EECS 151/251 A Discussion 8: Midterm Concept Review

March 8, 2024

Kevin Anderson (he/him)

Berkeley
UNIVERSITY OF CALIFORNIA

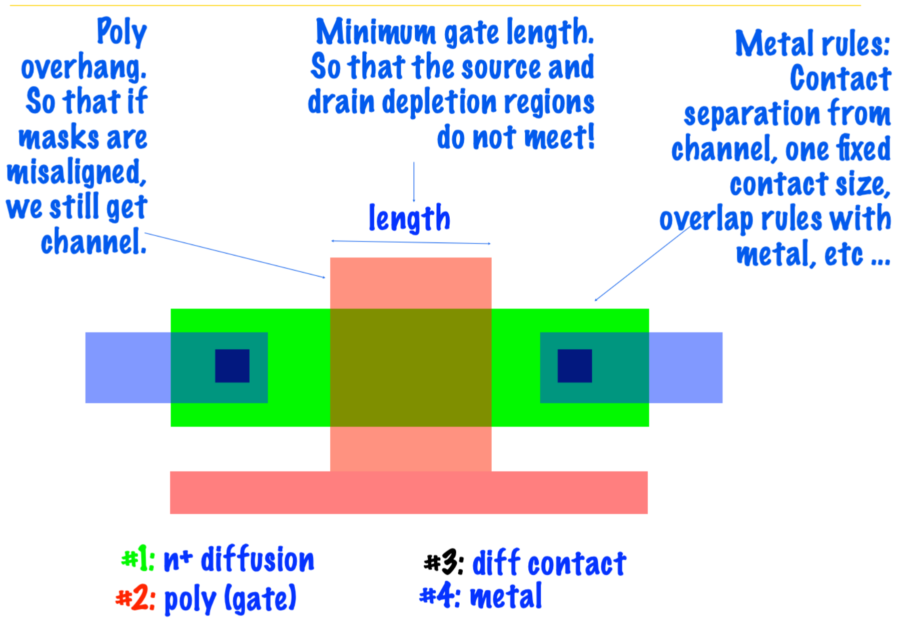
Overview

- GSIs have not seen the midterm so we know nothing! Do not ask us questions!
- This review was created to cover important topics
- Review composed of two parts

Part 1: Circuitry

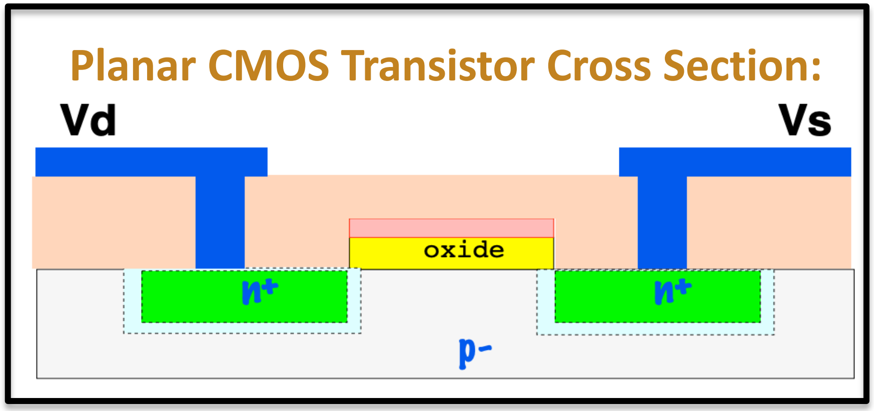
- Transistors (Layout)
- CMOS Static Circuits
- FPGA LUTs and Interconnect
- KMap

Circuit Layouts



Poly overhang. So that if masks are misaligned, we still get channel.
Minimum gate length. So that the source and drain depletion regions do not meet!
length
Metal rules: Contact separation from channel, one fixed contact size, overlap rules with metal, etc ...
#1: n+ diffusion
#2: poly (gate)
#3: diff contact
#4: metal

- A layout is a top-down view of a transistor. It is another perspective of transistor
- Polysilicon represents the gate
- Boxes with or without X represents via

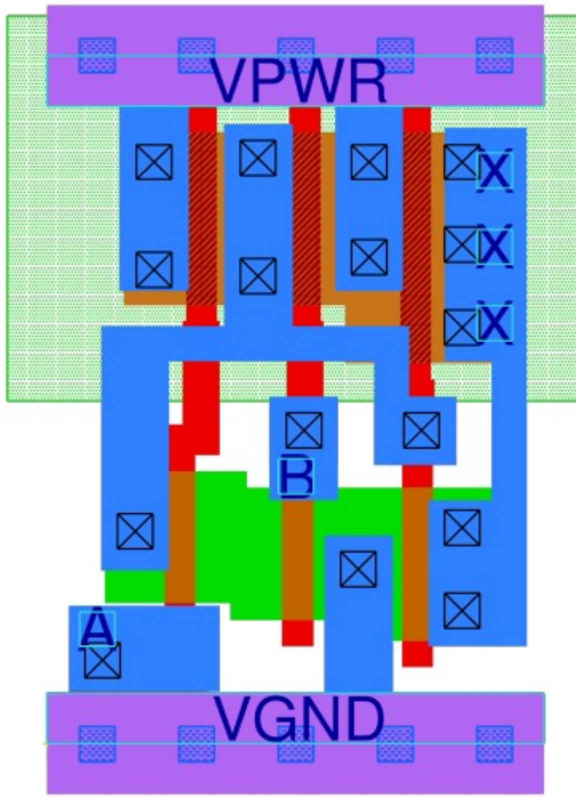


Circuit Layouts

Steps:

1. Identify transistors
2. Separate PUN and PDN
3. Look at transistors connected to rails
4. Find shared diffusion regions and metal with via to discover inter-transistor connections

This is a AND gate



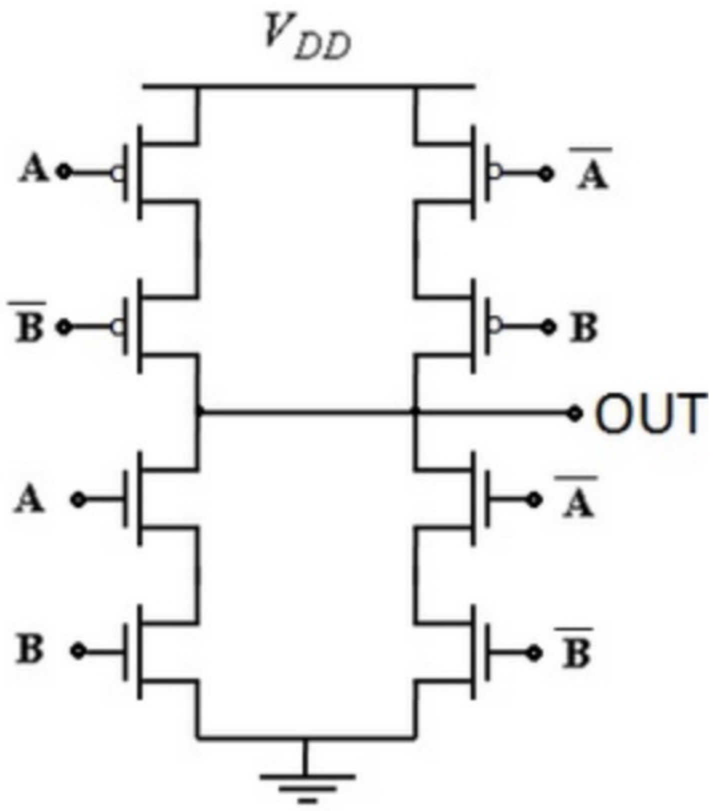
Static CMOS Circuits

- PUN and PDN with single output
- Transistors in series represents a logical AND
- Transistor in parallel represents a logical OR
- PUN and PDN are duals; Boolean expression is the negation of the other
- Also simplify Boolean expression before drawing circuit

Static CMOS Circuits

- Let's write the Boolean expression for this circuit:

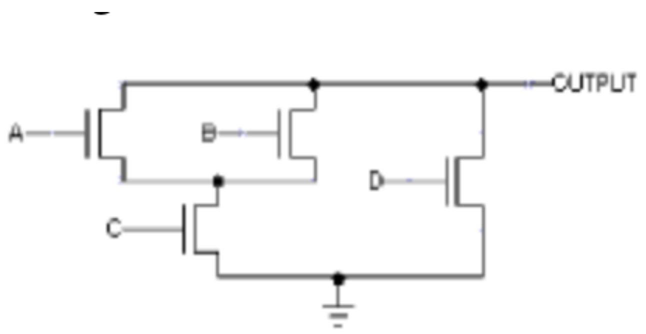
- On your own, draw the circuit for an XNOR:



Berkeley
UNIVERSITY OF CALIFORNIA

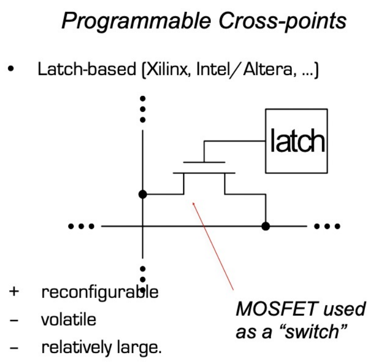
Static CMOS Circuits

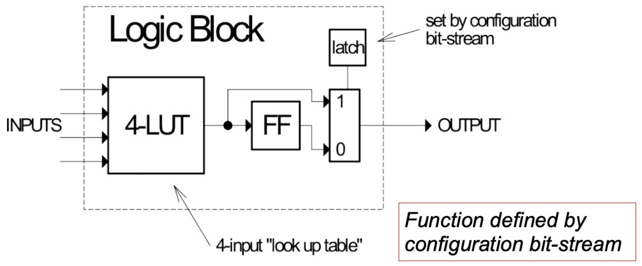
- What part of the static CMOS circuit is this?
- Draw the other half of the CMOS circuit

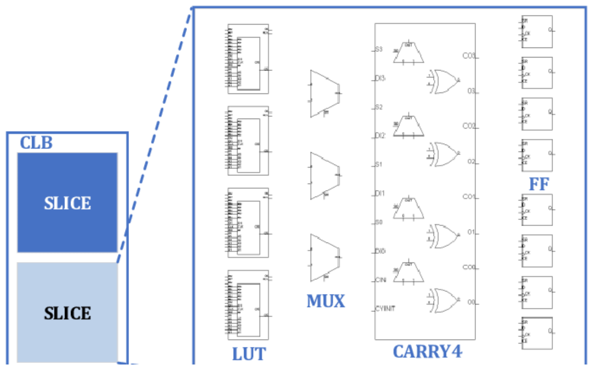


FPGA

- FPGAs is a configurable HW platform (i.e. not fixed function)
  - The configurability comes from the interconnect and LUT
- Xilinx Logic Element Hierarchy: LUTs -> Slice -> CLBs
- Interconnect has latches to connect CLBs together
- **There are no gates in an FPGA**
- How many logical expressions can a LUT-X hold?


Programmable Cross-points
Latch-based (Xilinx, Intel/Altera, ...)
latch
+ reconfigurable
– volatile
– relatively large.
MOSFET used as a "switch"
Simplified interconnect latch


Logic Block
latch
set by configuration bit-stream
INPUTS
4-LUT
FF
1
0
OUTPUT
4-input "look up table"
Function defined by configuration bit-stream
Simplified LUT-4



Xilinx Series 7 CLB architecture

Karnaugh Map

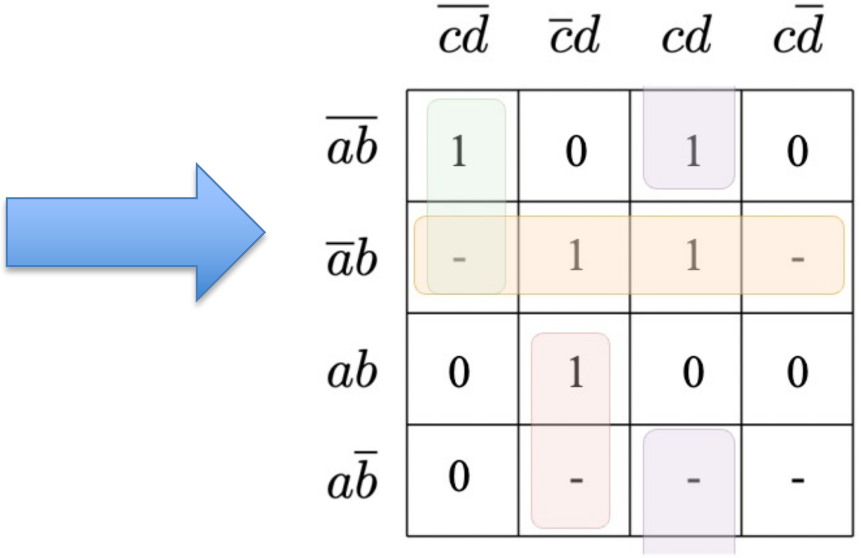
- Write the **optimal** SOP and POS:

	$\overline{cd}$	$\bar{c}d$	cd	$c\bar{d}$
$\overline{ab}$	1	0	1	0
$\bar{a}b$	-	1	1	-
ab	0	1	0	0
$a\bar{b}$	0	-	-	-

Karnaugh Map

- **All KMaps must be Gray coded**
- Write the **optimal** SOP and POS:

	$\overline{c}\overline{d}$	$\overline{c}d$	cd	$c\overline{d}$
$\overline{a}\overline{b}$	1	0	1	0
$\overline{a}b$	-	1	1	-
ab	0	1	0	0
$a\overline{b}$	0	-	-	-



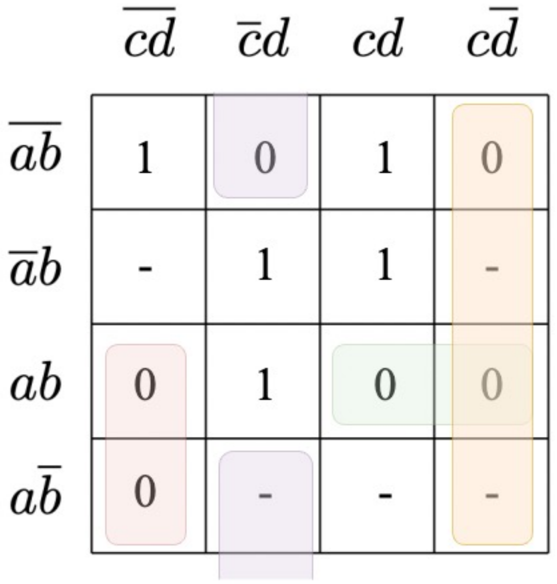
	$\overline{c}\overline{d}$	$\overline{c}d$	cd	$c\overline{d}$
$\overline{a}\overline{b}$	1	0	1	0
$\overline{a}b$	-	1	1	-
ab	0	1	0	0
$a\overline{b}$	0	-	-	-

SOP: $\overline{A}B + \overline{B}CD + A\overline{C}D + \overline{A}\overline{C}\overline{D}$

Karnaugh Map

- **All KMaps must be Gray coded**
- Write the **optimal** SOP and POS:

	$\overline{cd}$	$\bar{c}d$	cd	$c\bar{d}$
$\overline{ab}$	1	0	1	0
$\bar{a}b$	-	1	1	-
ab	0	1	0	0
$a\bar{b}$	0	-	-	-



	$\overline{cd}$	$\bar{c}d$	cd	$c\bar{d}$
$\overline{ab}$	1	0	1	0
$\bar{a}b$	-	1	1	-
ab	0	1	0	0
$a\bar{b}$	0	-	-	-

POS: $(\overline{A}+\overline{B}+\overline{C})(\overline{C}+D)(B+C+\overline{D})(\overline{A}+C+D)$

To-Know

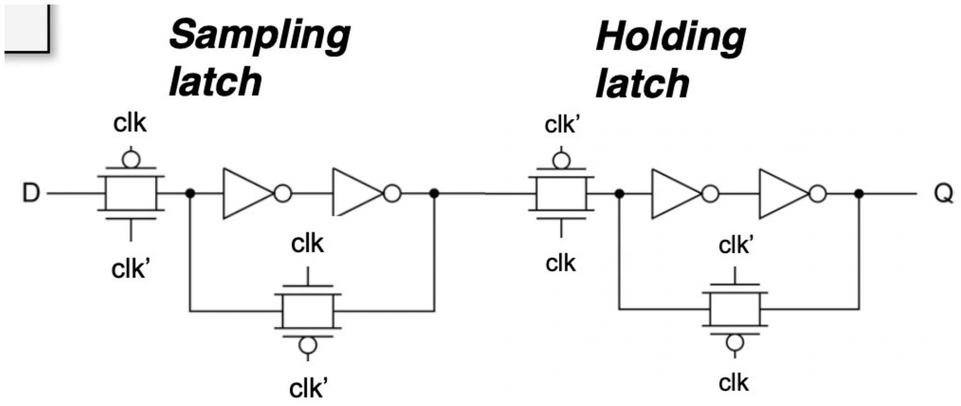
- Boolean Algebra
- Cost Analysis (NRE and Recurring Cost)
- NMOS and PMOS
- CMOS Fabrication Process
- CMOS Circuits for Common Logic Gates (**NAND** and **INV**)
- FPGA vs ASIC
- PMOS weak pull down, NMOS weak pull up
- Timing diagram

Part 2: Verilog and Blocks

- Positive-Edge FF
- Finite State Machines
- Verilog Basics
- Verilog Advanced

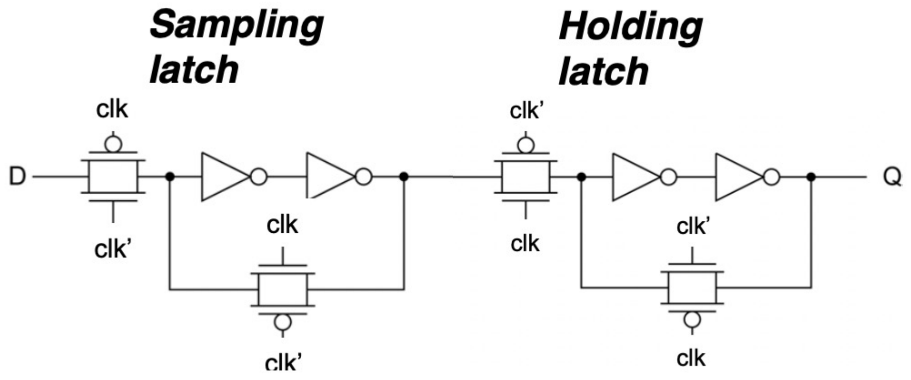
Positive Edge Flip-Flop

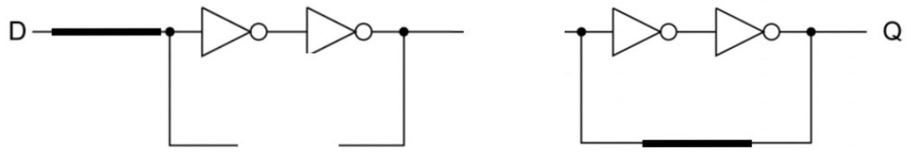
- Flip–flop is composed of two latches
- Transmission gates orchestrate exchange between latches



Positive Edge Flip-Flop

- What is the clock state represented by the figure below?



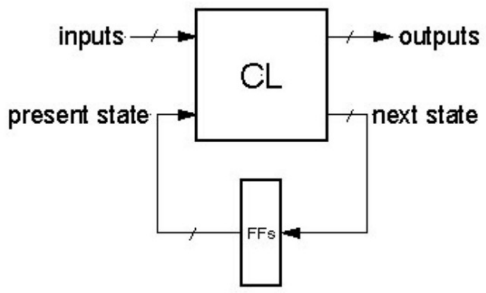


Ready to capture new value on clock edge

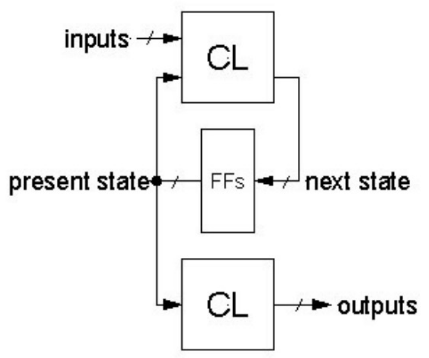
Is outputing last previous captured value.

Finite State Machines

- Mealy vs Moore
- One-hot vs Binary vs BCD Encoding
  - BCD not really used for FSMs, but good to know the encoding
  - Gray Coding and Hamming Distance (for Binary and BCD only)
- State Diagrams
  - Idle state
  - Transitions without conditions taken on following cycle always. Transitions with condition taken on cycle when condition is met
  - Outputs that change on transition are Mealy. Outputs that change within state are Moore. If outputs change on both, then it's a mixture of both



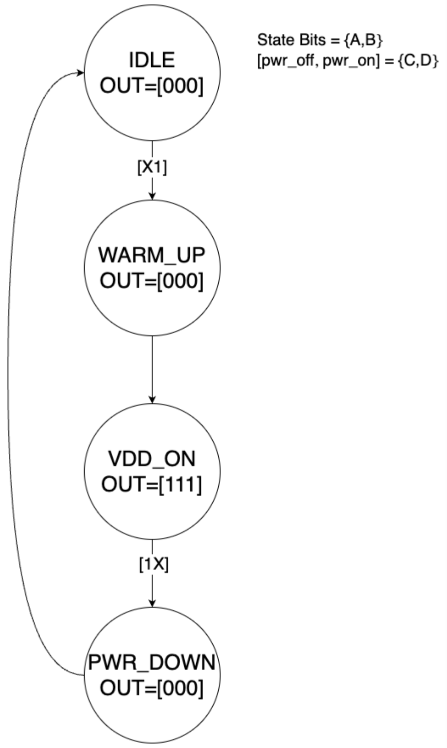
Conceptual diagram of Mealy



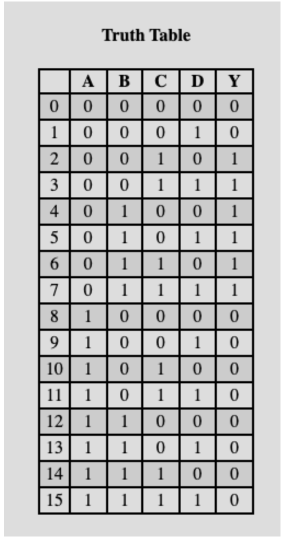
Conceptual diagram of Moore

Finite State Machines

- Verilog FSM from discussion 5



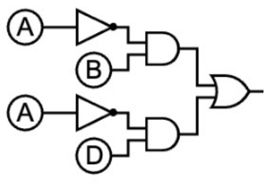
State LSb


Truth Table

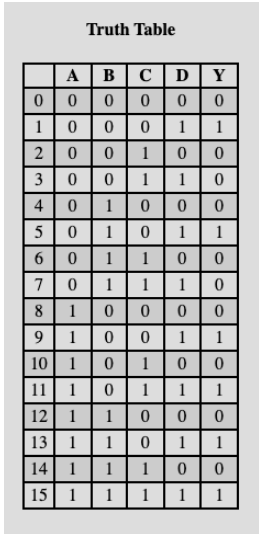
	A	B	C	D	Y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

SOP: $\overline{A}D + \overline{A}B$

CD \ AB	00	01	11	10
00	0	1	0	0
01	1	1	0	0
11	1	1	0	0
10	0	1	0	0



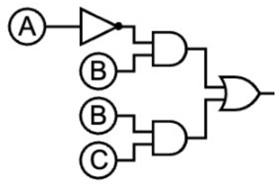
State MSb


Truth Table

	A	B	C	D	Y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

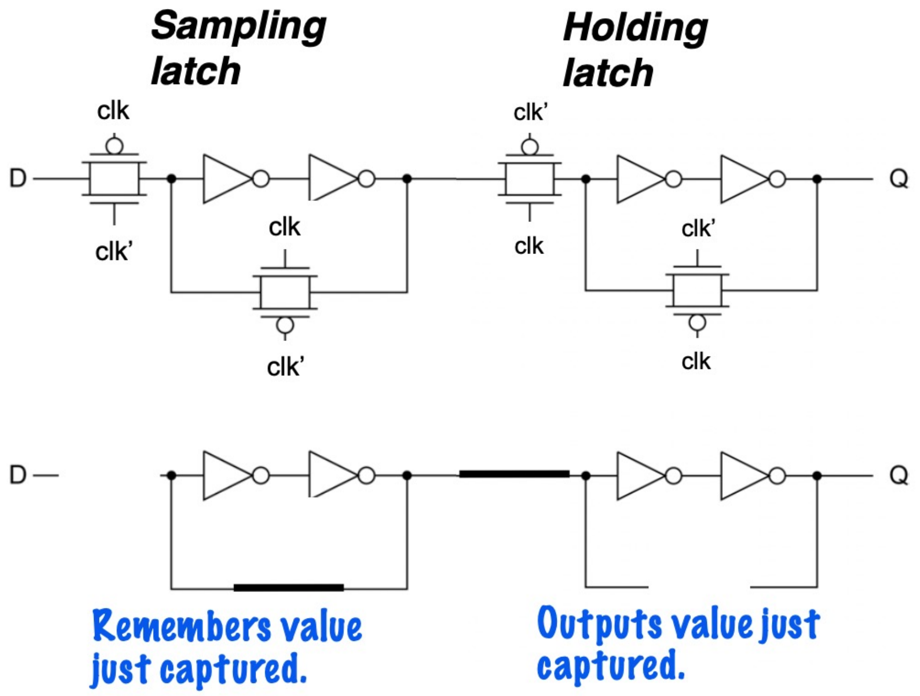
SOP: $BC + \overline{A}B$

CD \ AB	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	0	1	1	0
10	0	1	1	0



Positive Edge Flip-Flop

- What is the clock state represented by the figure below?



Verilog Basics

- **No register inference!**
- wire used in continuous assignment
- reg used in procedural assignment
- case statement for FSMs
- Use named ports for instantiation
- Width matters!!
  - e.x. `wire [1:0] tmp; assign tmp = 4;`

Verilog Advanced

- FSMs must be written in style 2 (slide 37 Verilog Review) because no register inference
- Parameterized modules are generators
- `generate` creates copies of hardware that run in parallel
  - Types: `generate` for, `generate` if, `generate` case
  - Loops ≠ `generate`

Example:

```
module naryand (in, out);
  parameter N = 1;

  input [N-1:0] in;
  output out;

  wire [N-1:0] tmp;
  buf(tmp[0], in[0]);
  buf(out, tmp[N-1]);
  genvar i;
  generate
    for(i = 1; i < N; i = i + 1) begin : ands
      and(tmp[i], in[i], tmp[i-1]);
    end
  endgenerate
endmodule

// Instantiation of generator
wire [4:0] f;
wire g;
naryand #(.N(5)) and5(.in(f), .out(g));
```

Berkeley
UNIVERSITY OF CALIFORNIA

To-Know

- Transmission Gates
- Tri-state buffer vs Tri-state inverter
- Create state diagram from word problem
- Continuous Assignments and Procedural Assignments
- Ternary Operator
- Labelling state diagram
- Shift Register in Verilog