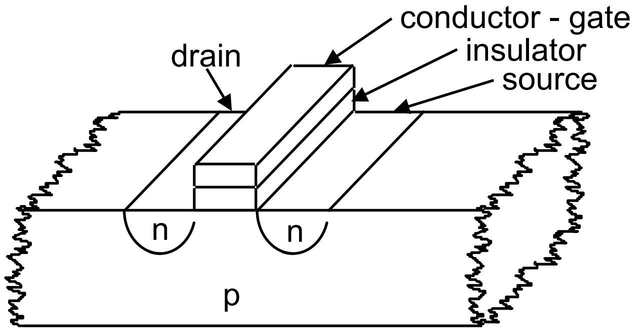
How do FPGAs work?

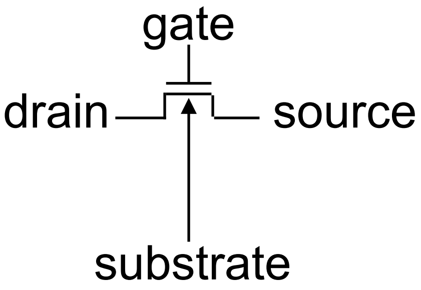
- Basic CMOS devices
- Building gates from transistors
- Building wiring from transistors

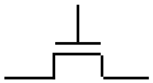
CMOS Transistors

- All circuit elements built from transistors of two <u>C</u>omplementary types

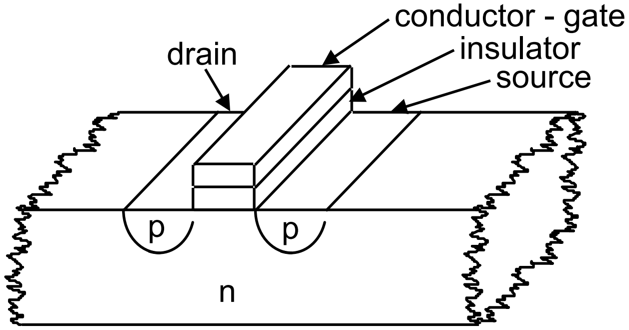
n-type

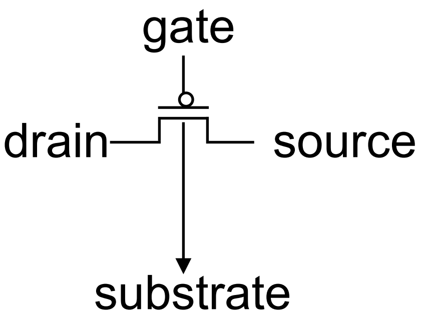


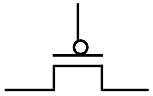




p-type





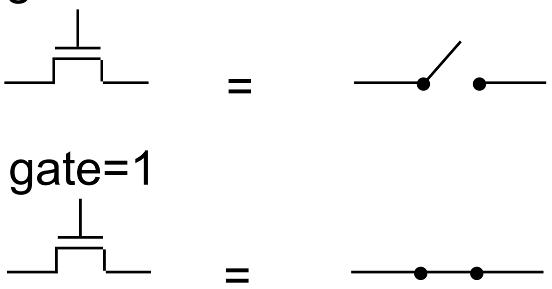


Transistor Switches

TRUE = 1 = 2.3 to 3 Volts
FALSE = 0 = 0 to 0.7 Volts

1.6v is the “noise margin”

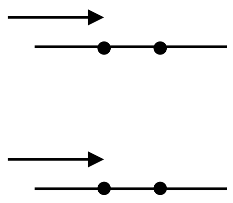
n-type



gate=0 = open switch

gate=1 = closed switch

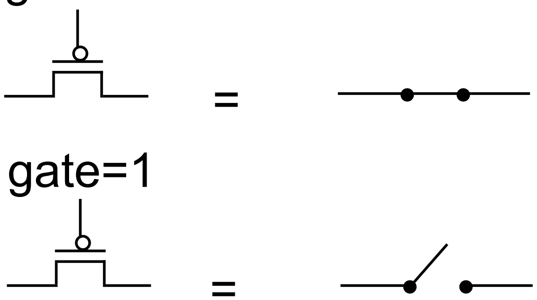
however:



0 good 0

1 <u>poor 1</u>

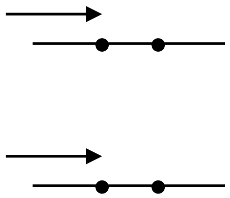
p-type



gate=0 = closed switch

gate=1 = open switch

however:

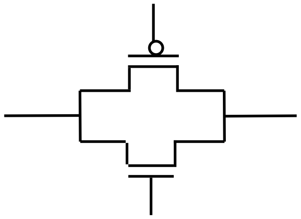


0 <u>poor 0</u>

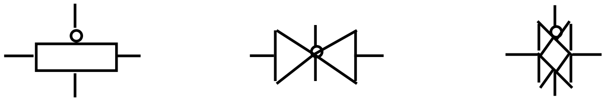
1 good 1

How do we make a better switch

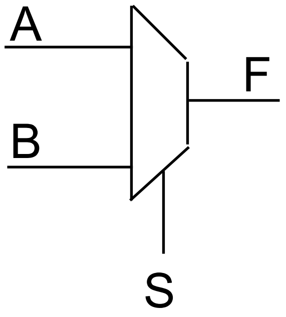
- Transmission gate or pass gate
  - Use a switch of each type in parallel



  - Alternative symbols for pass gates

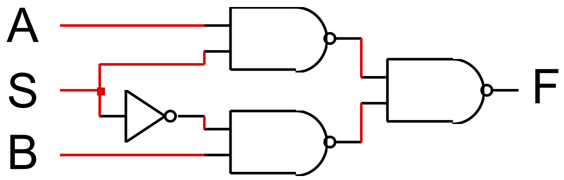


Muliplexors

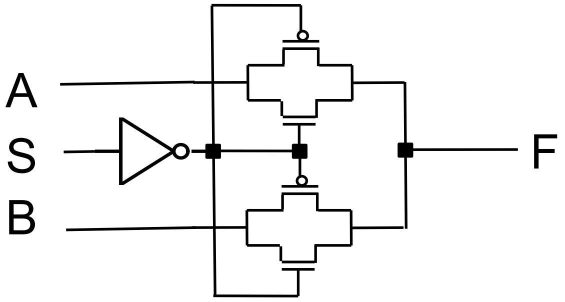


S	A	B	F
0	0	X	0
0	1	X	1
1	X	0	0
1	X	1	1

- How do we build a 2:1 Mux?
  - Using logic gates



  - Using pass gates (output ties two signals together?)

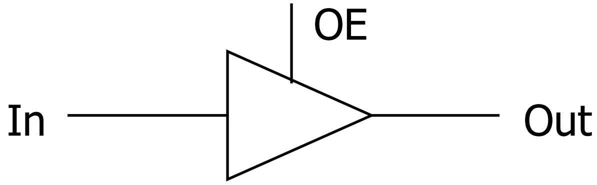


Tri-state logic

- Multiple outputs can be tied to one wire, but only if they are guaranteed to never conflict (driven simultaneously to 0 and 1)
- The “Z” state for a signal – when nothing is driving it
  and it is just left “floating”
- Something has to “pull it up” or “pull it down” to set a value
  - This can be gate outputs (but have to ensure they never conflict)
  - Or it can be pass outputs (but we have to ensure they never conflict)
- The difference is that pass gates can just “disconnect” by turning off – gates can’t do that

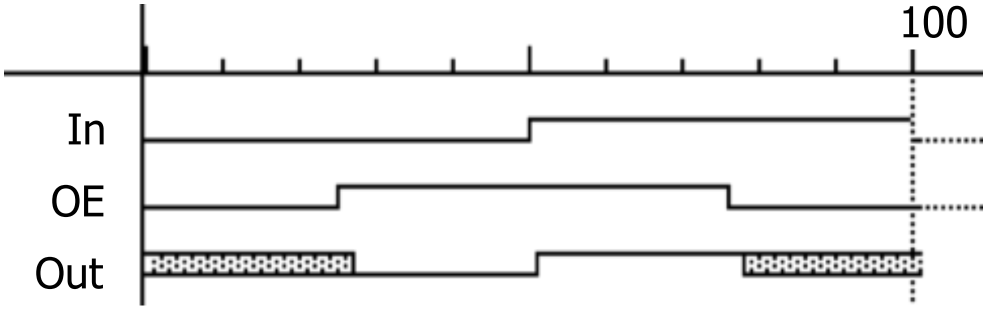
Tri-state gates

- The third value
  - logic values: “0”, “1”
  - don't care not a value: “X” (must be 0 or 1 in real circuit!)
  - third value or state: “Z” — floating, no connection
- Tri-state gates
  - additional input – output enable (OE)
  - output values are 0, 1, and Z
  - when OE is high, the gate functions normally
  - when OE is low, the gate is disconnected from wire at output
  - allows more than one gate to be connected to the same output wire
    - as long as only one has its output enabled at any one time



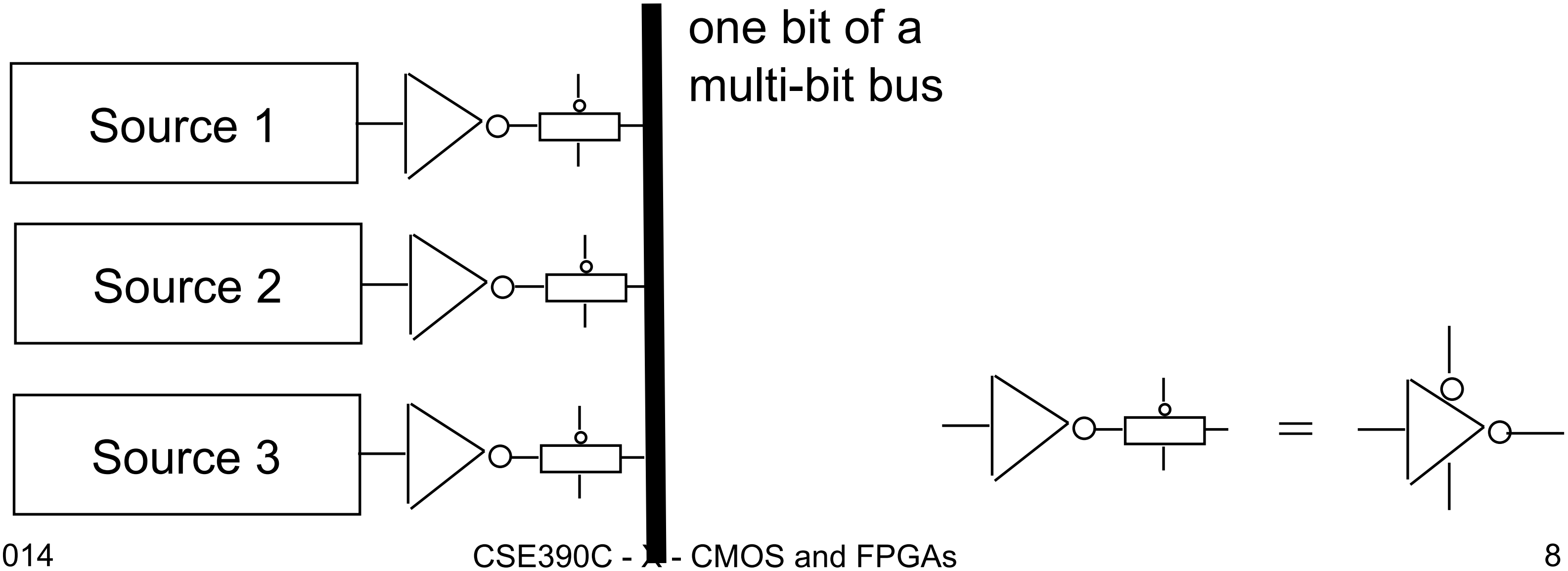
non-inverting
tri-state
buffer

In	OE	Out
X	0	Z
0	1	0
1	1	1



Where is tristate used?

- We just saw one example with a multiplexer
- Larger scale multiplexer is when we can connect many more signals onto a “bus” – a group of wires that are logically related – e.g., the output of a register such as number
- Only one pass gate can be enabled at a time
- A typical “bus” driver



Source 1

Source 2

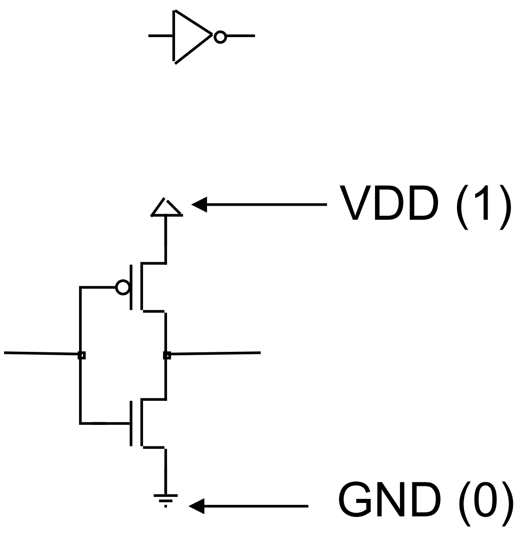
Source 3

one bit of a multi-bit bus

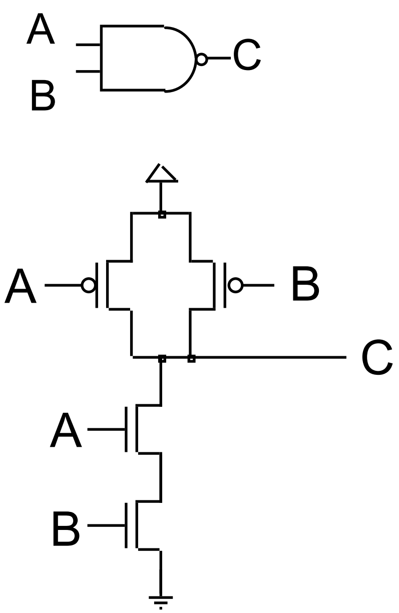
=

Basic Gates

INV

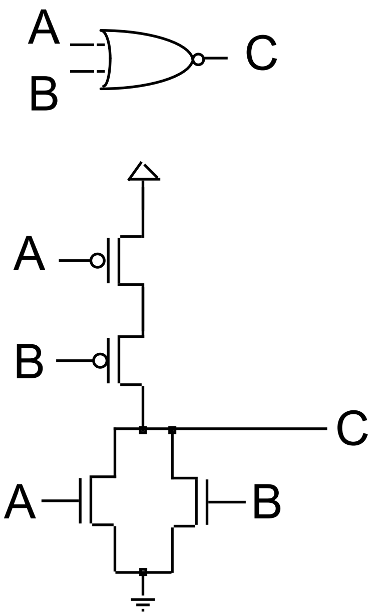


NAND



		A	
		0	1
B	0	1	1
	1	1	0

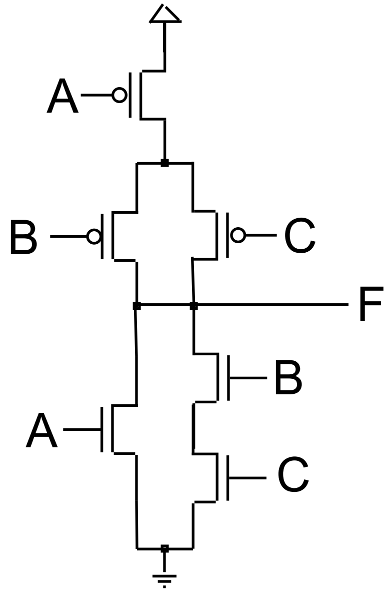
NOR



		A	
		0	1
B	0	1	0
	1	0	0

More complex or compound gates

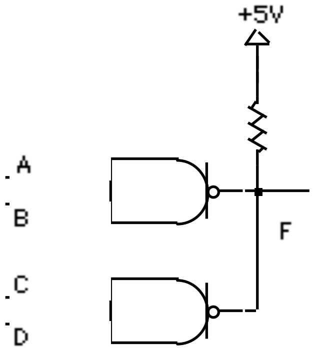
- A complex boolean function can be built from basic gates (inverter, NAND, NOR)
- Alternatively, for example, F = A’(B’ + C’) = (A + BC)’ can be realized this way:



Open-collector gates and wired-AND

- Open collector: another way to connect gate outputs to the same wire
  - gate only has the ability to pull its output low
  - it cannot actively drive the wire high (default – pulled high through resistor)
- Wired-AND can be implemented with open collector logic
  - if A and B are "1", output is actively pulled low
  - if C and D are "1", output is actively pulled low
  - if one gate output is low and the other high, then low wins
  - if both gate outputs are "1", the wire value "floats", pulled high by resistor
    - low to high transition usually slower than it would have been with a gate pulling high
  - hence, the two NAND functions are ANDed together

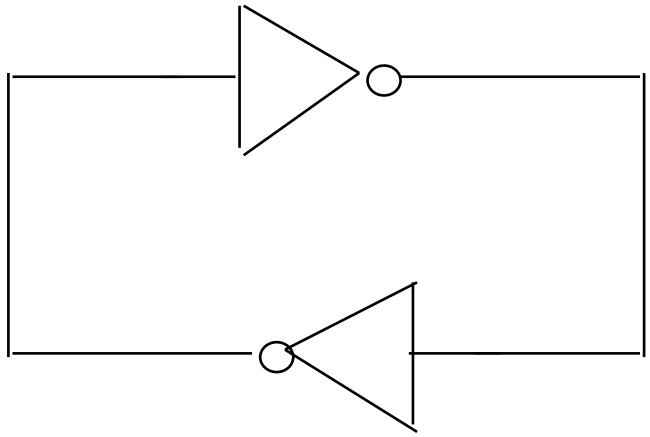
open-collector
NAND gates

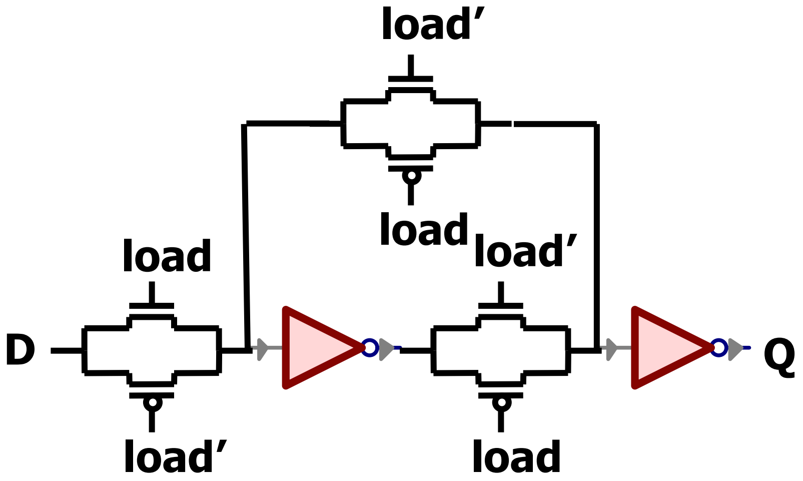


with ouputs wired together
using "wired-AND"
to form (AB)'(CD)'

Static D FF using pass gates

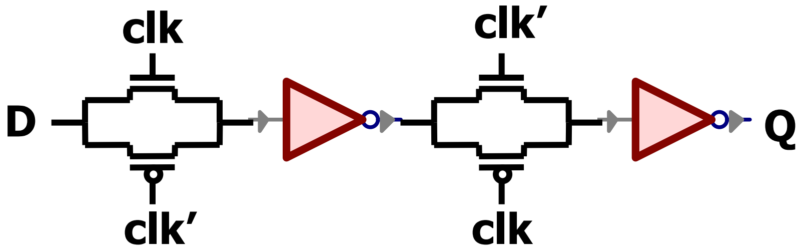
- A pair of inverters can hold a value, static memory
- A value can be read, but how written?



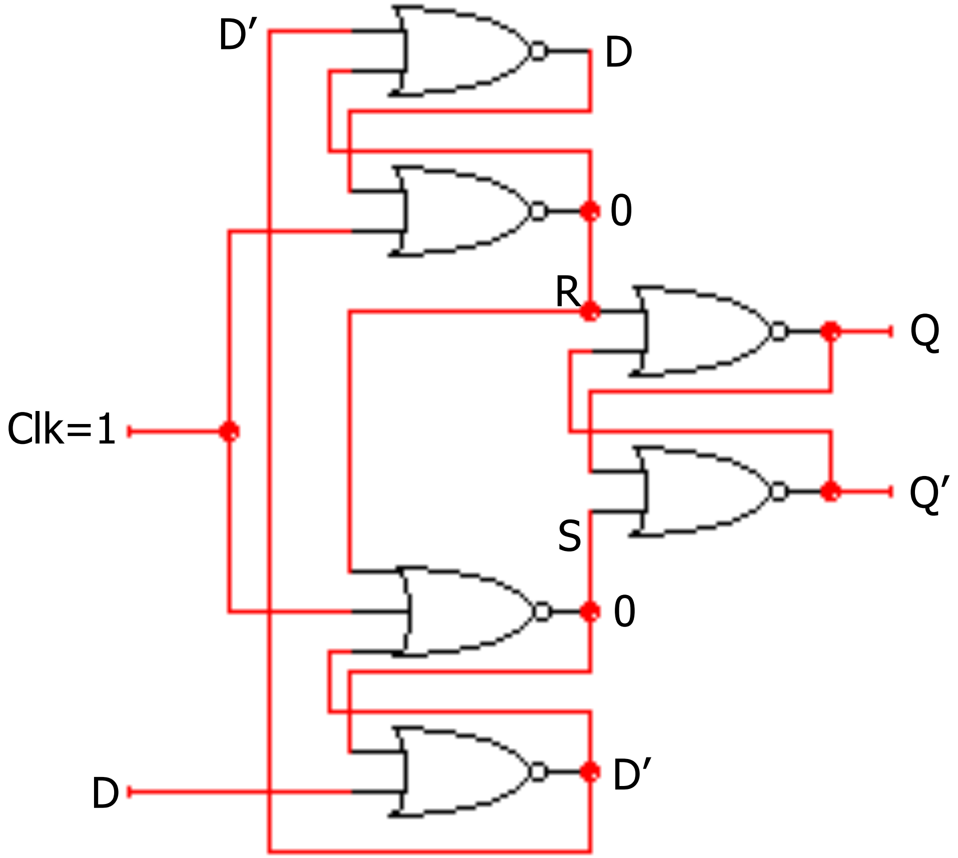


Dynamic D FF

- Remove feedback loop
- Must guarantee clock is frequent enough that charges stored on inputs of inverters does NOT lead away

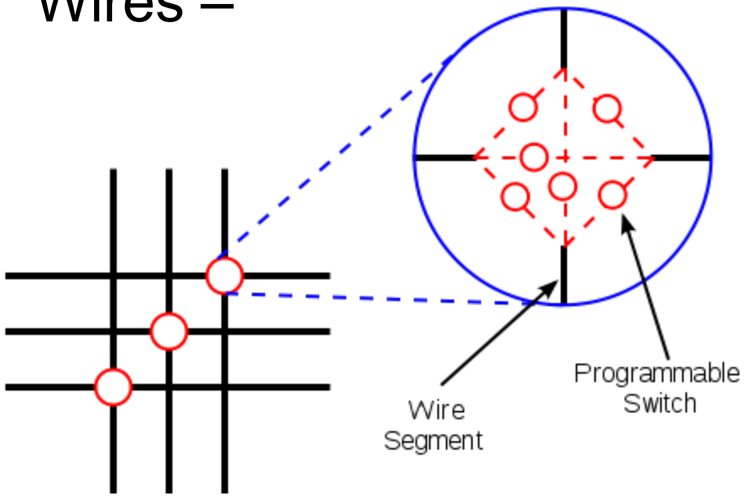


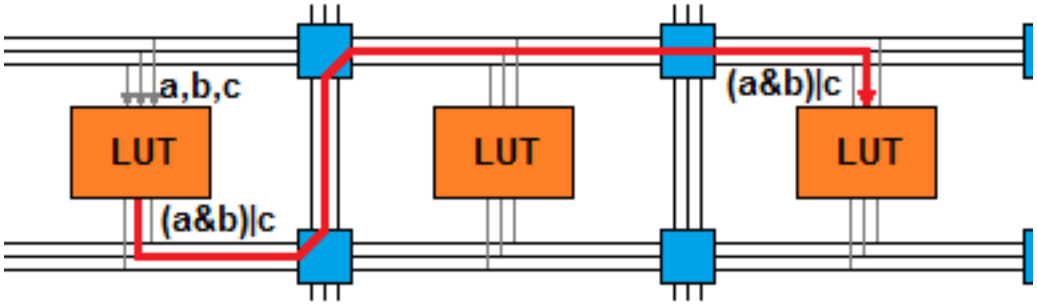
D FF using logic gates



How are FPGAs built?

- Logic - lookup tables – multiplexers (gates or pass gates)
- D-FF – static memory
- Wires –





- Programming – static memory that can be loaded with bit patterns that control inputs to LUTs and control wire connections