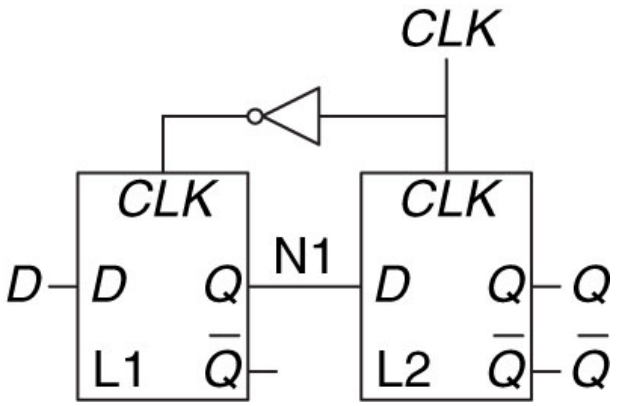
Timing Sequential Logic

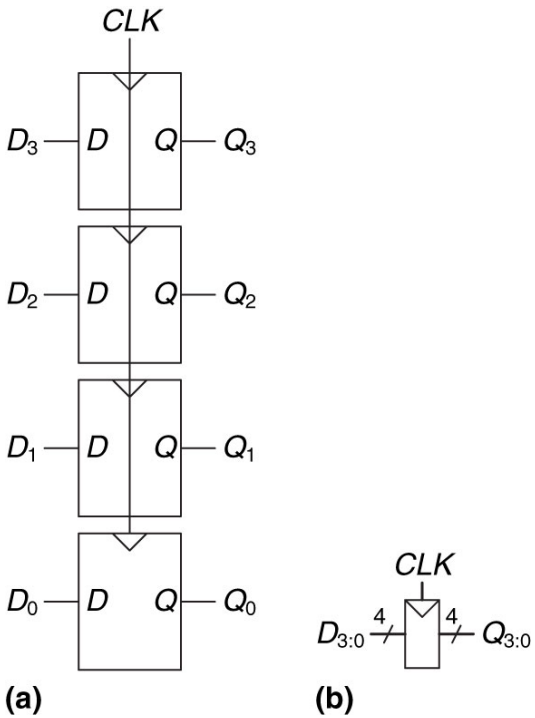
Based on slides by Jared Moore

Recall: D Flip-Flop

A D flip-flop copies D to Q on the rising edge of the clock, and remembers its state at all other times.



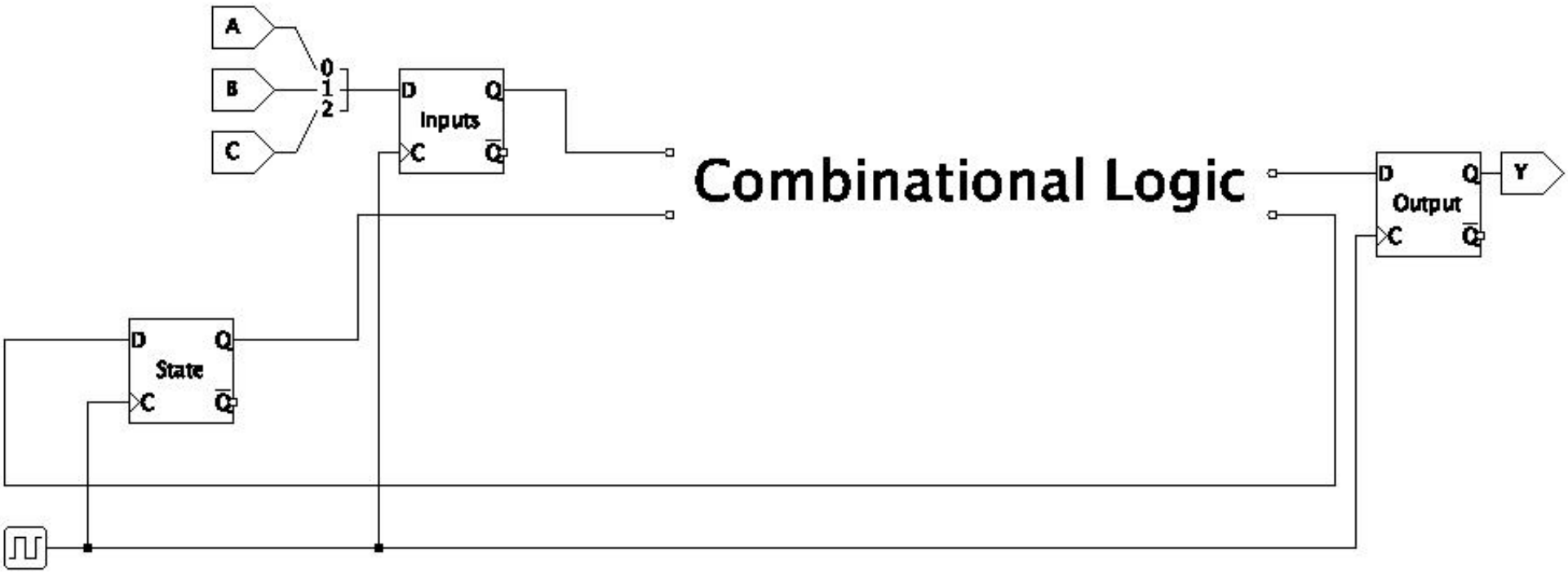
Recall: Register

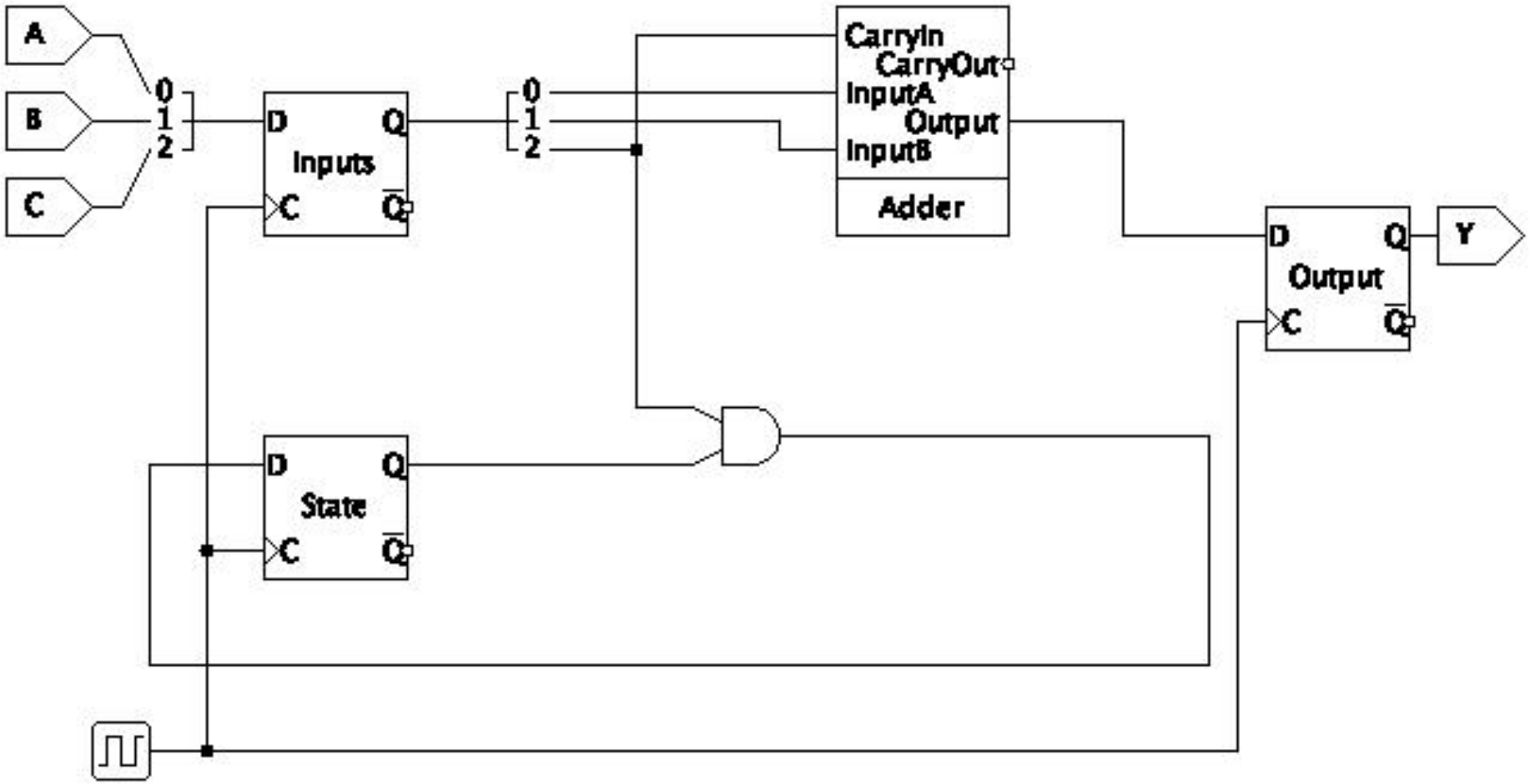


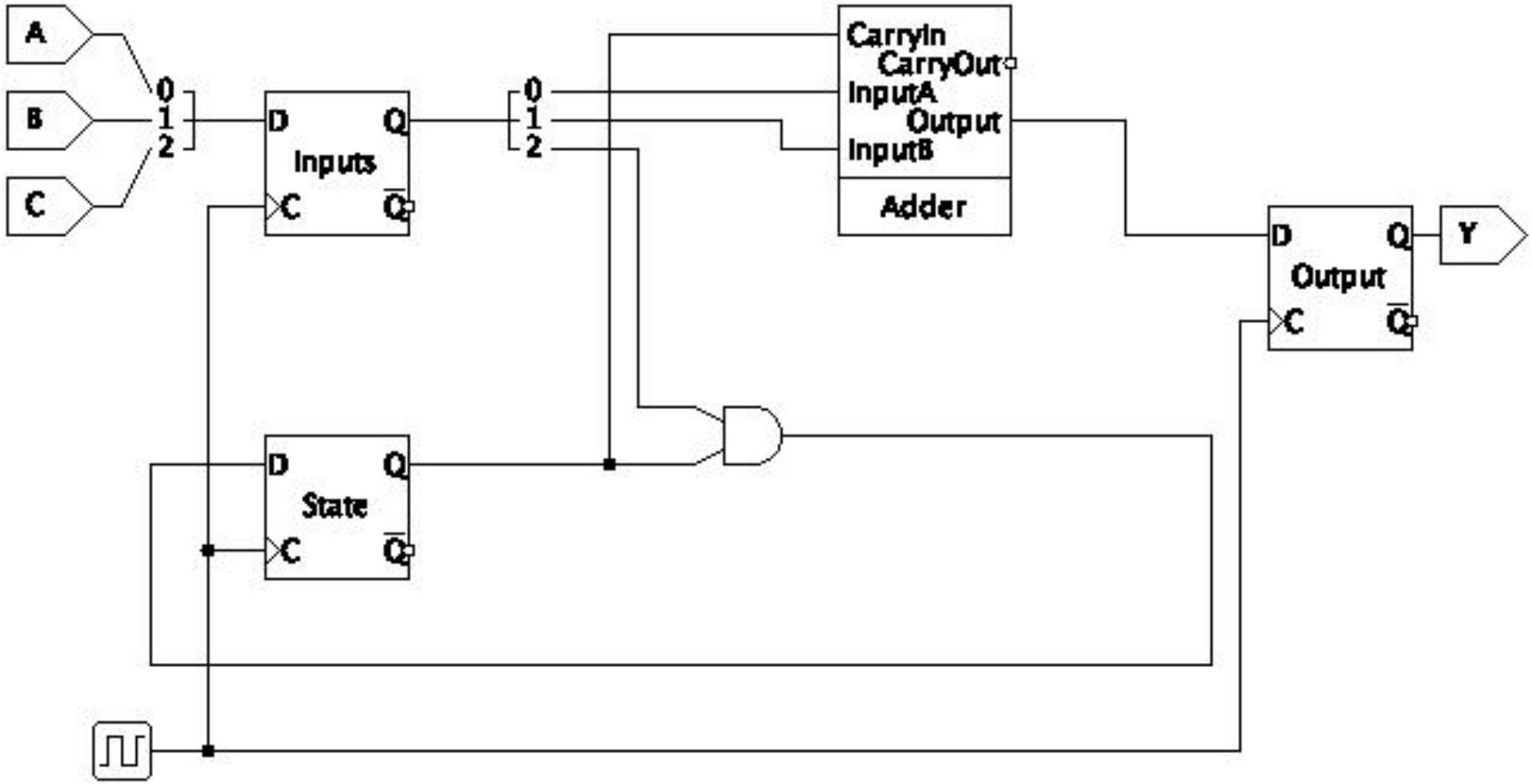
Combining multiple D flip-flops together creates a basic register.

When CLK is asserted, the flip-flops are updated simultaneously.

Basic memory!


A
B
C
0
1
2
D
Q
Inputs
C
Q
Combinational Logic
D
Q
Output
C
Q
Y
D
Q
State
C
Q

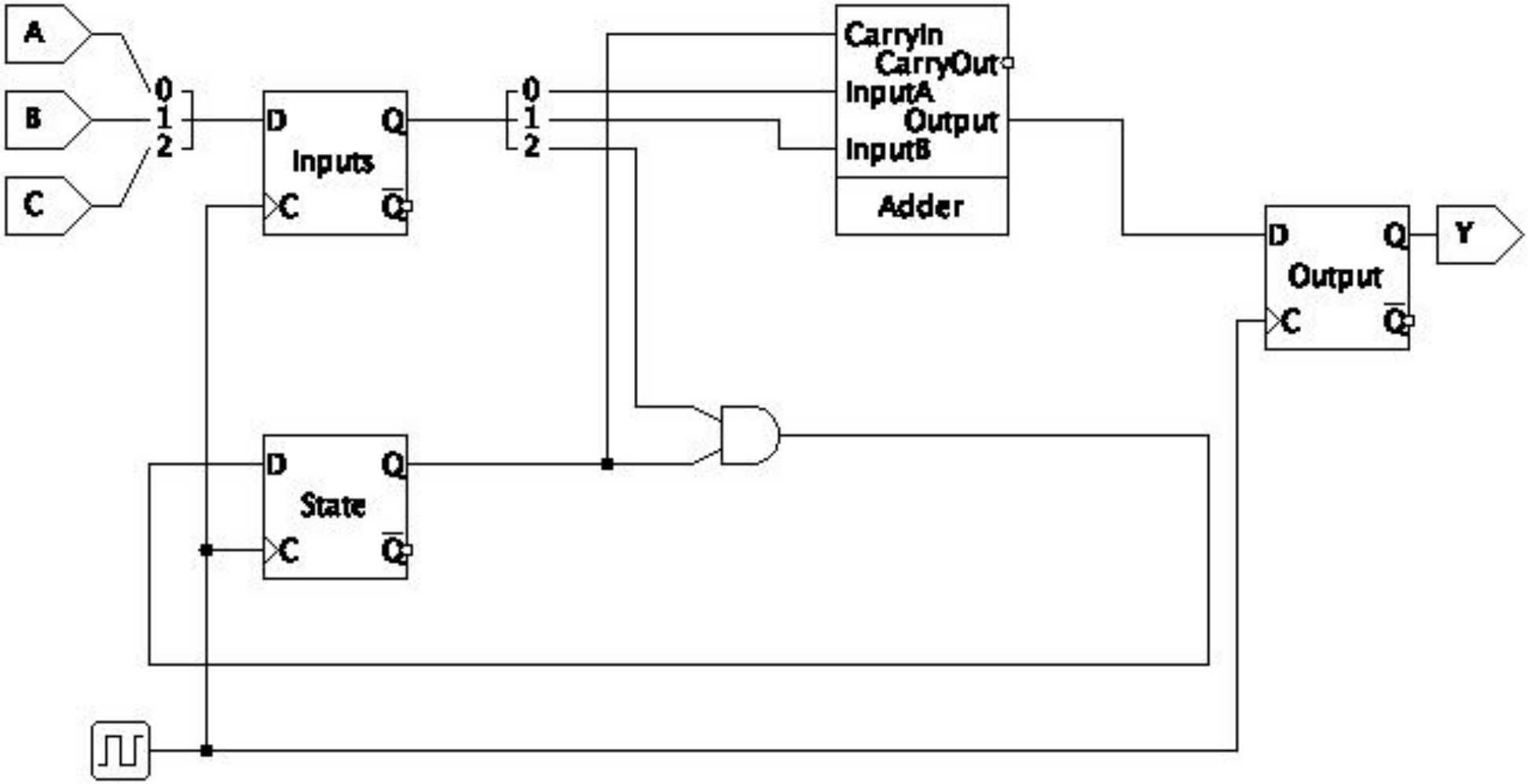

A
B
C
0
1
2
D
Q
Inputs
C
Q
0
1
2
CarryIn
CarryOut
InputA
Output
InputB
Adder
D
Q
Output
C
Q
Y
D
Q
State
C
Q


A
B
C
0
1
2
D
Q
Inputs
C
Q
0
1
2
CarryIn
CarryOut
InputA
Output
InputB
Adder
D
Q
Output
C
Q
Y
D
Q
State
C
Q

Timing Sequential Logic

Maximum delay of combinational logic dictates how many gates can be on a critical path of a high-speed circuit.

- Must finish before a clock cycle!


A
B
C
0
1
2
D
Q
Inputs
C
Q
0
1
2
CarryIn
CarryOut
InputA
Output
InputB
Adder
D
Q
Output
C
Q
Y
D
Q
State
C
Q

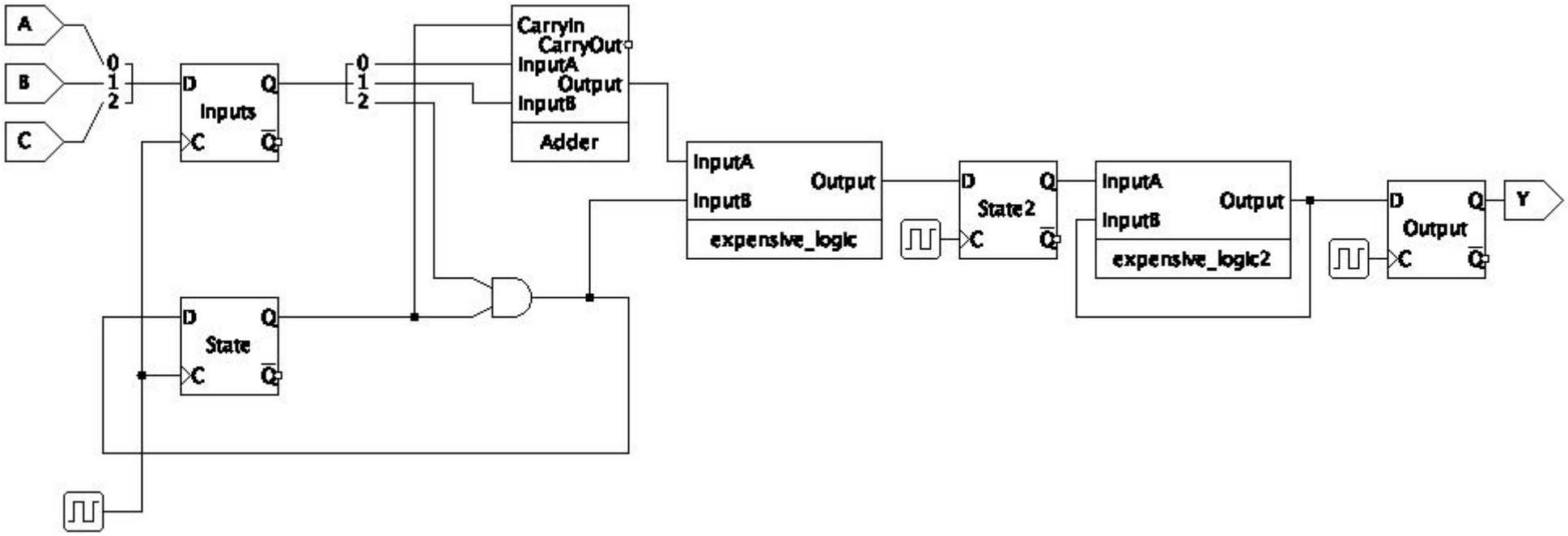
Timing Sequential Logic

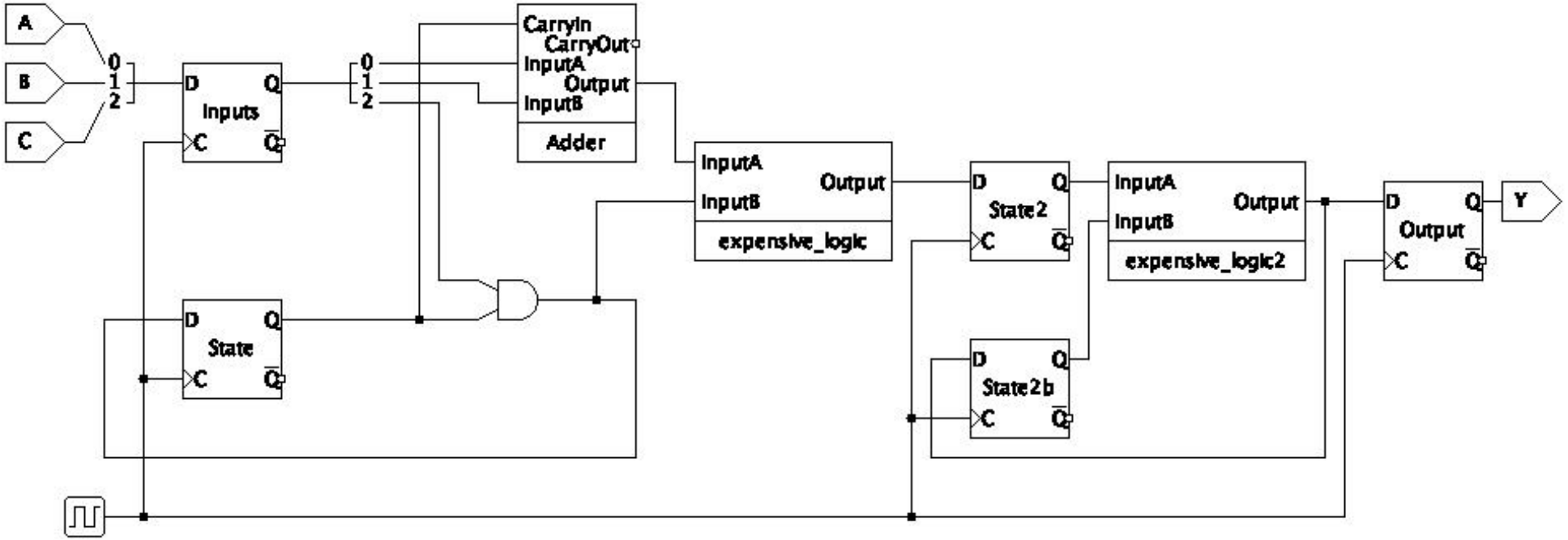
Maximum delay of combinational logic dictates how many gates can be on a critical path of a high-speed circuit.

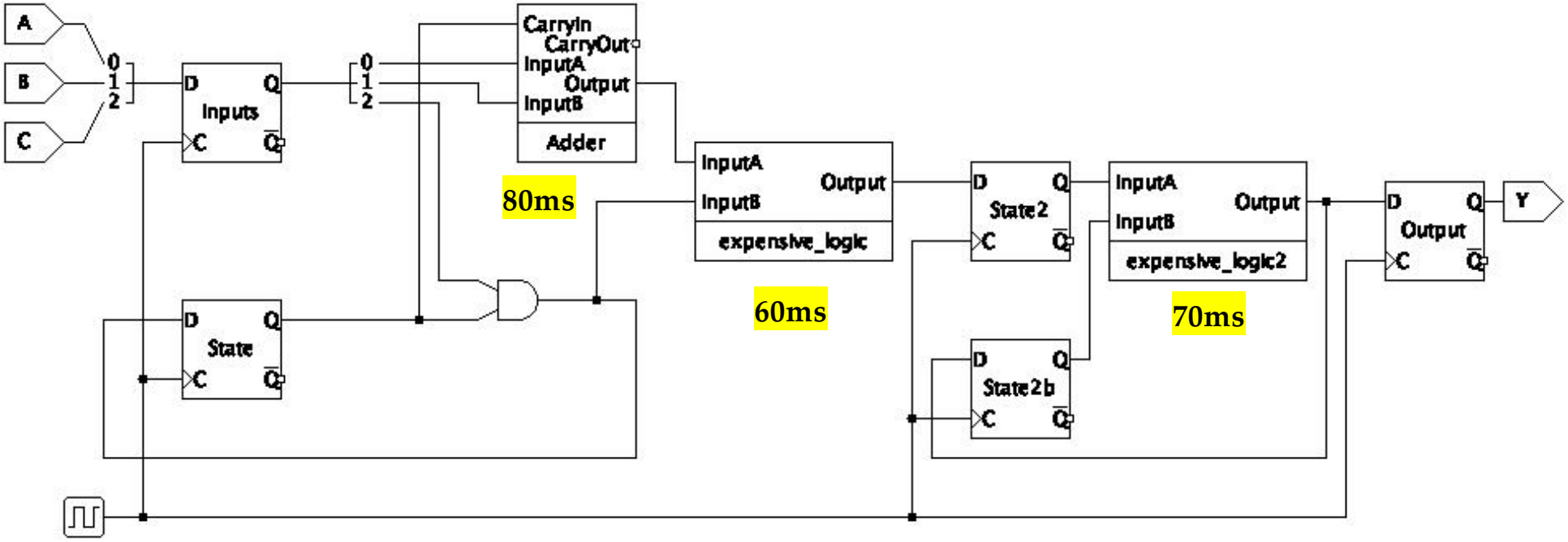
- Must finish before a clock cycle!

Mitigate by placing registers in the logic, breaking it up into more manageable chunks.

Two mistakes




A
B
C
0
1
2
D
Q
Inputs
C
Q
0
1
2
CarryIn
CarryOut
InputA
Output
InputB
Adder
InputA
Output
InputB
expensive_logic
D
Q
State2
C
Q
InputA
Output
InputB
expensive_logic2
D
Q
Output
C
Q
Y
D
Q
State
C
Q
D
Q
State2b
C
Q

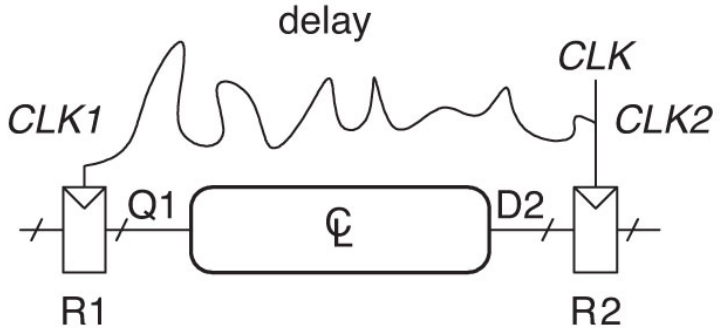

A
B
C
0
1
2
D
Q
Inputs
C
Q
0
1
2
CarryIn
CarryOut
InputA
Output
InputB
Adder
80ms
InputA
Output
InputB
expensive_logic
60ms
D
Q
State2
C
Q
InputA
Output
InputB
expensive_logic2
70ms
D
Q
Output
C
Q
Y
D
Q
State
C
Q
D
Q
State2b
C
Q

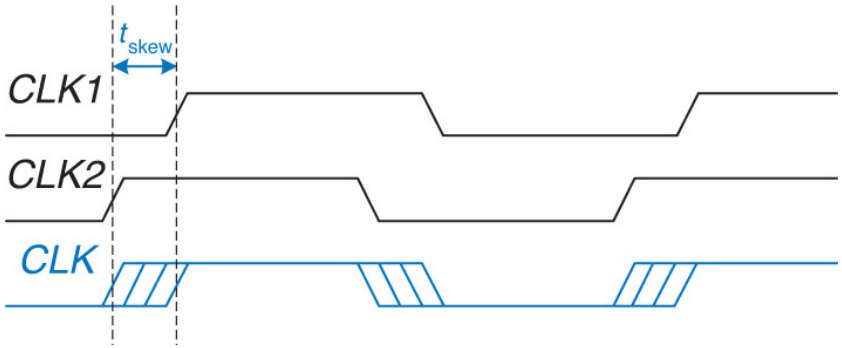
Clock Skew

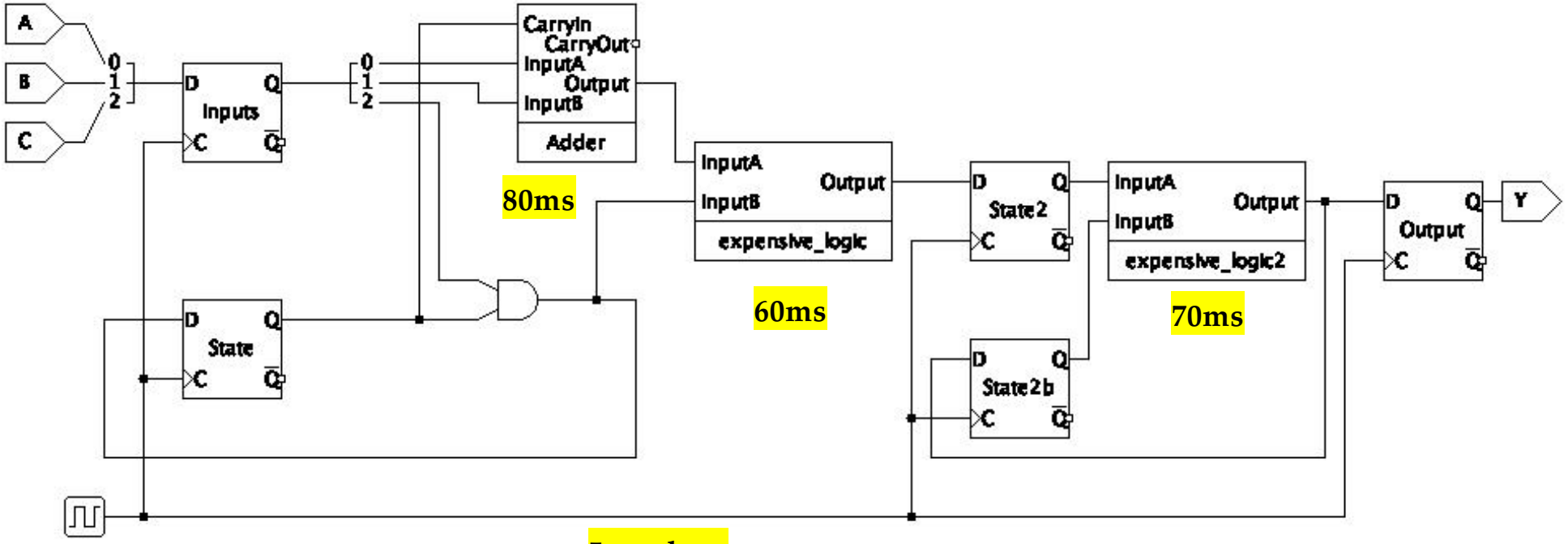
CLK doesn't reach all registers at the same time.

Adds to the overhead of timing.

Must increase minimum delay through combinatorial logic to compensate.



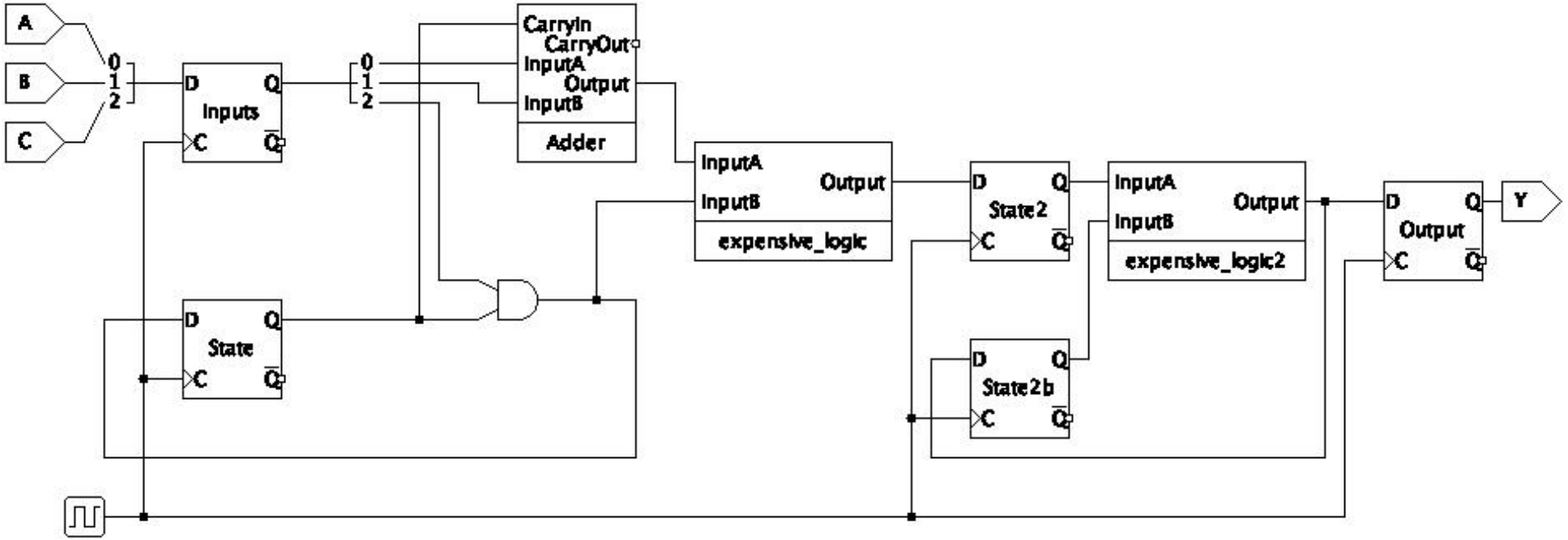



A
B
C
0
1
2
D
Q
Inputs
C
Q
0
1
2
CarryIn
CarryOut
InputA
Output
InputB
Adder
80ms
InputA
Output
InputB
expensive_logic
60ms
D
Q
State2
C
Q
InputA
Output
InputB
expensive_logic2
70ms
D
Q
Output
C
Q
Y
D
Q
State
C
Q
D
Q
State2b
C
Q
5ms skew

Pipelining

Claimed that we put a register in the middle to decrease clock period

Have we saved time by doing this?


A
B
C
0
1
2
D
Q
Inputs
C
Q
0
1
2
CarryIn
CarryOut
InputA
Output
InputB
Adder
InputA
Output
InputB
expensive_logic
D
Q
State2
C
Q
InputA
Output
InputB
expensive_logic2
D
Q
Output
C
Q
Y
D
Q
State
C
Q
D
Q
State2b
C
Q

Pipelining

Claimed that we put a register in the middle to decrease clock period

Have we saved time by doing this?

Not exactly

We will see later that even though one instruction goes more slowly now, we have made circuit more efficient