



CSCI 150

Introduction to Digital and Computer System Design

Lecture 5: Registers IV

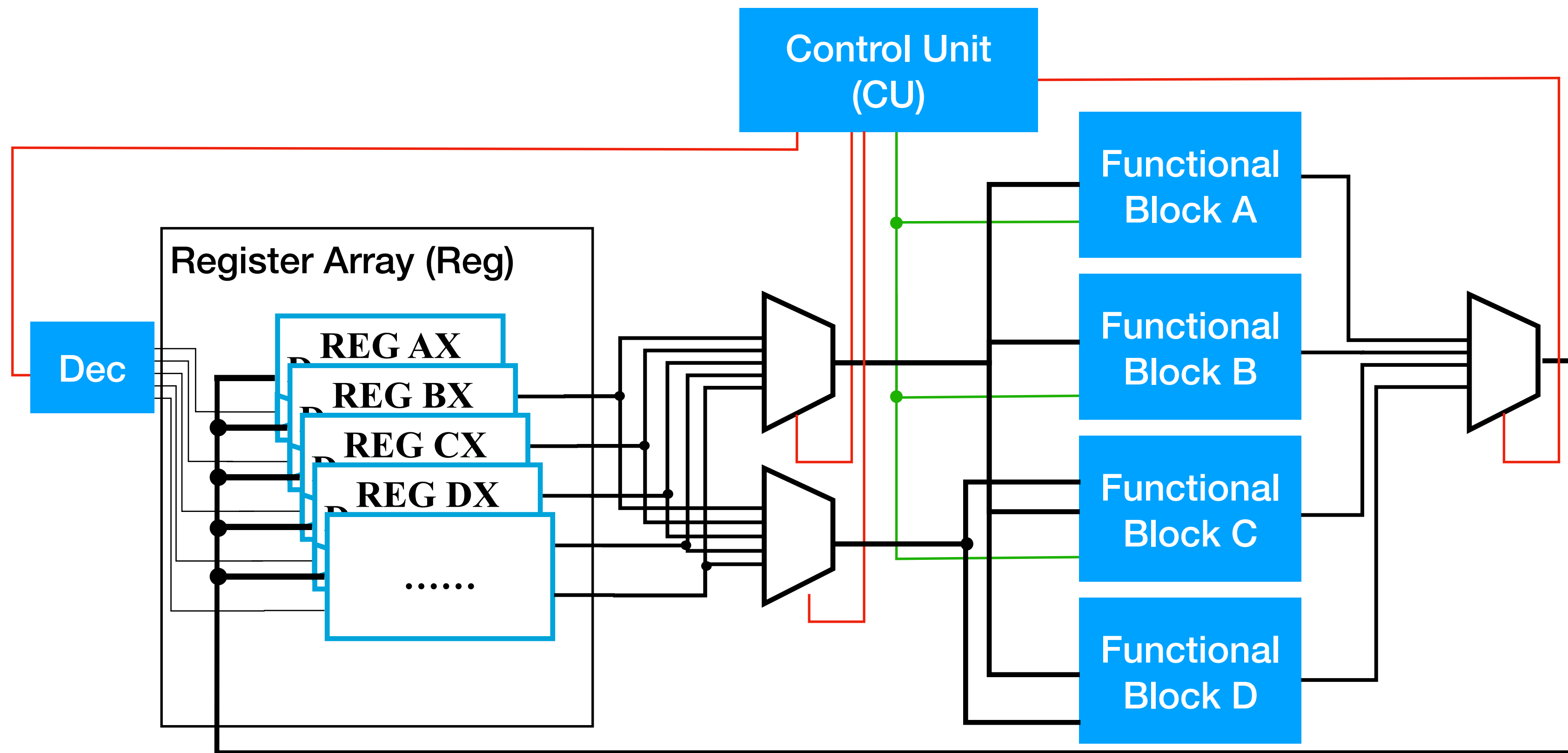


Jetic Gū
2020 Summer Semester (S2)

Overview

- Focus: Fundamentals of Complex Digital Circuit Design
- Architecture: von Neumann
- Textbook v4: Ch7 7.6, 7.7; v5: Ch6 6.6, 6.7
- Core Ideas:
 1. Register-Cell Design
 2. Counter

Example Datapath Architecture



Register Cell Design

One register for AND, OR, XOR

1. Specification

- Input: n -bit input A , n -bit output B
- Mode: M_{AND} , M_{OR} , M_{XOR} , only one of these can be 1. If all 0s preserve current value
- Output: n -bit output back to input register A (D_A)
 - If $M_{\text{AND}} = 1$, $D_A = A \cdot B$
 - If $M_{\text{OR}} = 1$, $D_A = A + B$
 - If $M_{\text{XOR}} = 1$, $D_A = A \oplus B$
 - If $M_{\text{AND}} + M_{\text{OR}} + M_{\text{XOR}} = 0$, $D_A = A$

2. Formulation

3. State Assignment

- For every bit $[0, n - 1]$

Present A State	Next A State (D_A)							
	AND=0 XOR=0 OR=0	OR=1 B=0	OR=1 B=1	AND=1 B=0	AND=1 B=1	XOR=1 B=0	XOR=1 B=1	
0	0	0	1	0	0	0	1	
1	1	1	1	0	1	1	0	

Example

4. Flip-Flop Input Equation

5. Output Equation

- For every bit $[0, n - 1]$

$$\begin{aligned} D_A = & M_{\text{AND}} \cdot (AB) \\ & + M_{\text{OR}} \cdot (A + B) \\ & + M_{\text{XOR}} \cdot (A\bar{B} + \bar{A}B) \\ & + \overline{M_{\text{OR}}} \cdot \overline{M_{\text{AND}}} \cdot \overline{M_{\text{XOR}}} \cdot A \end{aligned}$$

6. Optimisation

- For every bit $[0, n - 1]$

$$\begin{aligned}
 D_A &= M_{\text{AND}} \cdot (AB) && = M_{\text{AND}} \cdot \Sigma_m(3) \\
 &+ M_{\text{OR}} \cdot (A + B) && + M_{\text{OR}} \cdot \Sigma_m(1,2,3) \\
 &+ M_{\text{XOR}} \cdot (A\bar{B} + \bar{A}B) && + M_{\text{XOR}} \cdot \Sigma_m(1,2) \\
 &+ \overline{M_{\text{OR}}} \cdot \overline{M_{\text{AND}}} \cdot \overline{M_{\text{XOR}}} \cdot A && + \overline{M_{\text{OR}}} \cdot \overline{M_{\text{AND}}} \cdot \overline{M_{\text{XOR}}} \cdot \Sigma_m(2,3)
 \end{aligned}$$

6. Optimisation

- For every bit $[0, n - 1]$

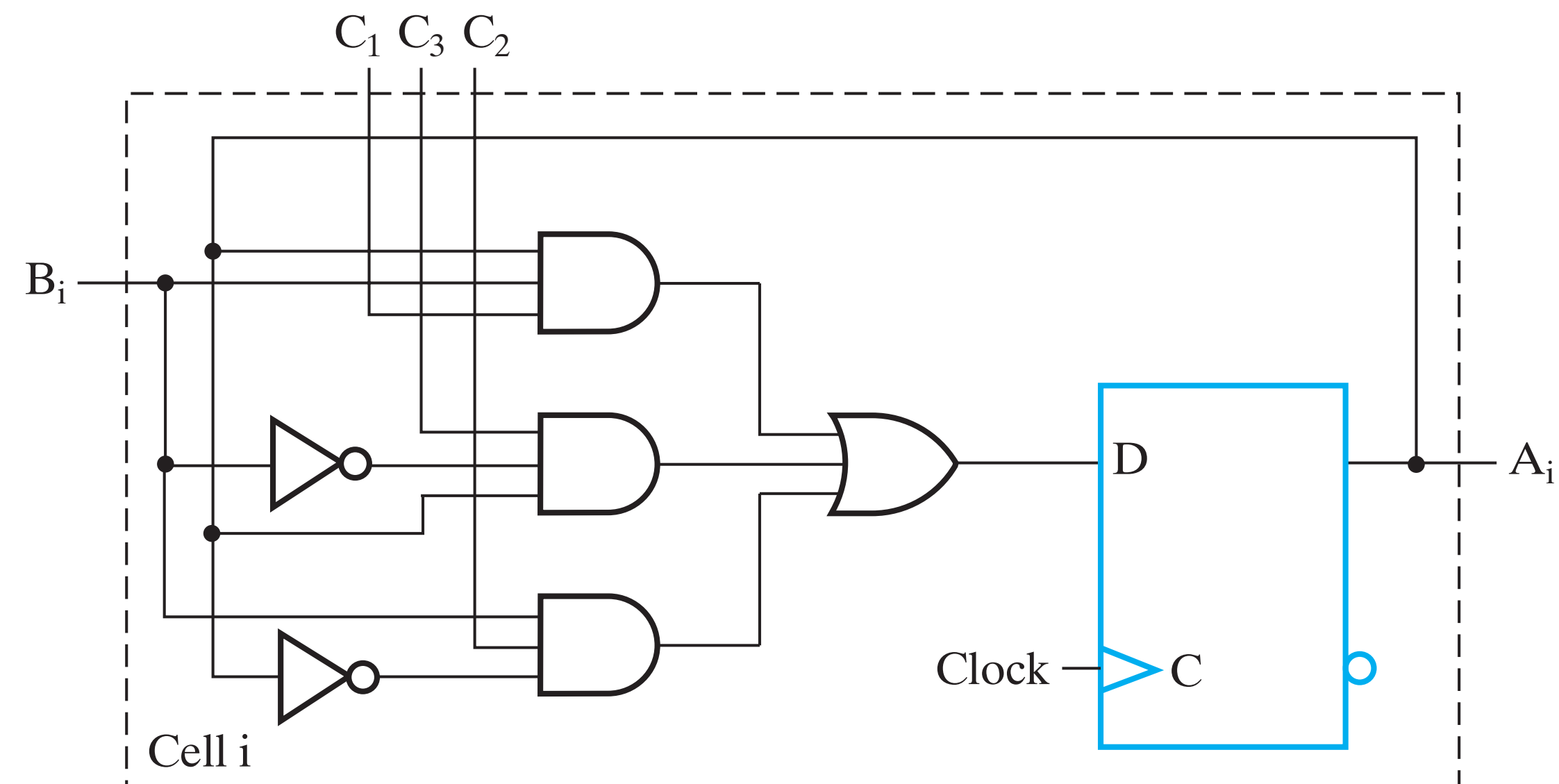
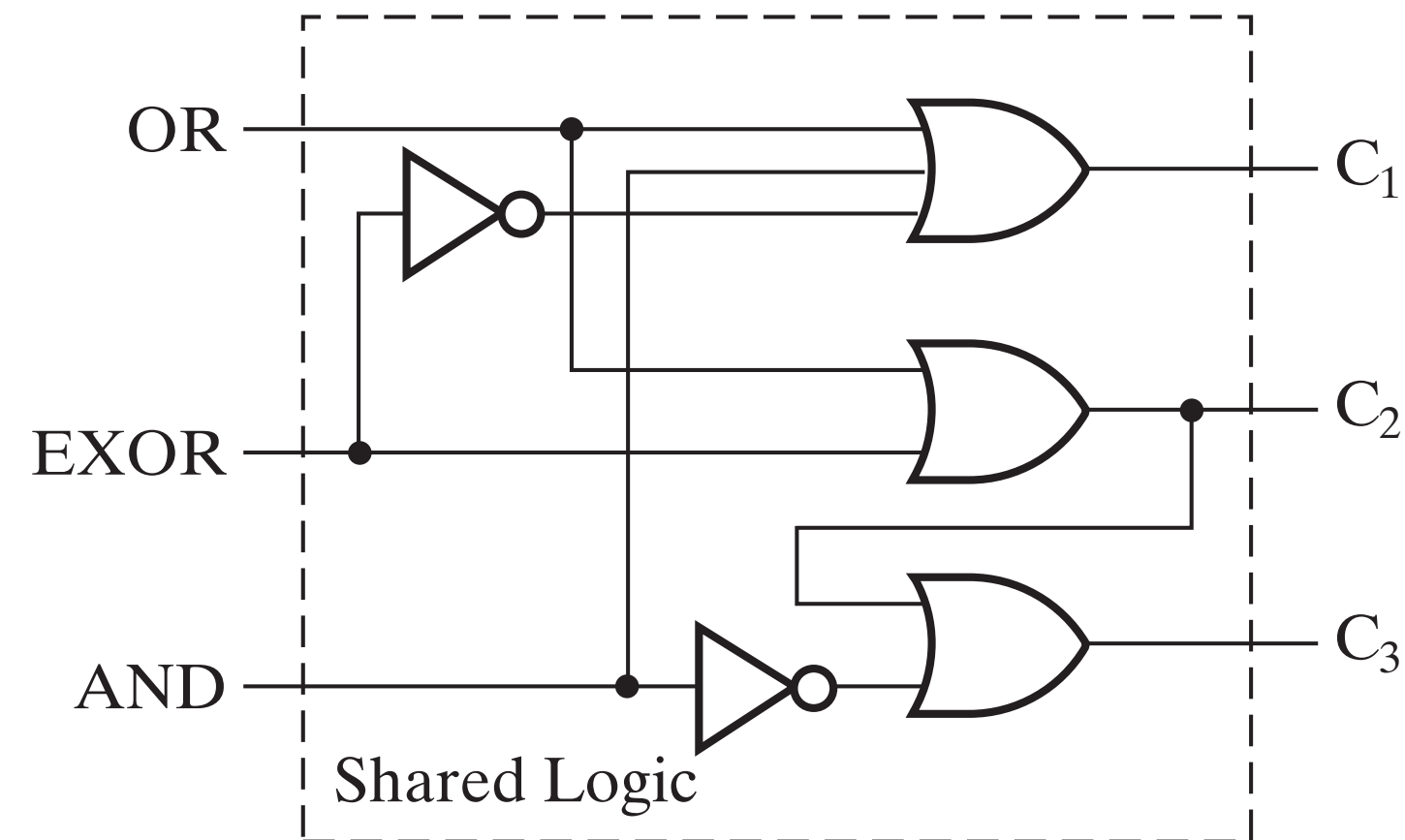
$$\begin{aligned}
 D_A &= M_{\text{AND}} \cdot (AB) && = M_{\text{AND}} \cdot \Sigma_m(3) \\
 &+ M_{\text{OR}} \cdot (A + B) && + M_{\text{OR}} \cdot \Sigma_m(1,2,3) \\
 &+ M_{\text{XOR}} \cdot (A\bar{B} + \bar{A}B) && + M_{\text{XOR}} \cdot \Sigma_m(1,2) \\
 &+ \overline{M_{\text{OR}}} \cdot \overline{M_{\text{AND}}} \cdot \overline{M_{\text{XOR}}} \cdot A && + \overline{M_{\text{OR}}} \cdot \overline{M_{\text{AND}}} \cdot \overline{M_{\text{XOR}}} \cdot \Sigma_m(2,3) \\
 \\
 &= m_3(M_{\text{AND}} + M_{\text{OR}} + \overline{M_{\text{OR}}} \cdot \overline{M_{\text{AND}}} \cdot \overline{M_{\text{XOR}}}) \\
 &\quad + m_2(M_{\text{OR}} + M_{\text{XOR}} + \overline{M_{\text{OR}}} \cdot \overline{M_{\text{AND}}} \cdot \overline{M_{\text{XOR}}}) \\
 &\quad + m_1(M_{\text{OR}} + M_{\text{XOR}})
 \end{aligned}$$

6. Optimisation

- For every bit $[0, n - 1]$

$$\begin{aligned}
 D_A &= M_{\text{AND}} \cdot (AB) && = M_{\text{AND}} \cdot \Sigma_m(3) \\
 &+ M_{\text{OR}} \cdot (A + B) && + M_{\text{OR}} \cdot \Sigma_m(1,2,3) \\
 &+ M_{\text{XOR}} \cdot (A\bar{B} + \bar{A}B) && + M_{\text{XOR}} \cdot \Sigma_m(1,2) \\
 &+ \overline{M_{\text{OR}}} \cdot \overline{M_{\text{AND}}} \cdot \overline{M_{\text{XOR}}} \cdot A && + \overline{M_{\text{OR}}} \cdot \overline{M_{\text{AND}}} \cdot \overline{M_{\text{XOR}}} \cdot \Sigma_m(2,3) \\
 &&& = m_3(M_{\text{AND}} + M_{\text{OR}} + \overline{M_{\text{XOR}}}) \\
 &&& + m_2(M_{\text{OR}} + M_{\text{XOR}} + \overline{M_{\text{AND}}}) \\
 &&& + m_1(M_{\text{OR}} + M_{\text{XOR}})
 \end{aligned}$$

7. Technology Mapping



Register Cells

- Register Cells are specific register designed to perform certain computation
 - What we just did was for AND, OR, XOR
- Register Cell for AND, OR, XOR, and NOT
- Register Cell for AND, OR, XOR, NOT, and Shifts
- etc.

Counter Functional Blocks

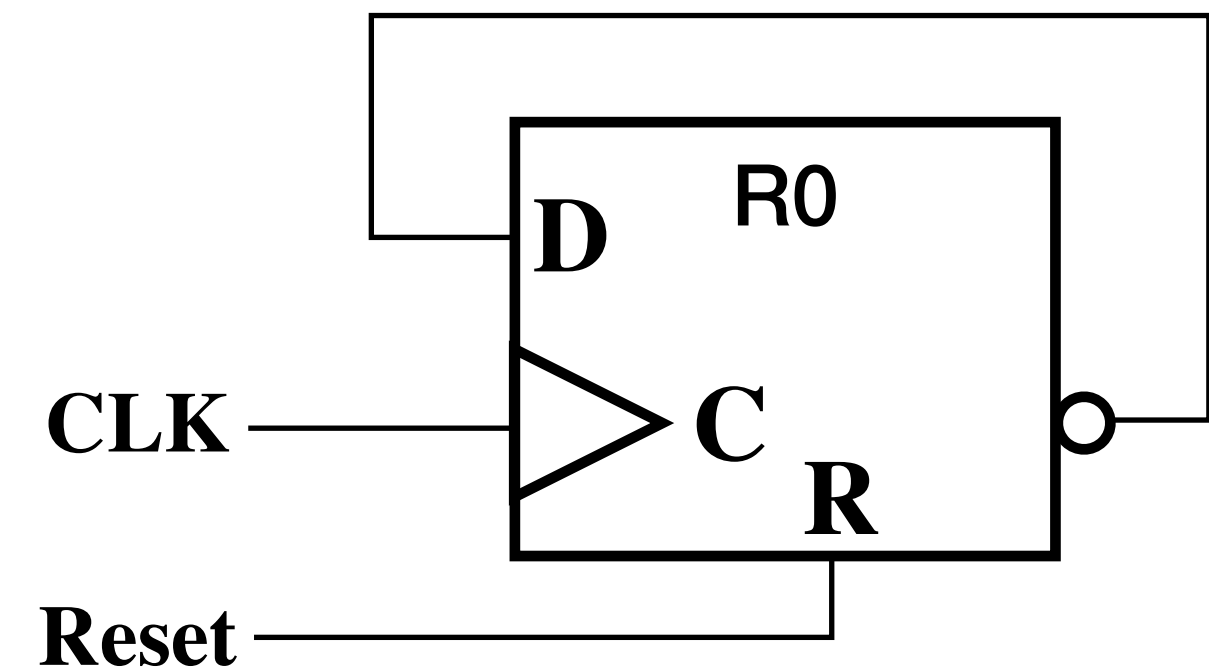
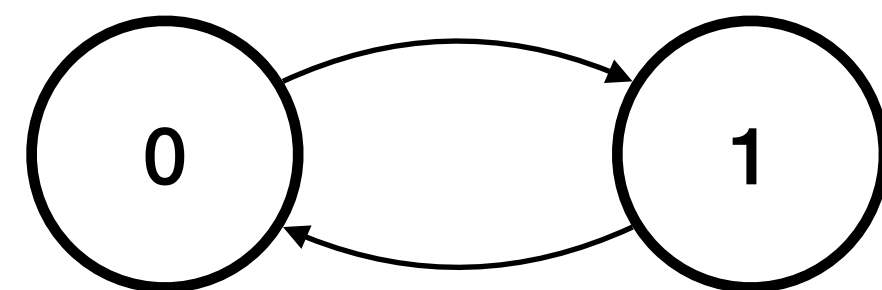
Ripple Counter; Synchronous Binary Counter;
BCD Counter

Counter

- Register Cells for counting
 - Reset: set counter to 0
 - Every CLK tick: add 1 to the register
1. Ripple Counter
 2. Synchronous Counters
 3. BCD Counter

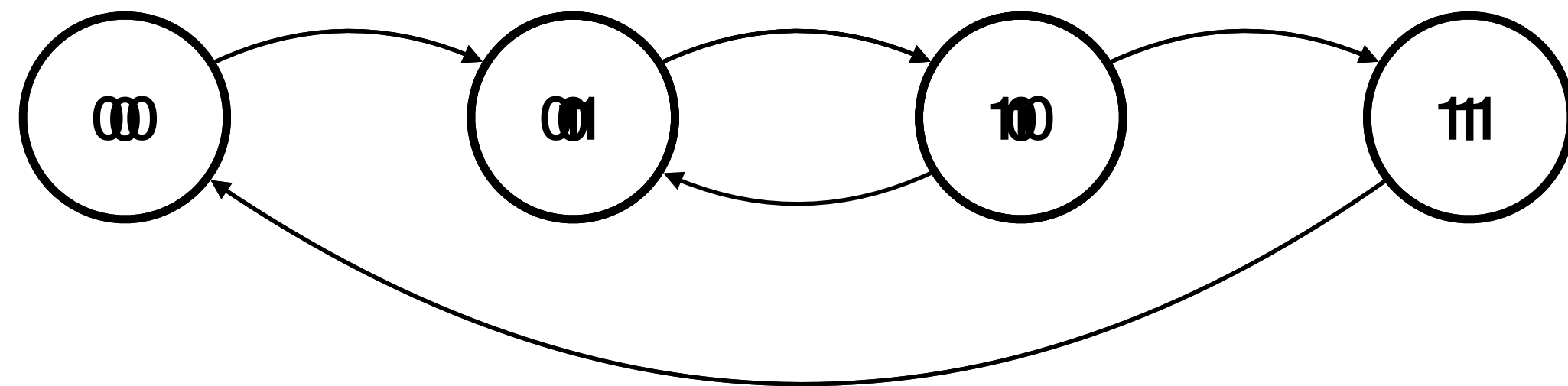
Ripple Counter

- 1-bit Counter
- What is the behaviour of the circuit on the right?

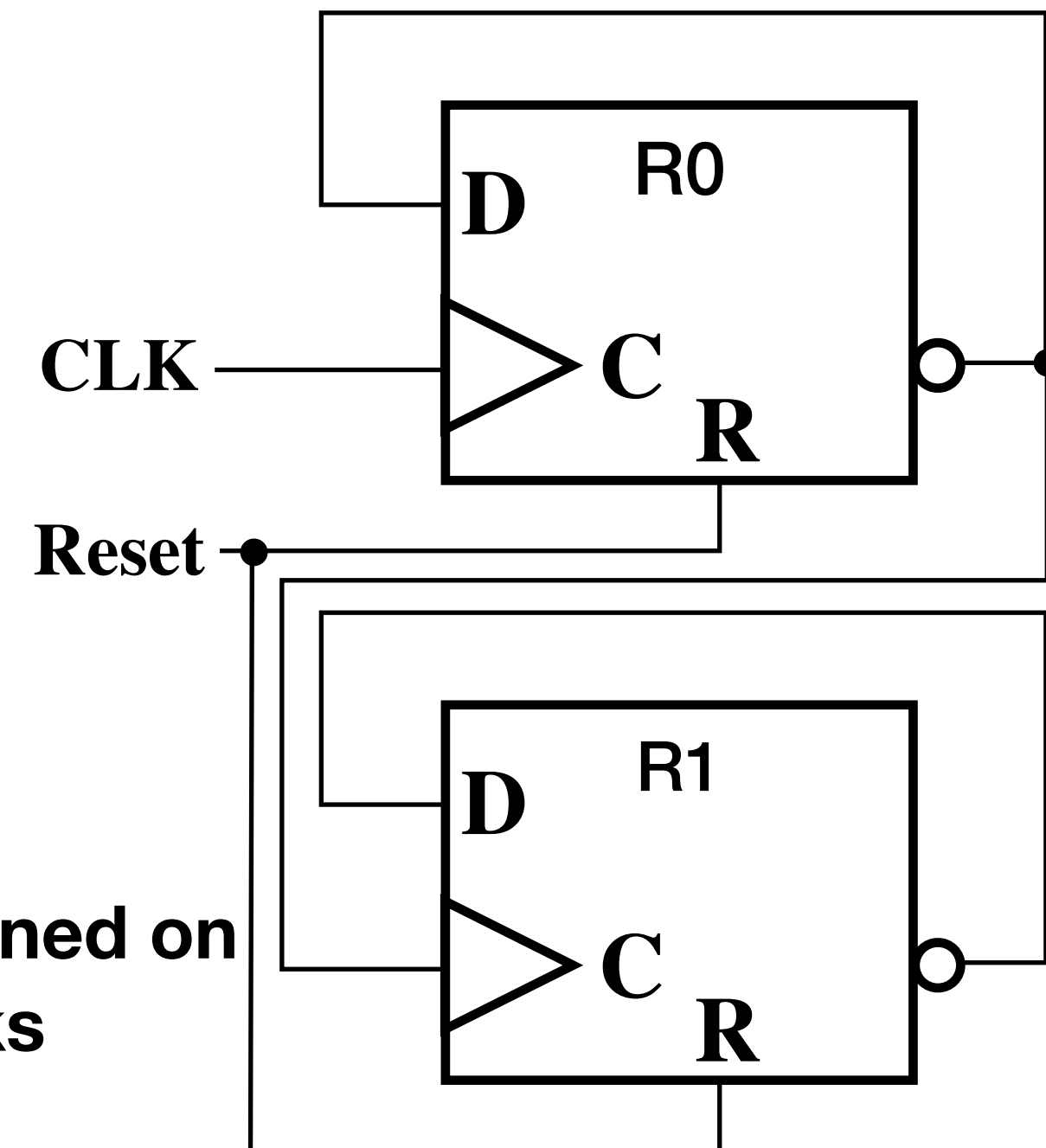


Ripple Counter

- 2-bit Counter
 - What is the behaviour of the circuit on the right?

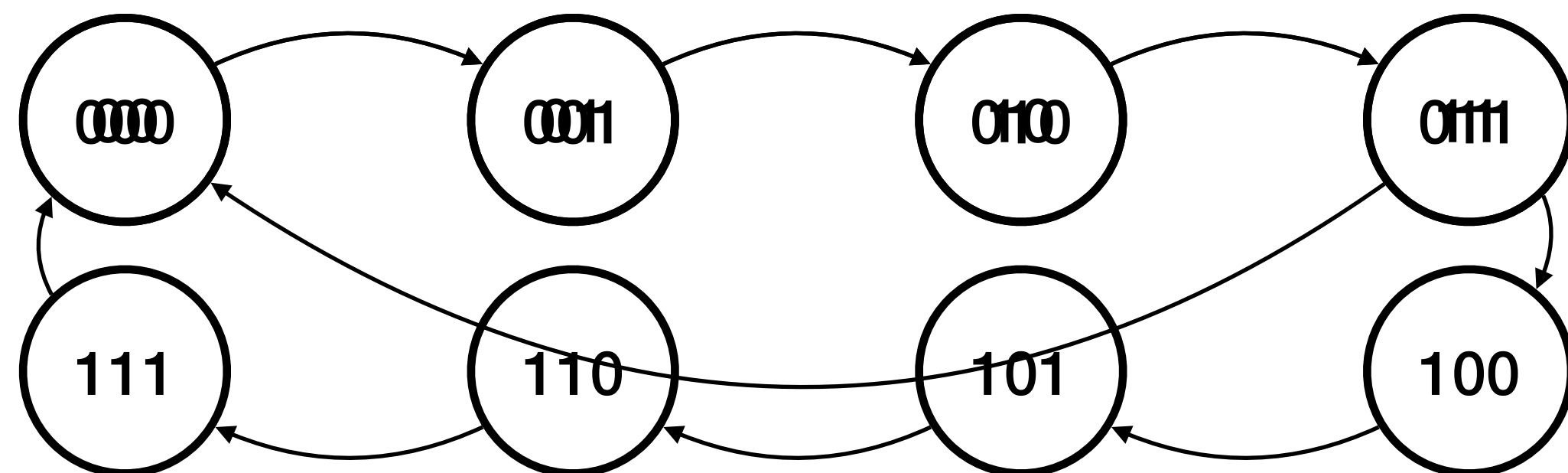


C pin of R1 is turned on every 2 ticks



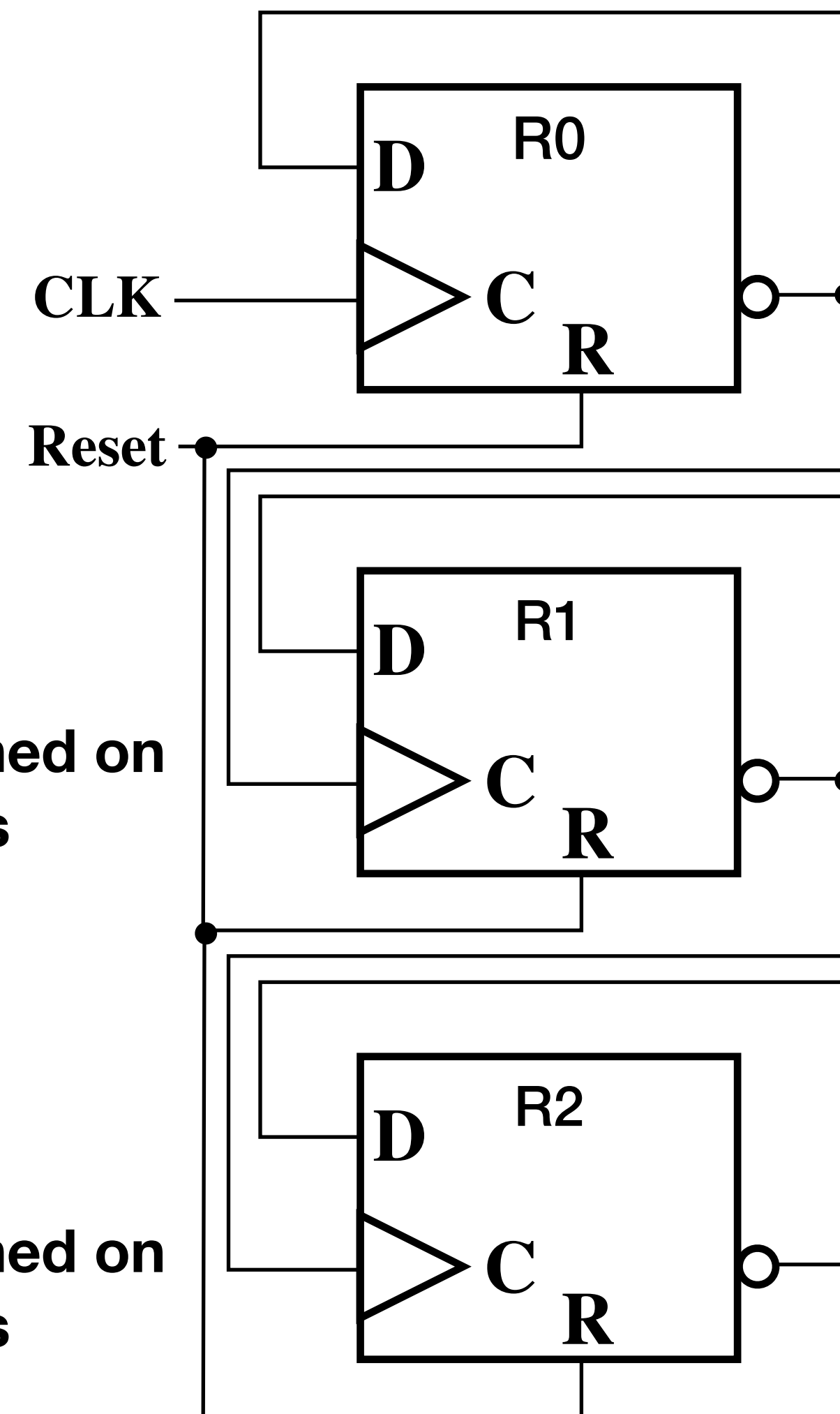
Ripple Counter

- 3-bit Counter
- What is the behaviour of the circuit on the right?



C pin of R1 is turned on every 2 ticks

C pin of R2 is turned on every 4 ticks



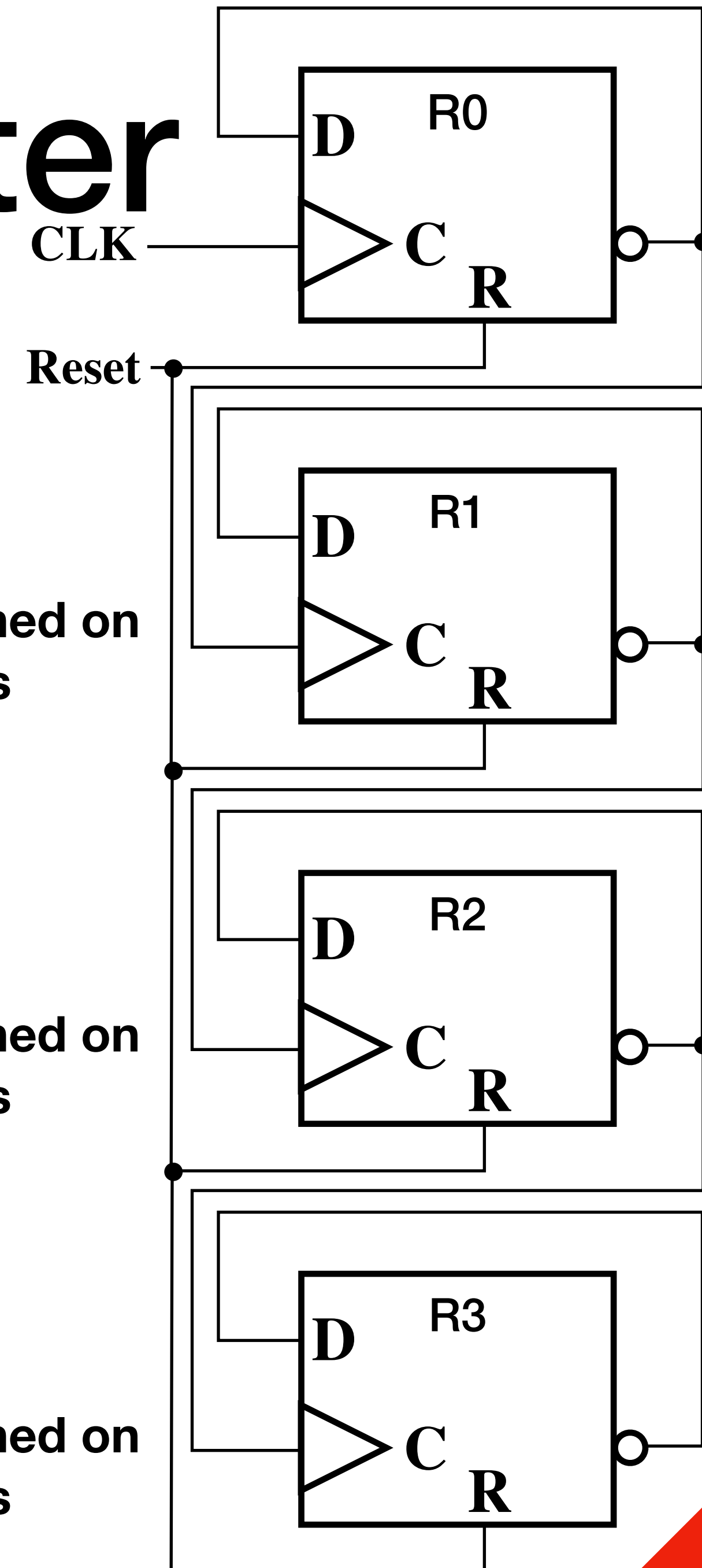
Ripple Counter

- 4-bit Counter
- What is the behaviour of the circuit on the right?

C pin of R1 is turned on every 2 ticks

C pin of R2 is turned on every 4 ticks

C pin of R3 is turned on every 8 ticks



Synchronous Binary Counter

Upward Counting Sequence

Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Downward Counting Sequence

Q_3	Q_2	Q_1	Q_0
1	1	1	1
1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	1
0	0	1	0
0	0	0	1
0	0	0	0

Synchronous Binary Counter

Upward Counting Sequence				Downward Counting Sequence			
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	0
0	0	1	0	1	1	0	1
0	0	1	1	1	1	0	0
0	1	0	0	1	0	1	1
0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	0	0
1	0	0	0	0	1	1	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	1	1
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	0

- Upward Q_i flips when ?; Downward equals ?

Synchronous Binary Counter

- Upward
 - $D_{A0} = \overline{Q_0}$
 - $D_{Ai} = Q_i \oplus (\prod_{j < i} Q_j)$, for all $i \in [1, n - 1]$
- Downward
 - Take $\overline{Q_i}$ as output (using e.g. multiplexer), for all $i \in [0, n - 1]$

Exercise

- Implement 4-bit Ripple Counter in LogicWorks
- Implement 4-bit Synchronous Counter in LogicWorks
- Design 3 digit BCD counter using 4-bit binary counters

Tutorial

Bus, Register Cells, Datapath