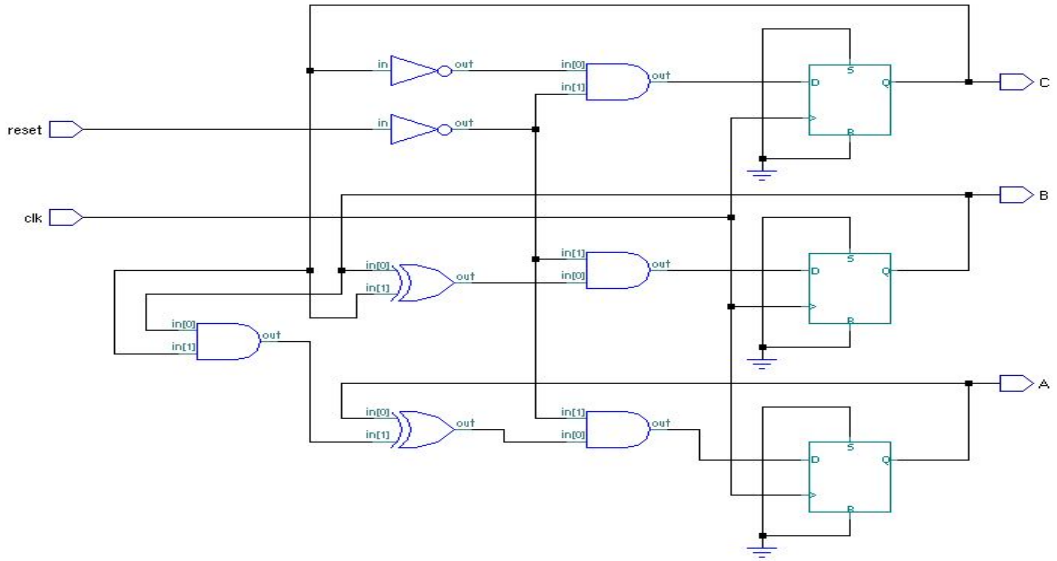
4-bit	1.875 transitions/clock
5-bit	1.9375
6-bit	1.96875
7-bit	1.984375
8-bit	1.9921875
9-bit	1.99609375

000	--
001	1
010	2
011	1
100	3
101	1
110	2
111	1
000	3

Binary Encoding: 3-bit

- A = $a \oplus bc$
- B = $b \oplus c$
- C = $\overline{c}$

- 8 logic gates



Synthesized with Leonardo Spectrum → Standard Cell

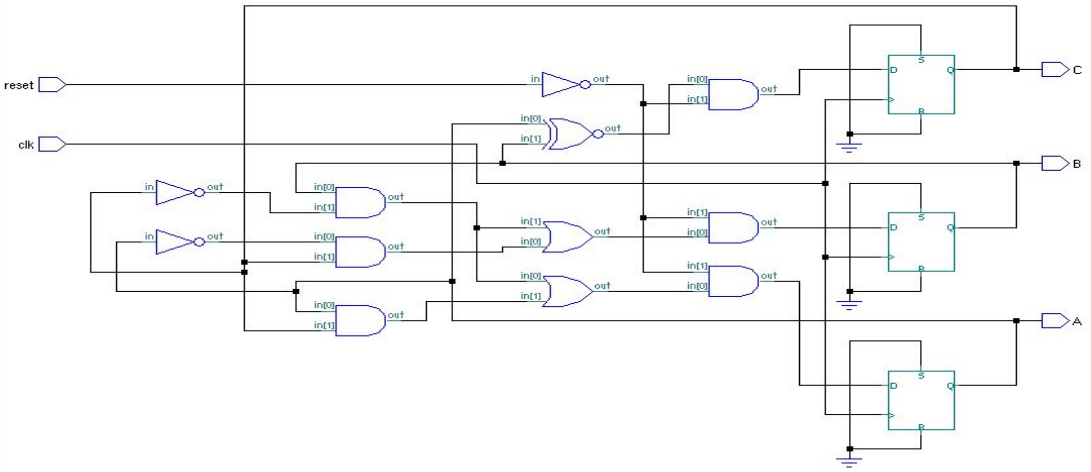
Gray Encoding

- 3-bit counter has 8 transitions in 8 cycles
  - 8/8 = 1 transition per cycle
- N-bit counter has 2^N transitions
  - Transitions per cycle is 1 for any size counter

000	--
001	1
011	1
010	1
110	1
111	1
101	1
100	1
000	1

Grey Encoding

- A = $b\bar{c} + ac$
- B = $\bar{a}c + b\bar{c}$
- C = $\overline{a \oplus b}$



- Adds 4 logic gates (total 12 gates)

Power Analysis (0.18 µm, 1.8V supply)

	Decimal Encoding	Gray Encoding	% Reduction
# of Gates	8	12	-50%
# of Glitches	3122	1618	48.17%
Glitch Power (µW)	1.852307	0.934297	49.56%
Dynamic Power (µW)	1.855127	0.940320	49.31%
Leakage Power (µW)	.914450	.889597	2.72%
Total Power (µW)	2.769578	1.829918	**34.18%**

Conclusion

- Much of dynamic power consumption comes from glitches
  - Can be reduced by path balancing
- Overall, gray counters are more power efficient

Minimizing Transitions on a Bus

```
// Code that resets the bus to default
status after valid gets de-asserted

always@(posedge clk or negedge reset)

begin
   if(!reset)
   data_bus <= 16'b0    ;
   else if (data_bus_valid)
   data_bus <=  data_o ;
   else
   data_bus  <= 16'b0   ;
end
```

```
// Code that holds the bus to its previous
value after valid gets de-asserted

always@(posedge clk or negedge reset)

begin
   if(!reset)
   data_bus <= 16'b0    ;
   else if (data_bus_valid)
   data_bus <=  data_o ;
end
```

The End

Leakage Trends