University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Another Example of Serialization: Power-of-2 Checker

Review of General Bit-Slice Model

General model parameters

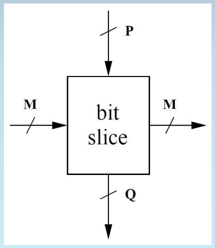
N-bit operands

P bits of input from operands

Q bits of output produced

M bits between bit slices

R bits of final output (not shown; produced by output logic operating on **M** bits from last bit slice).



Parameter Values for a Power-of-2 Checker

Power-of-2 checker parameters

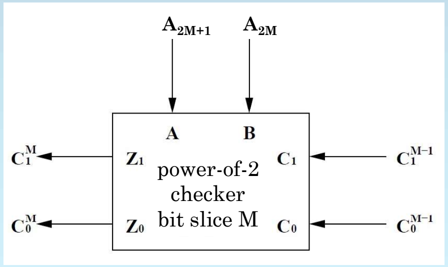
N-bit operands

P = 2

Q = 0

M = 2

R = 1



Initialization of a Power-of-2 Checker

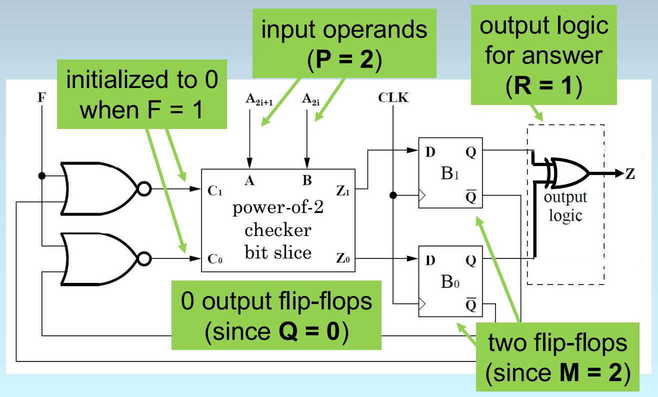
Our power-of-2 checker bit slice uses the representation shown here to pass information between slices.

What values should be passed to the first bit slice?

No 1 bits yet, so C_1C_0 = 00

C_1	C_0	meaning
0	0	no 1 bits
0	1	one 1 bit
1	0	not used
1	1	>one 1 bit

Example: A Power-of-2 Checker


initialized to 0 when F = 1
input operands (**P = 2**)
output logic for answer (**R = 1**)
F
A_{2i+1}
A_{2i}
CLK
C_1
A
B
Z_1
power-of-2 checker bit slice
C_0
Z_0
D
Q
B_1
$\bar{Q}$
D
Q
B_0
$\bar{Q}$
output logic
Z
0 output flip-flops (since **Q = 0**)
two flip-flops (since **M = 2**)

Discrete Time Implies Delayed Results: N = 8

yellow = inputs

(green = bit slice labels)

cycle #	F	A	B	B_1	B_0	C_1	C_0	Z_1	Z_0
0	1	0	0	bits	bits	0	0	0	0
1	0	0	1	0	0	0	0	0	1
2	0	0	0	0	1	0	1	0	1
3	0	1	0	0	1	0	1	1	1
4	x	x	x	1	1	?	?	?	?

blue = outputs **$Z = B_1 \oplus B_0 = 0$**

A Power-of-2 Checker Consists of Four Parts

Let's analyze the area of a power-of-2 checker.

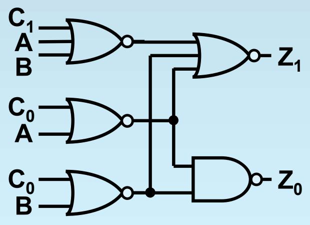
We have:
- one bit slice,
- two flip-flops,
- two 2-input NOR gates (selection logic),
- and a 2-input XOR.

16 2-input gates and four inverters

A Power-of-2 Checker Contains One Bit Slice

Here's our design for the power-of-2 checker bit slice.

So we need **three 2-input gates** and **two 3-input gates**.



A Power-of-2 Checker Consists of Four Parts

Let's analyze the area of a power-of-2 checker.

We have:
- one bit slice,
- two flip-flops,
- two 2-input NOR gates (selection logic),
- and a 2-input XOR.

three 2-input gates and two 3-input gates

16 2-input gates and four inverters

Total: 3+16+2 = **21 2-input gates**,
2 = **2 3-input gates**,
4 = **4 inverters**, and
1 2-input XOR.

Serial Design is Smaller for N ≥ 10

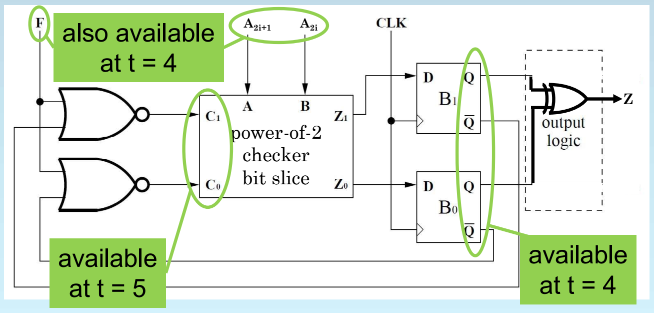
To handle **N-bit** operands (2 bits per bit slice), a bit-sliced design requires:
- **3N/2 2-input gates** (6N transistors),
- **N 3-input gates** (6N transistors), and
- **1 2-input XOR**.

A serial design (independent of **N**) requires
- **21 2-input gates** (84 transistors),
- **2 3-input gates** (12 transistors),
- **4 inverters** (8 transistors), and
- **1 2-input XOR**.

The serial design is smaller for N ≥ 10.

When Are Inputs Available?

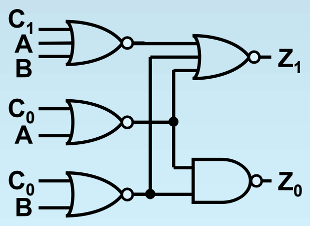
A rising edge arrives at t = 0 (gate delays).



Delay Analysis for the Bit Slice

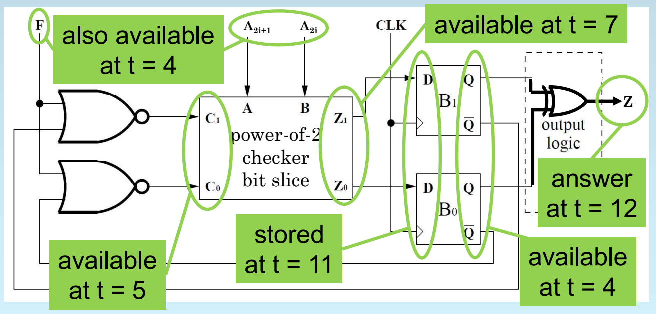
And delay?

2 on all paths.



When Are Results Stored?

A rising edge arrives at t = 0 (gate delays).



Serial Design is At Least 5.5x Slower

To handle **N-bit** operands, a bit-sliced design requires **N + 1 gate delays**.

For a serial design,

- the clock cycle must be **at least 11 gate delays**, and
- we must execute for **N/2** cycles, so
- **N-bit** operands require **at least 11N/2 gate delays**.

The serial design is at least ~5.5x slower.

(And may be even slower!)