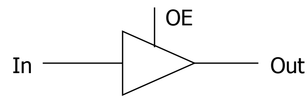
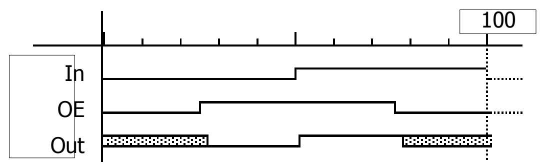
Tri-state gates

- The third value
  - logic values: “0”, “1”
  - don't care: “X” (must be 0 or 1 in real circuit!)
  - third value or state: “Z” — high impedance, infinite R, 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 (otherwise, sparks could fly)



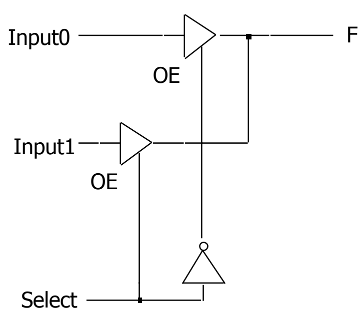
non-inverting tri-state buffer

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



Tri-state and multiplexing

- When using tri-state logic
  - (1) make sure never more than one "driver" for a wire at any one time (pulling high and low at the same time can severely damage circuits)
  - (2) make sure to only use value on wire when its being driven (using a floating value may cause failures)
- Using tri-state gates to implement an economical multiplexer



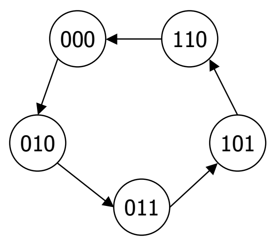
when Select is high
Input1 is connected to F

when Select is low
Input0 is connected to F

this is essentially a 2:1 mux

More complex counter example

- Complex counter
  - repeats 5 states in sequence
  - not a binary number representation
- Step 1: derive the state transition diagram
  - count sequence: 000, 010, 011, 101, 110
- Step 2: derive the state transition table from the state transition diagram



Present State			Next State		
C	B	A	C+	B+	A+
0	0	0	0	1	0
0	0	1	–	–	–
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	–	–	–
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	–	–	–

note the don't care conditions that arise from the unused state codes

More complex counter example (cont'd)

- Step 3: K-maps for next state functions

C+ (C over the last two columns, B over the middle two columns)

	0	0	0	X
A	X	1	X	1

B+ (C over the last two columns, B over the middle two columns)

	1	1	0	X
A	X	0	X	1

A+ (C over the last two columns, B over the middle two columns)

	0	1	0	X
A	X	1	X	0

C+ := A

B+ := B' + A'C'

A+ := BC'

Comparison

⌘ T FF: 5 gates, 10 literals, 15 wires

⌘ RS FF: 3 , 5 , 12

⌘ JK FF: 2 , 4 , 9

⌘ D FF: 3, 5, 9

Summary :

⌘ JK FF are most gate and literal efficient

⌘ T FF are good mainly for straight forward counters

⌘ D FF: Donot need the next state mapping stage. And **less wiring complexity**

⌘ JK is **always** better than RS