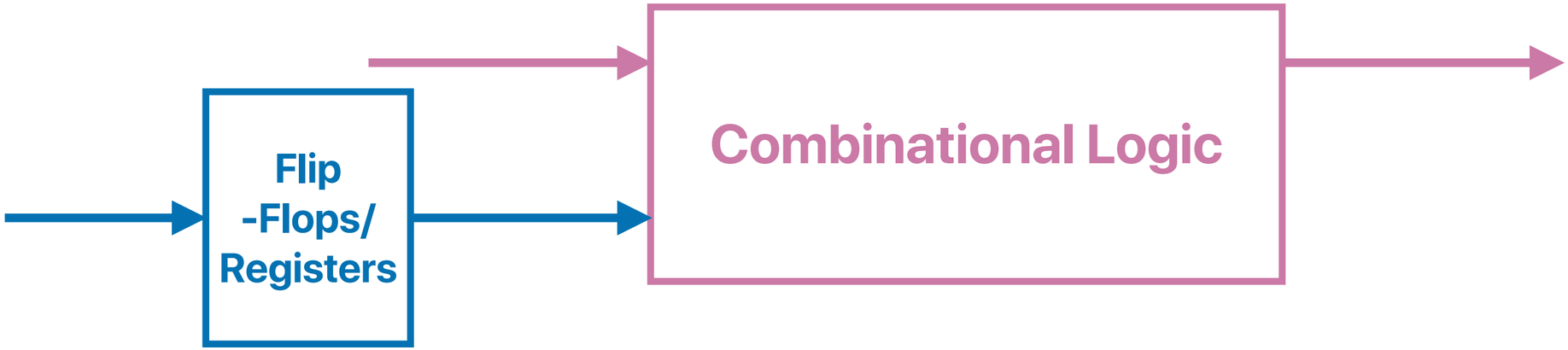
Final Review

Prof. Usagi

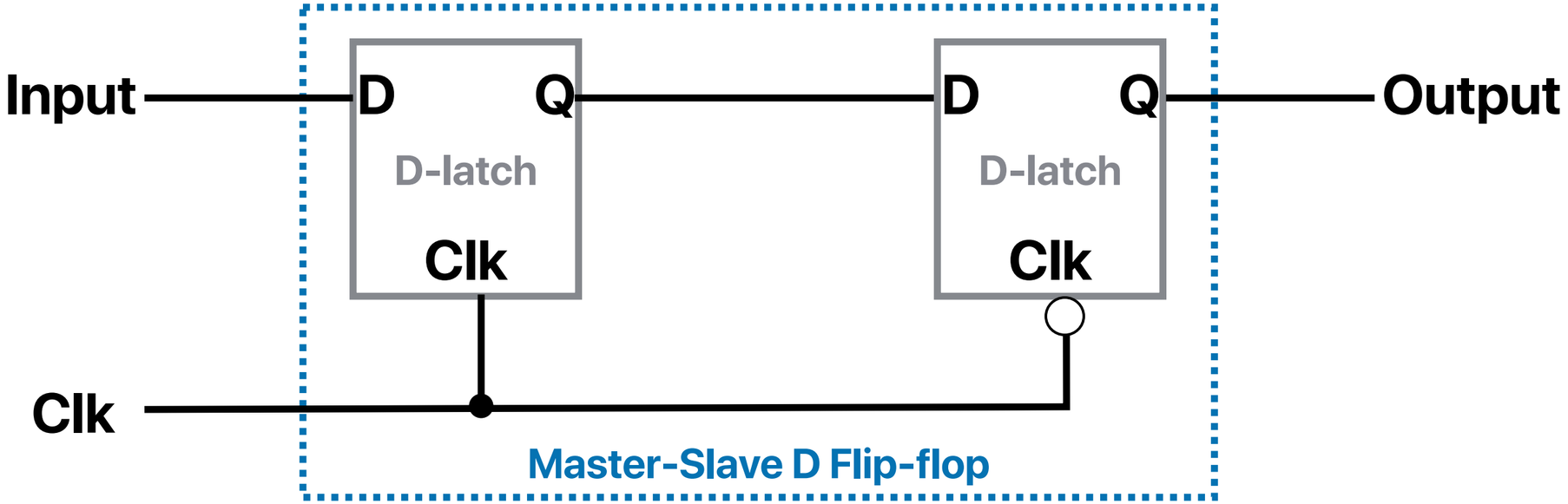
Big picture of a digital design

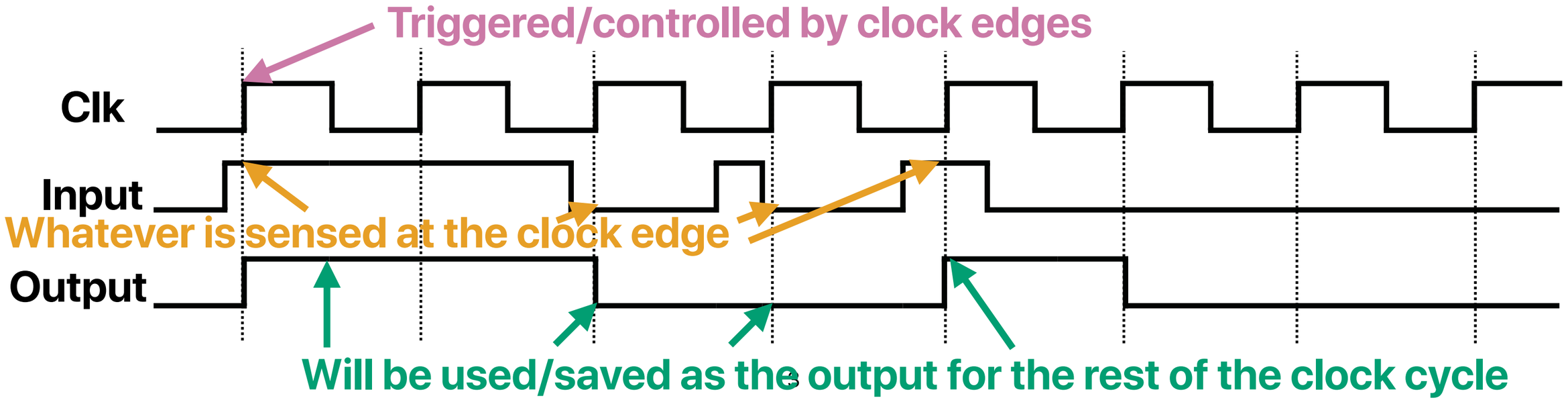


- Flip-flops provide "memory"
- Circuits can use the values in "memory" as another input
- With FFs, the design is now called a "sequential network" that implements a "finite state machine"

- An implementation of "boolean functions"
- Idempotent — the output only depends on the input
- Has no memory

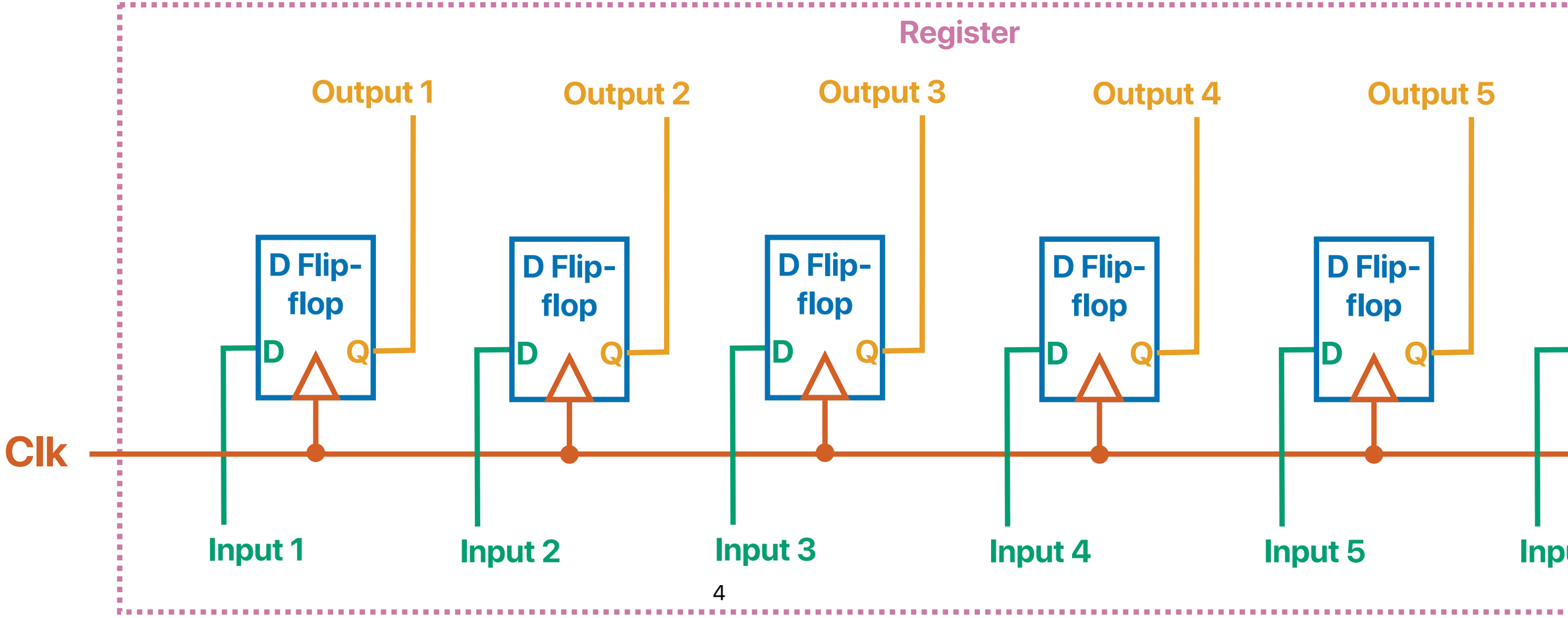
Recap: D flip-flop



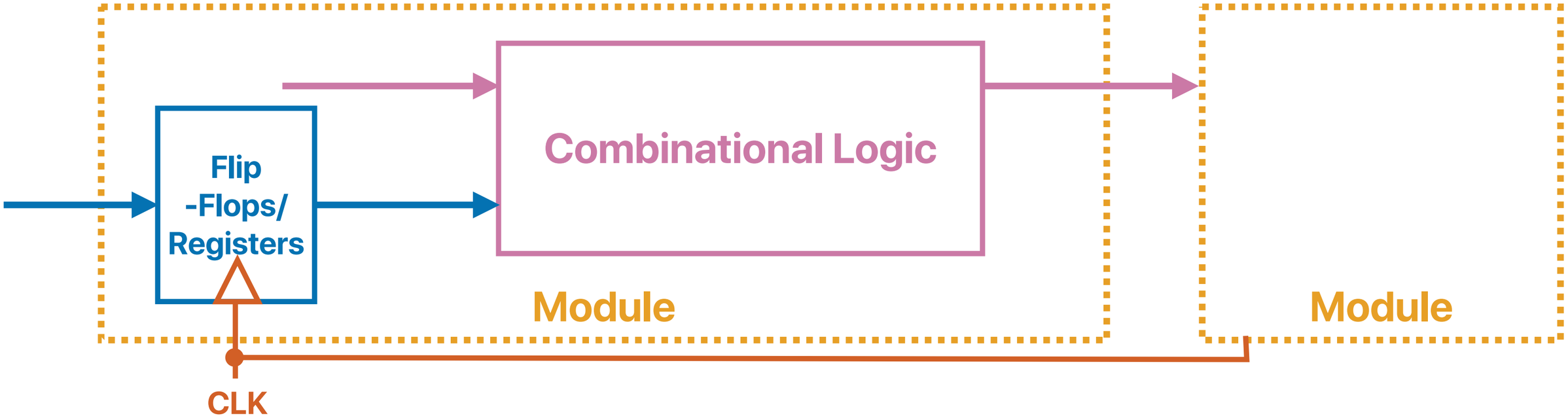


Registers

- Register: a sequential component that can store multiple bits
- A basic register can be built simply by using multiple D-FFs



Big picture of a digital design



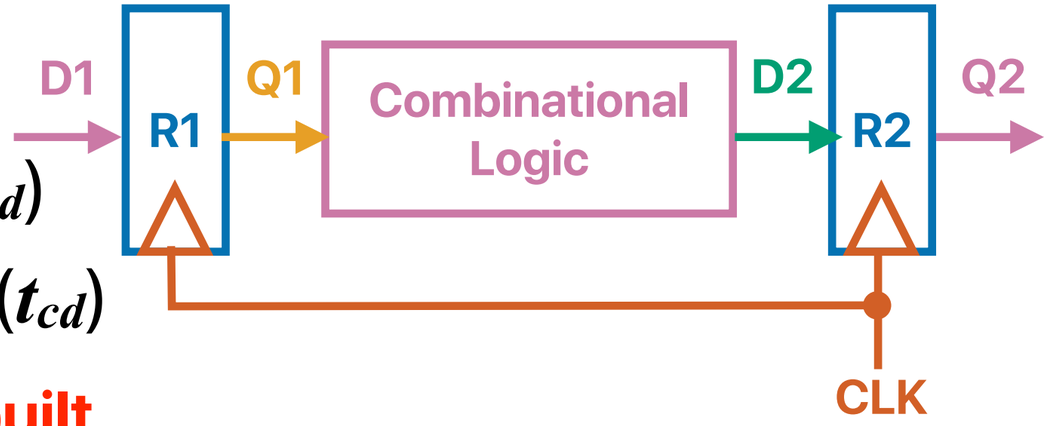
- Flip-flops provide "memory"
- Circuits can use the values in "memory" as another input
- With FFs, the design is now called a "sequential network" that implements a "finite state machine"

- An implementation of "boolean functions"
- Idempotent — the output only depends on the input
- Has no memory

Summary on timing constraints

- Combinational:
  - Maximum delay = **P**ropagation **d**elay (t_{pd})
  - Minimum delay = **C**ontamination **d**elay (t_{cd})
- Flip Flops:
  - Input
    - **Setup** time (t_{setup})
    - **Hold** time (t_{hold})
  - Output
    - **P**ropagation **c**lock-to-**Q** time (t_{pcq})
    - **C**ontamination **c**lock-to-**Q** time (t_{ccq})

Once the logic/FFs are built, these timing characteristics are fixed properties

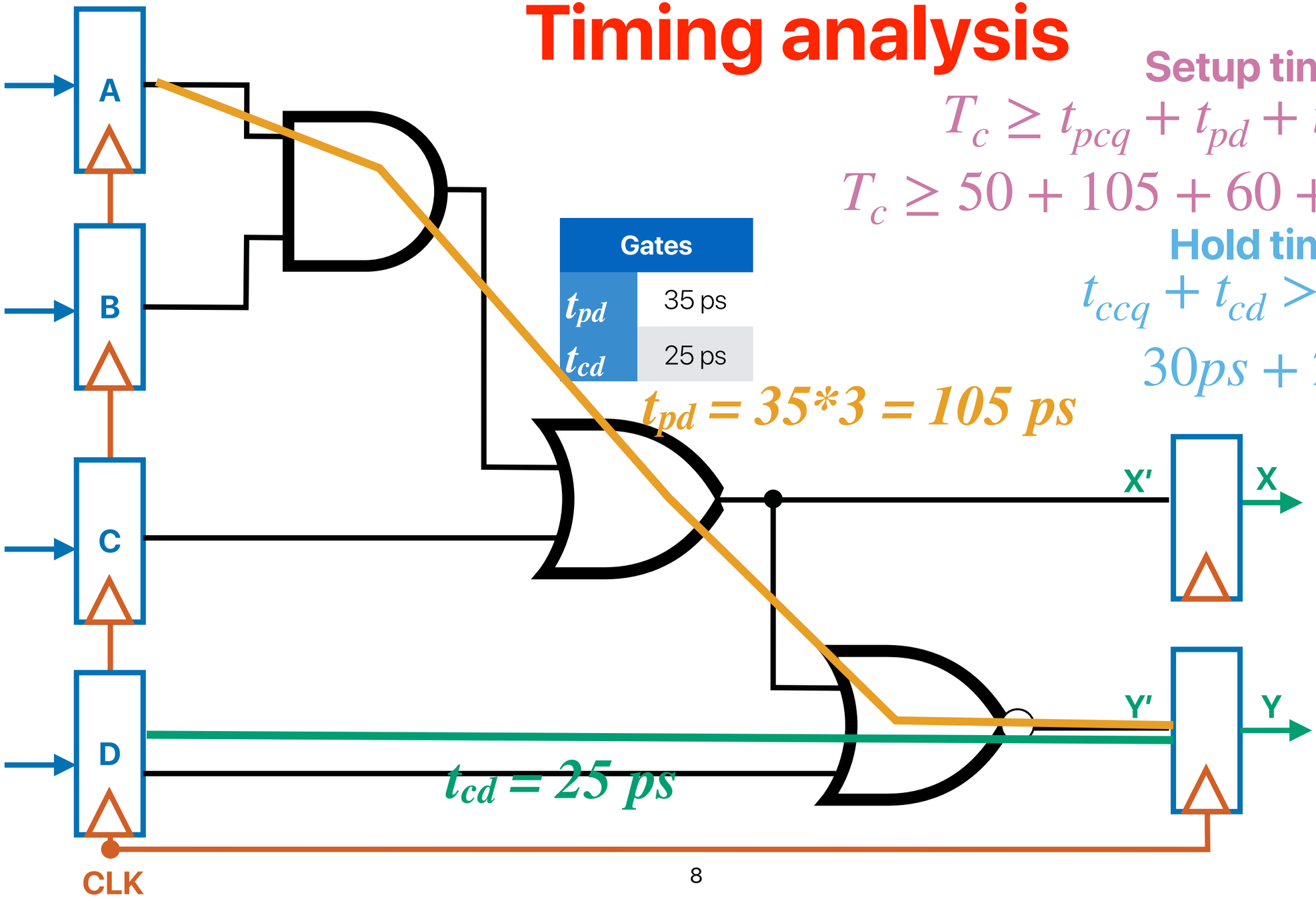


How long is the clock cycle time?

$$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

$$t_{ccq} + t_{cd} > t_{hold} + t_{skew}$$

Timing analysis



Gates	
t_{pd}	35 ps
t_{cd}	25 ps

$t_{pd} = 35*3 = 105\ ps$

$t_{cd} = 25\ ps$

Setup time constraints

$$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

$$T_c \geq 50 + 105 + 60 + 0 = 215ps$$

Hold time constraints

$$t_{ccq} + t_{cd} > t_{hold}$$

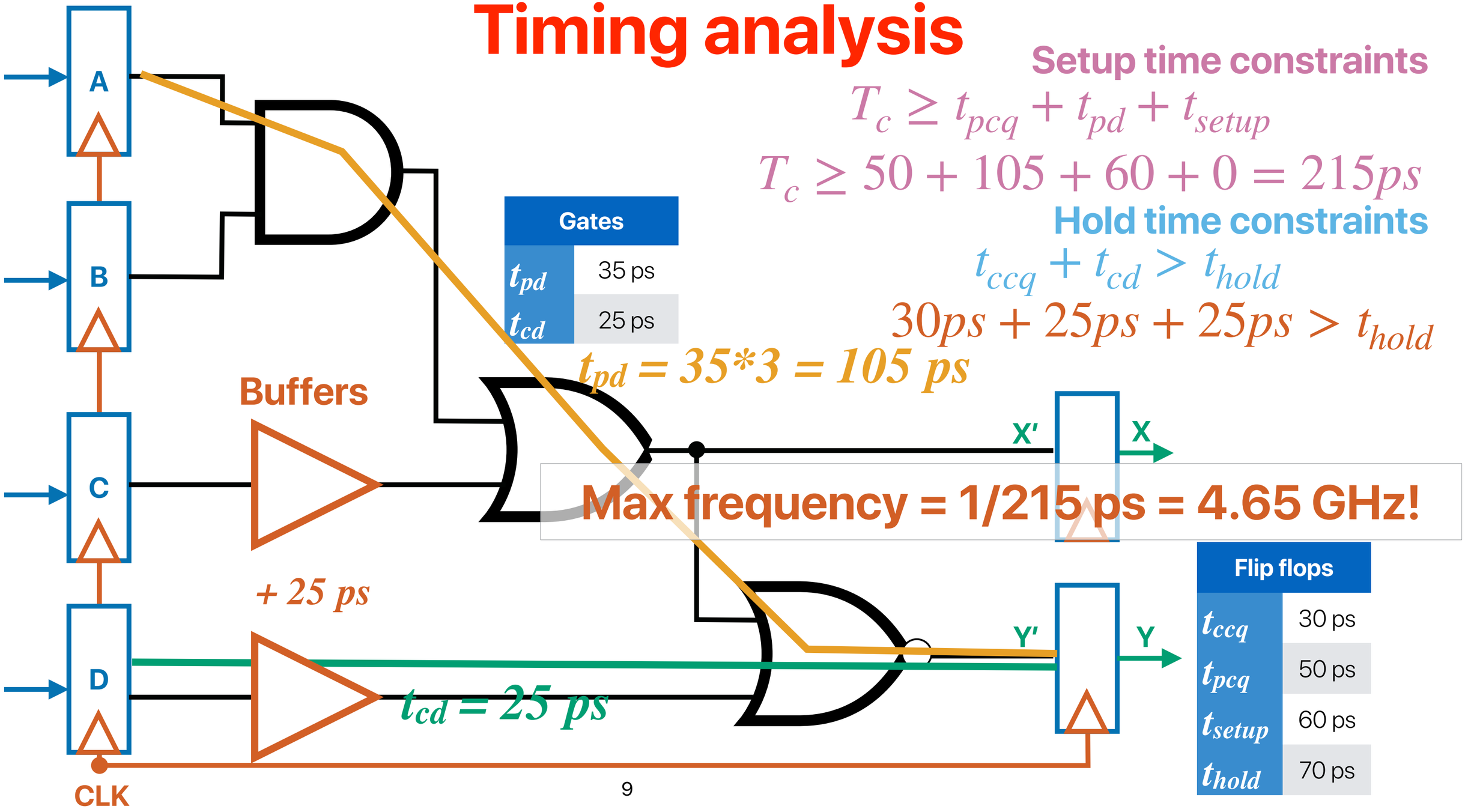
$$30ps + 25ps > t_{hold}$$

t_{hold} **= 70 ps!**

No!!!

Flip flops	
t_{ccq}	30 ps
t_{pcq}	50 ps
t_{setup}	60 ps
t_{hold}	70 ps

Timing analysis



Setup time constraints

$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$

$$T_c \geq 50 + 105 + 60 + 0 = 215ps$$

Hold time constraints

$$t_{ccq} + t_{cd} > t_{hold}$$

$$30ps + 25ps + 25ps > t_{hold}$$

Gates	
t_{pd}	35 ps
t_{cd}	25 ps

$t_{pd} = 35*3 = 105$ ps

$t_{cd} = 25$ ps

Max frequency = 1/215 ps = 4.65 GHz!

Flip flops	
t_{ccq}	30 ps
t_{pcq}	50 ps
t_{setup}	60 ps
t_{hold}	70 ps

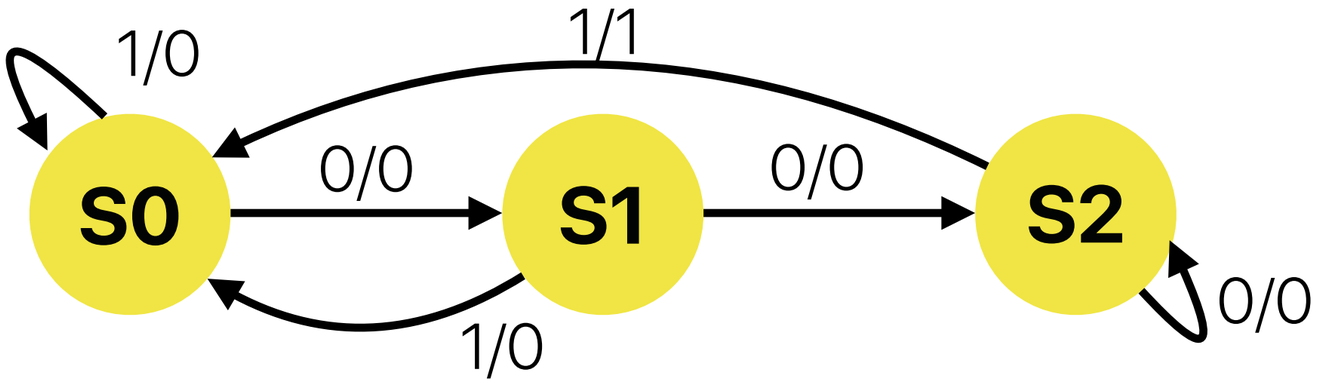
How to design a sequential network?

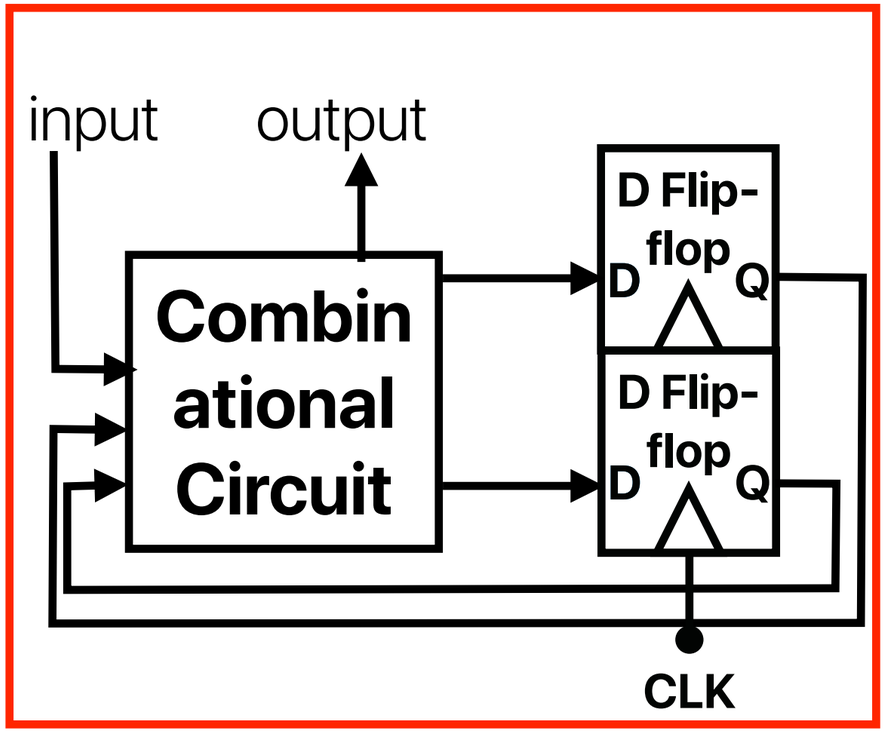
Sequential Circuit Design Flow

- Input Output Relation
- State Diagram (Transition of states)
  - State minimization (Reduction)
  - Finite state machine partitioning
- State Assignment (Map states into binary code)
  - Binary code, Gray encoding, One hot encoding, Coding optimization
- State Table (Truth table of states)
- Excitation Table (Truth table of the combinational circuits)
  - K Map, Minimal Expression
  - Logic Diagram

Life on Mars

State Diagram



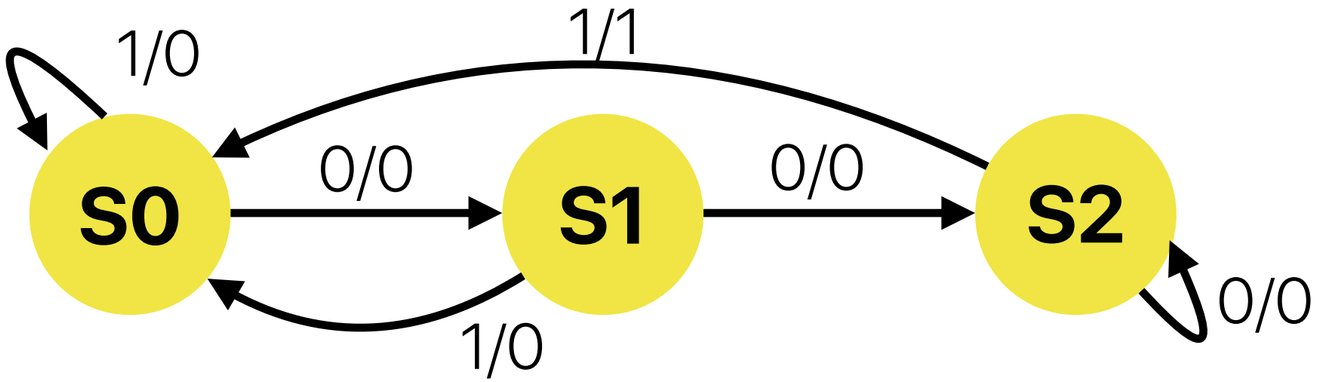


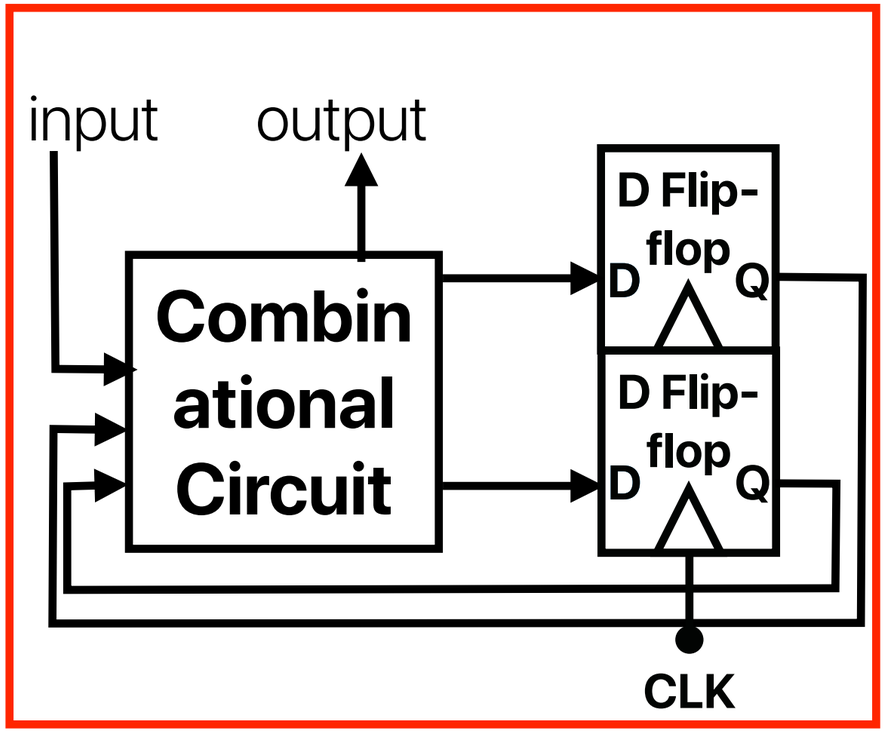
State Diagram

Current State	Next State, Output Input	
	0	1
S0	S1, 0	S0, 0
S1	S2, 0	S0, 0
S2	S2, 0	S0, 1

Life on Mars

State Diagram





State Diagram

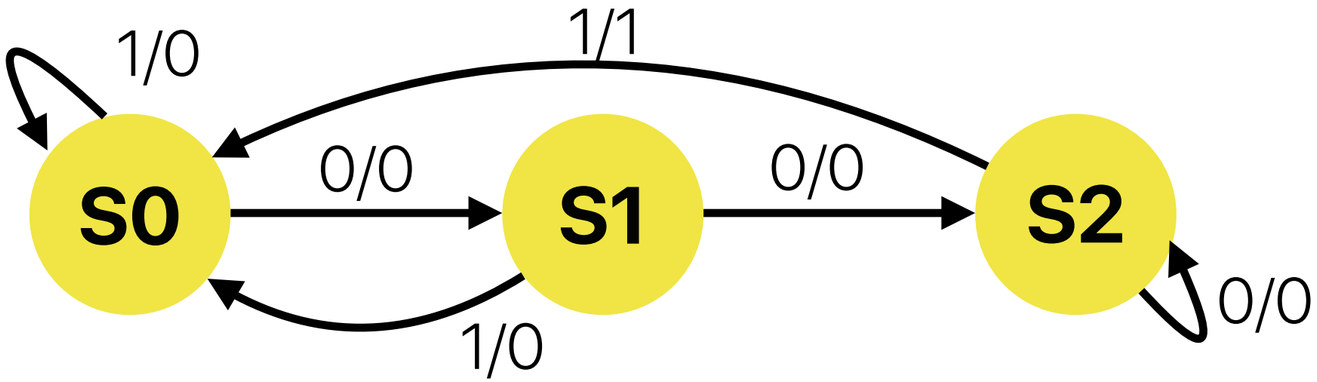
Current State	Next State, Output Input	
	0	1
S0	S1, 0	S0, 0
S1	S2, 0	S0, 0
S2	S2, 0	S0, 1

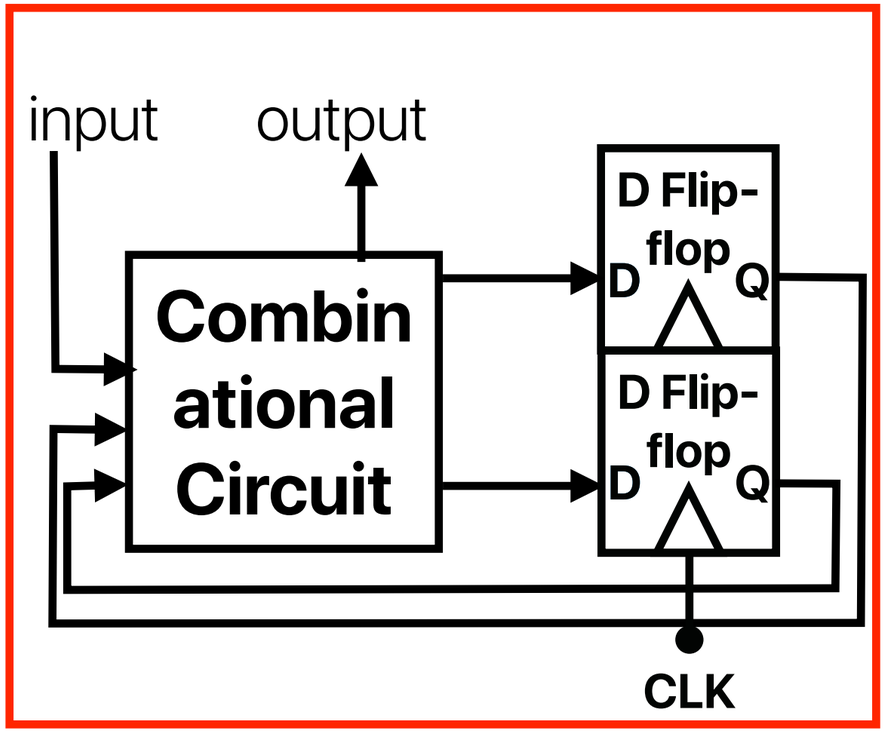
State Assignment

S0	00
S1	01
S2	10

Life on Mars

State Diagram





State Diagram

Current State	Next State, Output Input	
	0	1
S0	S1, 0	S0, 0
S1	S2, 0	S0, 0
S2	S2, 0	S0, 1

State Assignment

S0	00
S1	01
S2	10

State Truth Table

State\Input	0	1
00	01, 0	00, 0
01	10, 0	00, 0
10	10, 0	00, 1

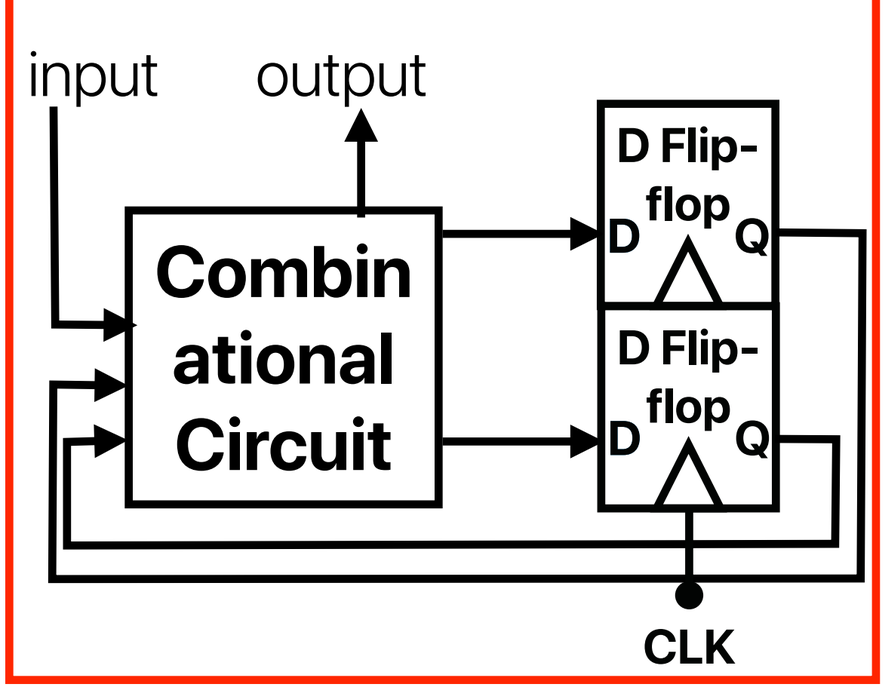
Life on Mars

State Truth Table

State\Input	0	1
00	01, 0	00, 0
01	10, 0	00, 0
10	10, 0	00, 1

Excitation Table

NextStateOput StateInput	D1	D0	y
000	0	1	0
001	0	0	0
010	1	0	0
011	0	0	0
100	1	0	0
101	0	0	1
110	X	X	X
111	X	X	X



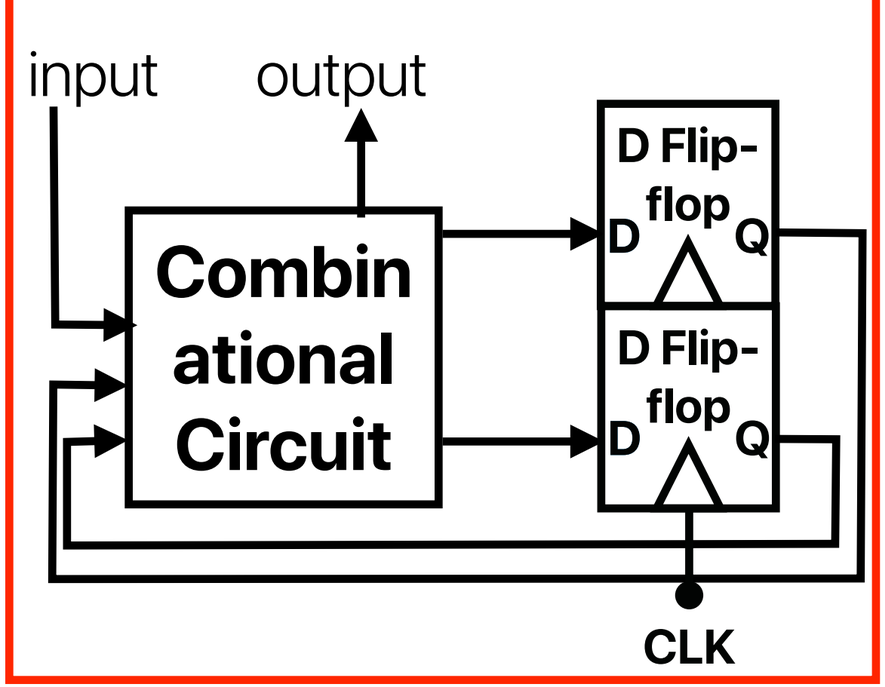
Life on Mars

State Truth Table

State\Input	0	1
00	01, 0	00, 0
01	10, 0	00, 0
10	10, 0	00, 1

Excitation Table

NextStateOput StateInput	D1	D0	y
000	0	1	0
001	0	0	0
010	1	0	0
011	0	0	0
100	1	0	0
101	0	0	1
110	X	X	X
111	X	X	X



K-Map — D1

	0,0	0,1	1,1	1,0
0	0	1	X	1
1	0	0	X	0

$D1 = x'Q_0 + x'Q_1$

K-Map — D0

	0,0	0,1	1,1	1,0
0	1	0	X	0
1	0	0	X	0

$D0 = Q_0'Q_1'x'$

K-Map — y

	0,0	0,1	1,1	1,0
0	0	0	X	0
1	0	0	X	1

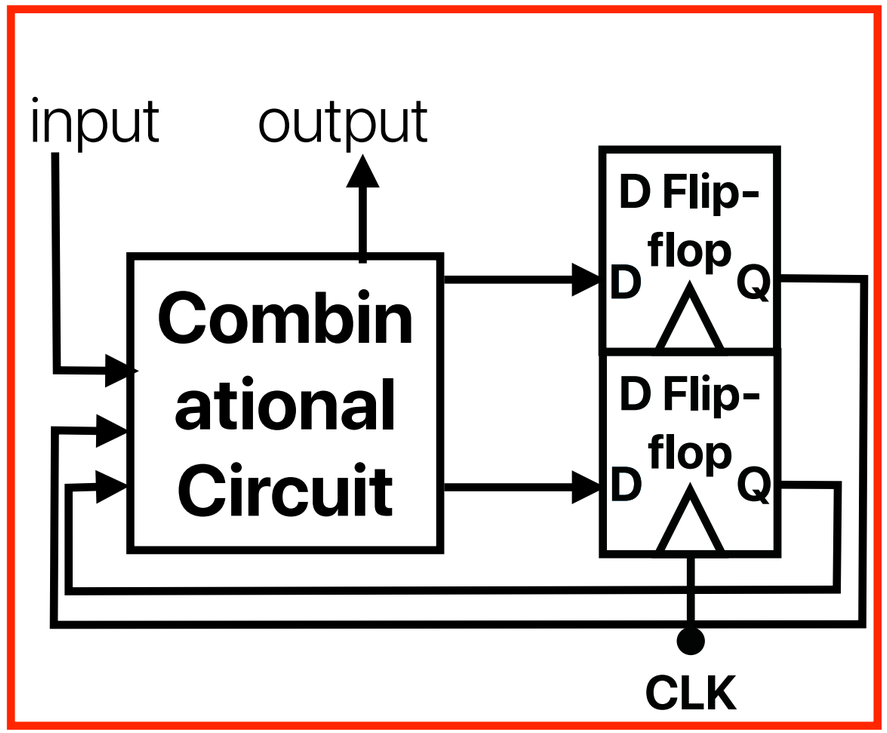
$y = Q_1'x$

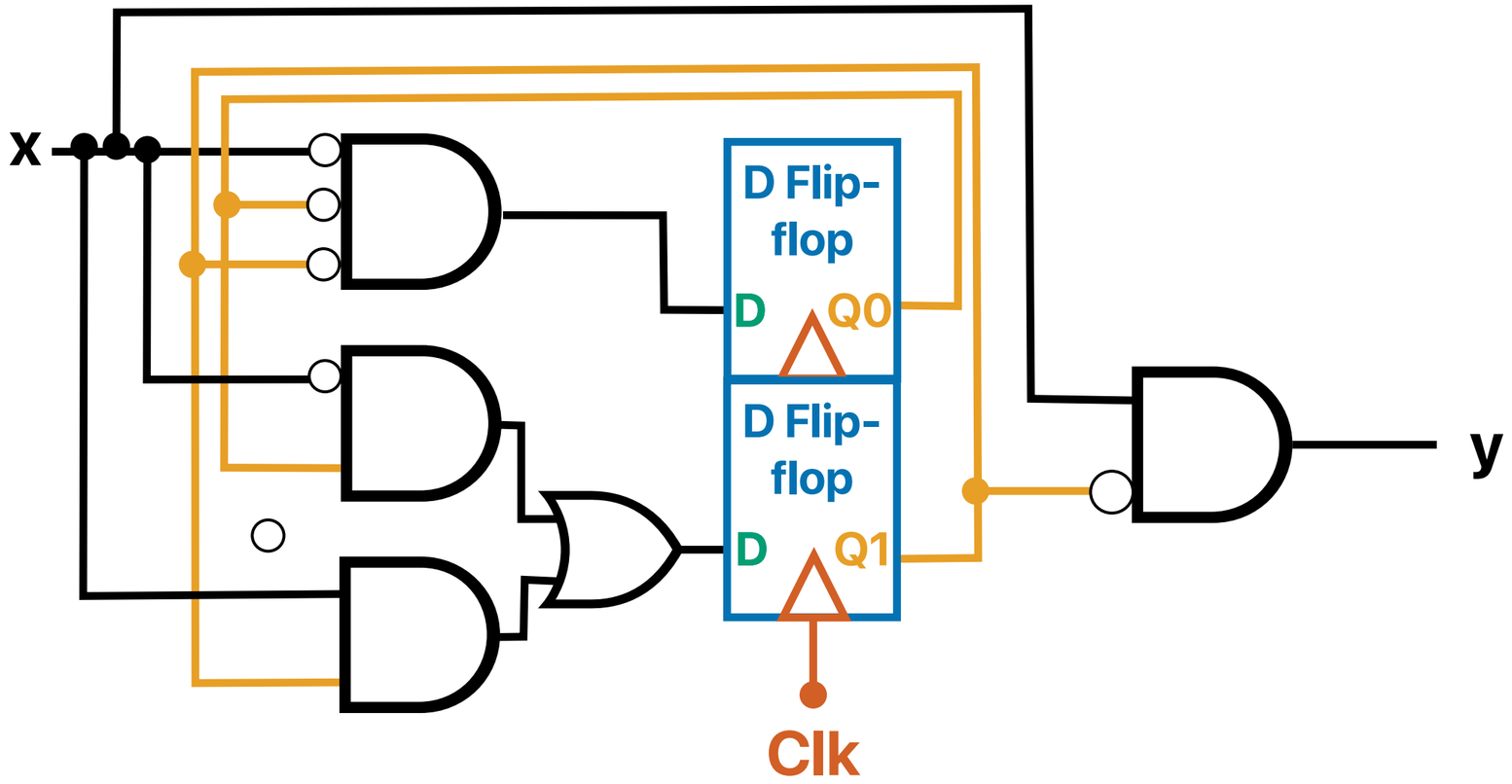
Circuit — Life on Mars

$D1 = x'Q_0 + x'Q_1$

$D0 = Q_0'Q_1'x'$

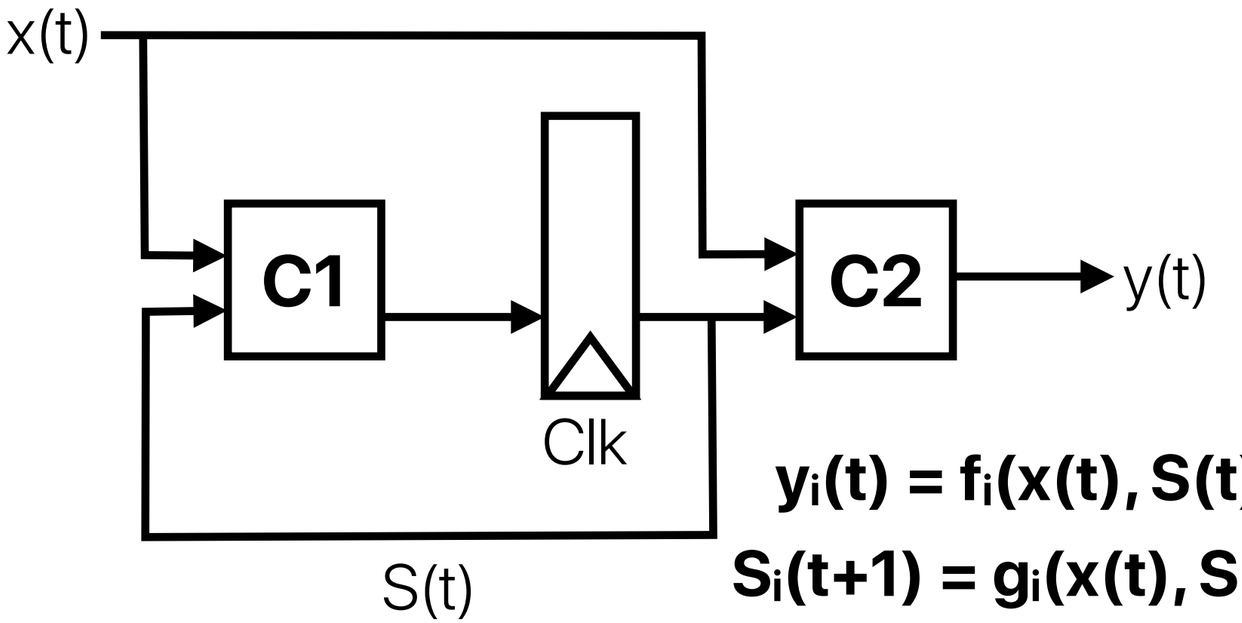
$y = Q_1'x$





Canonical Form: Mealy and Moore Machines

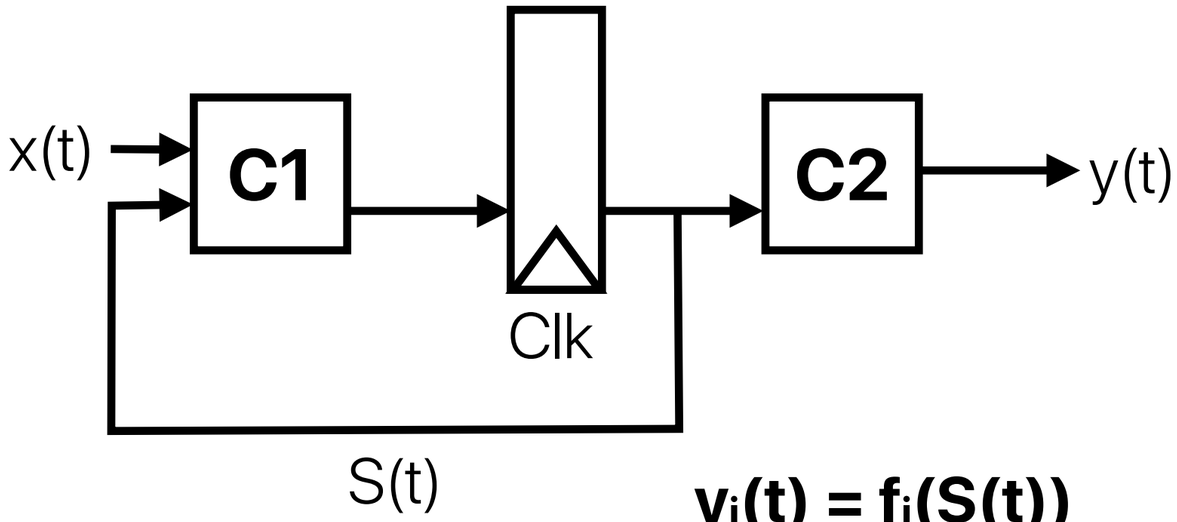
Mealy Machine



$$y_i(t) = f_i(x(t), S(t))$$

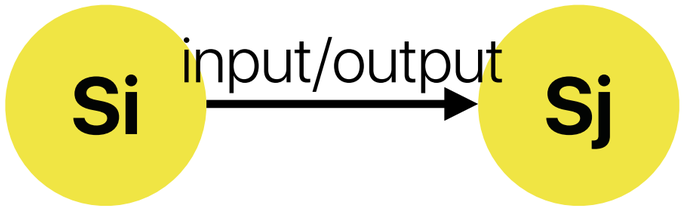
$$S_i(t+1) = g_i(x(t), S(t))$$

Moore Machine

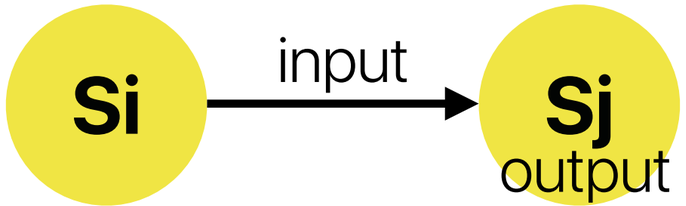


$$y_i(t) = f_i(S(t))$$

$$S_i(t+1) = g_i(x(t), S(t))$$



Result depends on both input and the current state



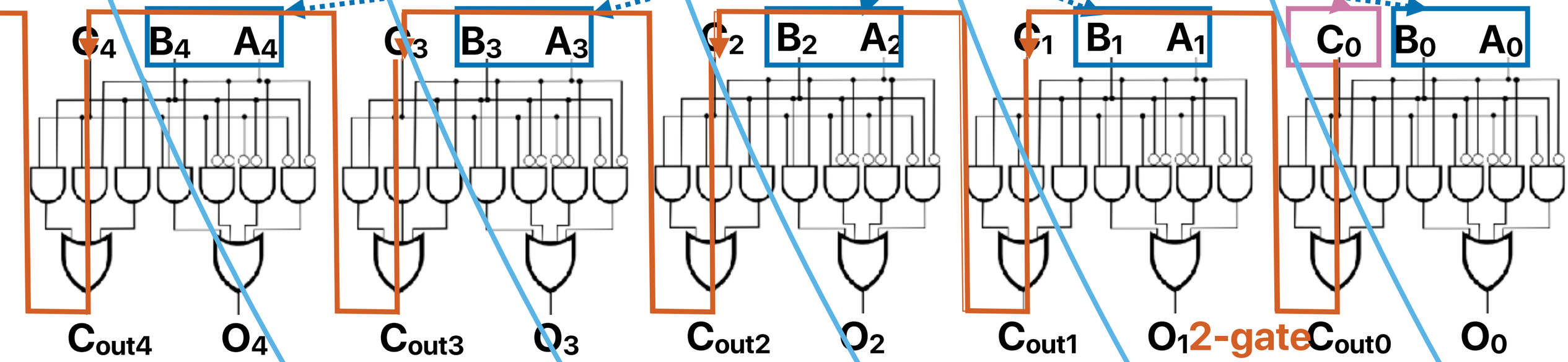
Result only depends on the current state

When sequential circuits meet datapath components (3)

The delay is determined by the "critical path"

Available in the very beginning

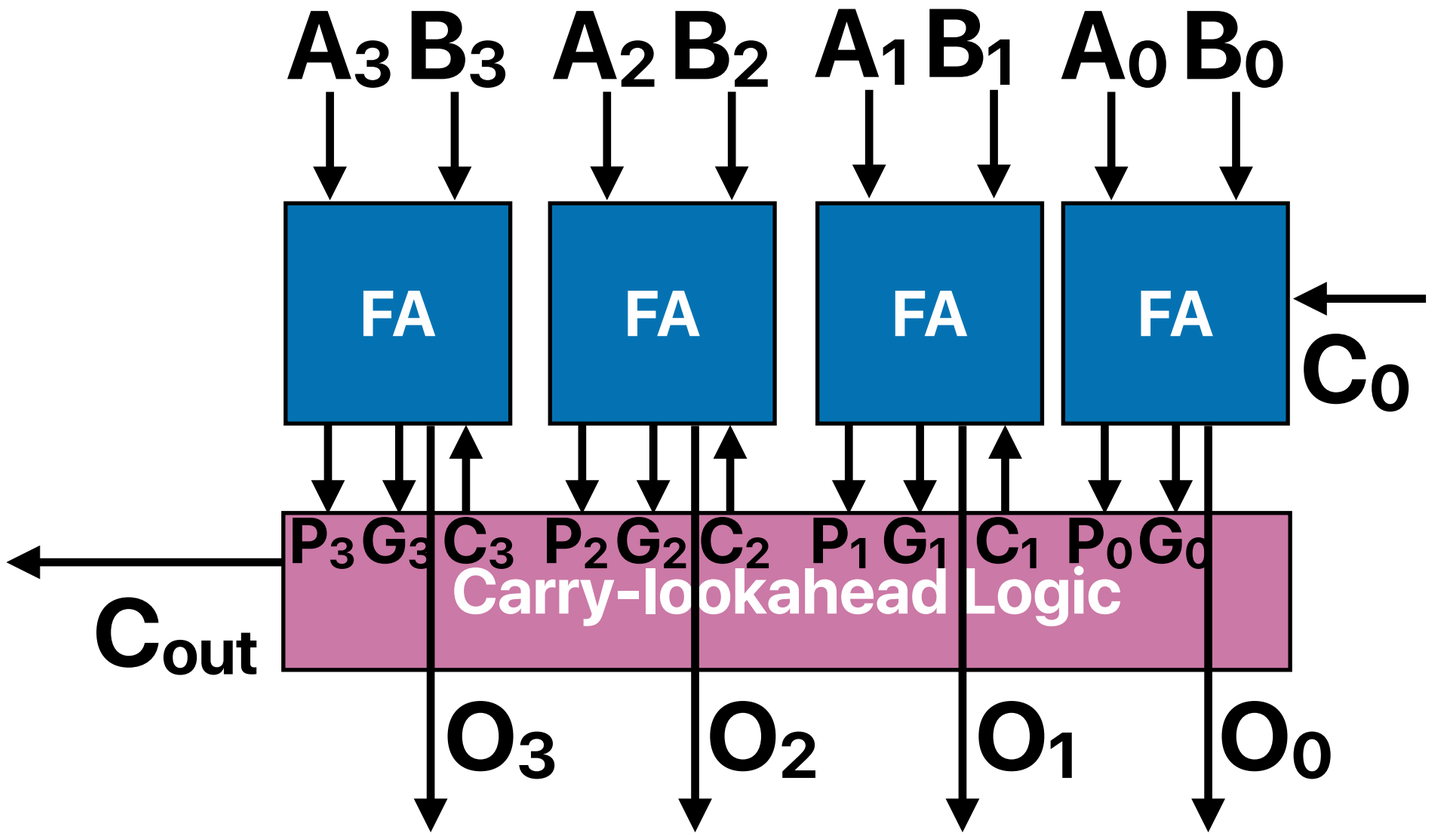
Only this is available in the beginning



Carry-Ripple Adder

CLA (cont.)

- All "G" and "P" are immediately available (only need to look over Ai and Bi), but "c" are not (except the c0).



$$G_i = A_i B_i$$

$$P_i = A_i \text{ XOR } B_i$$

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2$$
$$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

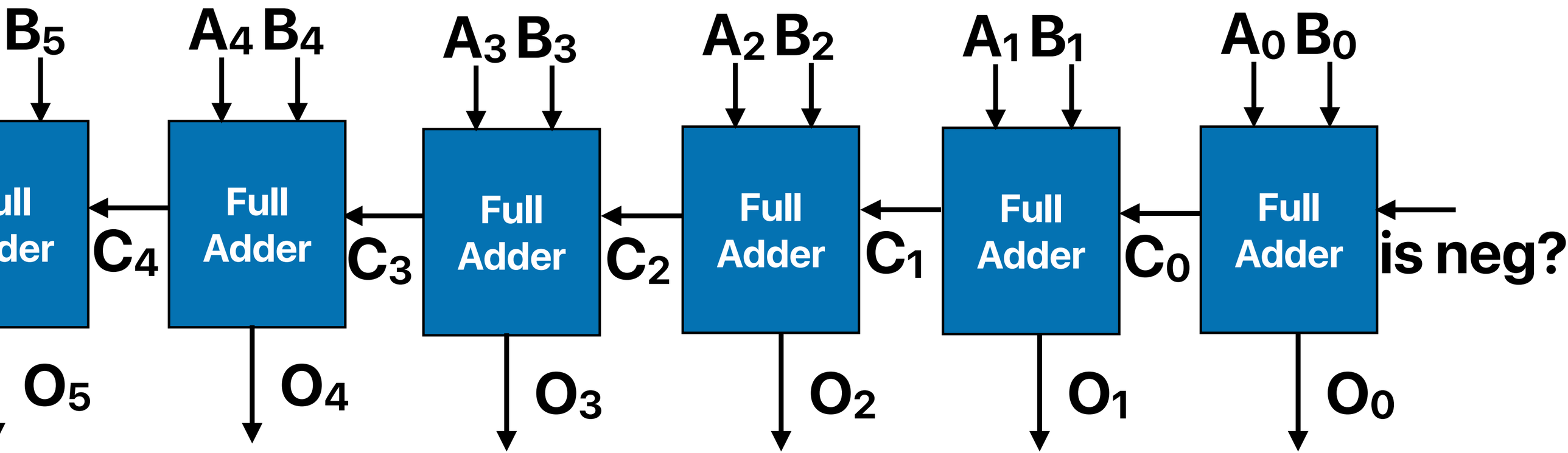
$$C_4 = G_3 + P_3 C_3$$
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$

CLA v.s. Carry-ripple

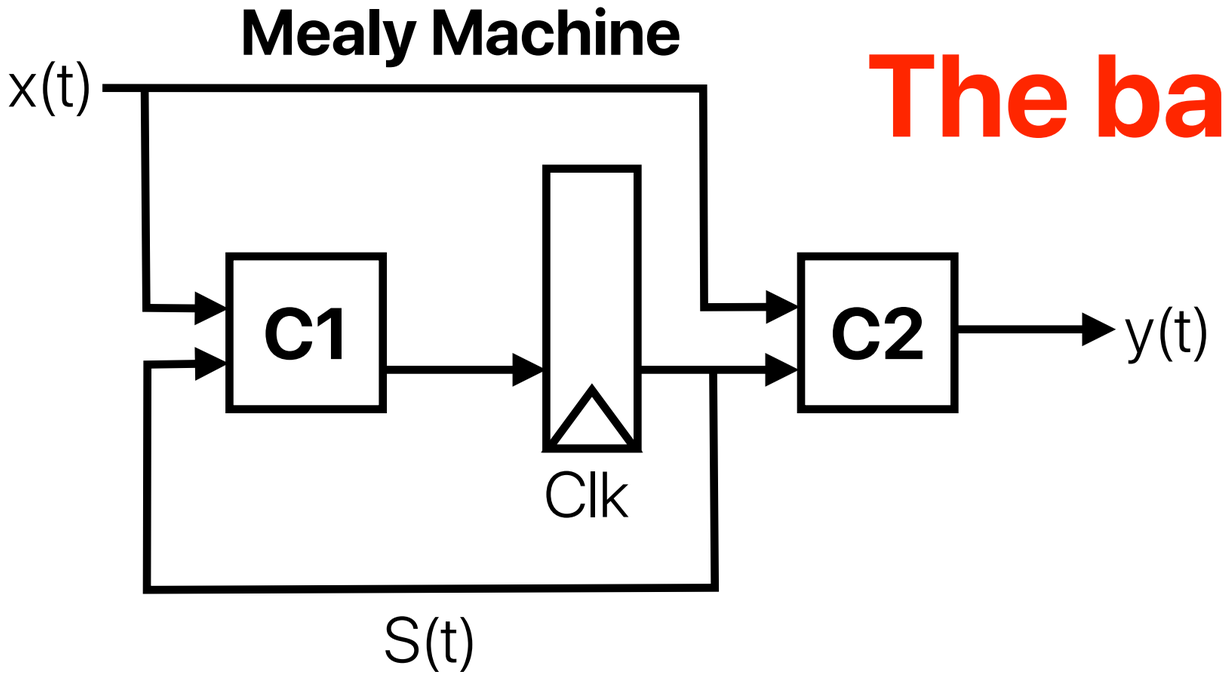
- Size:
  - 32-bit CLA with 4-bit CLAs — requires 8 of 4-bit CLA
    - Each requires 116 for the CLA 4*(4*6+8) for the A+B — 244 gates
    - 1952 transistors
  - 32-bit CRA
    - 1600 transistors **Win!**
- Delay
  - 32-bit CLA with 8 4-bit CLAs
    - 2 gates * 8 = 16 **Win!**
  - 32-bit CRA
    - 64 gates

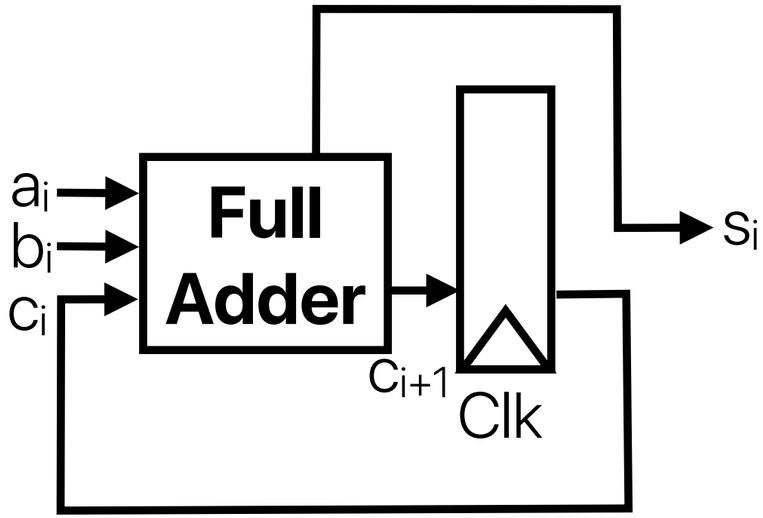
Area-Delay Trade-off!

Adder


B5
A4 B4
A3 B3
A2 B2
A1 B1
A0 B0
ull
der
Full Adder
Full Adder
Full Adder
Full Adder
Full Adder
C4
C3
C2
C1
C0
is neg?
O5
O4
O3
O2
O1
O0

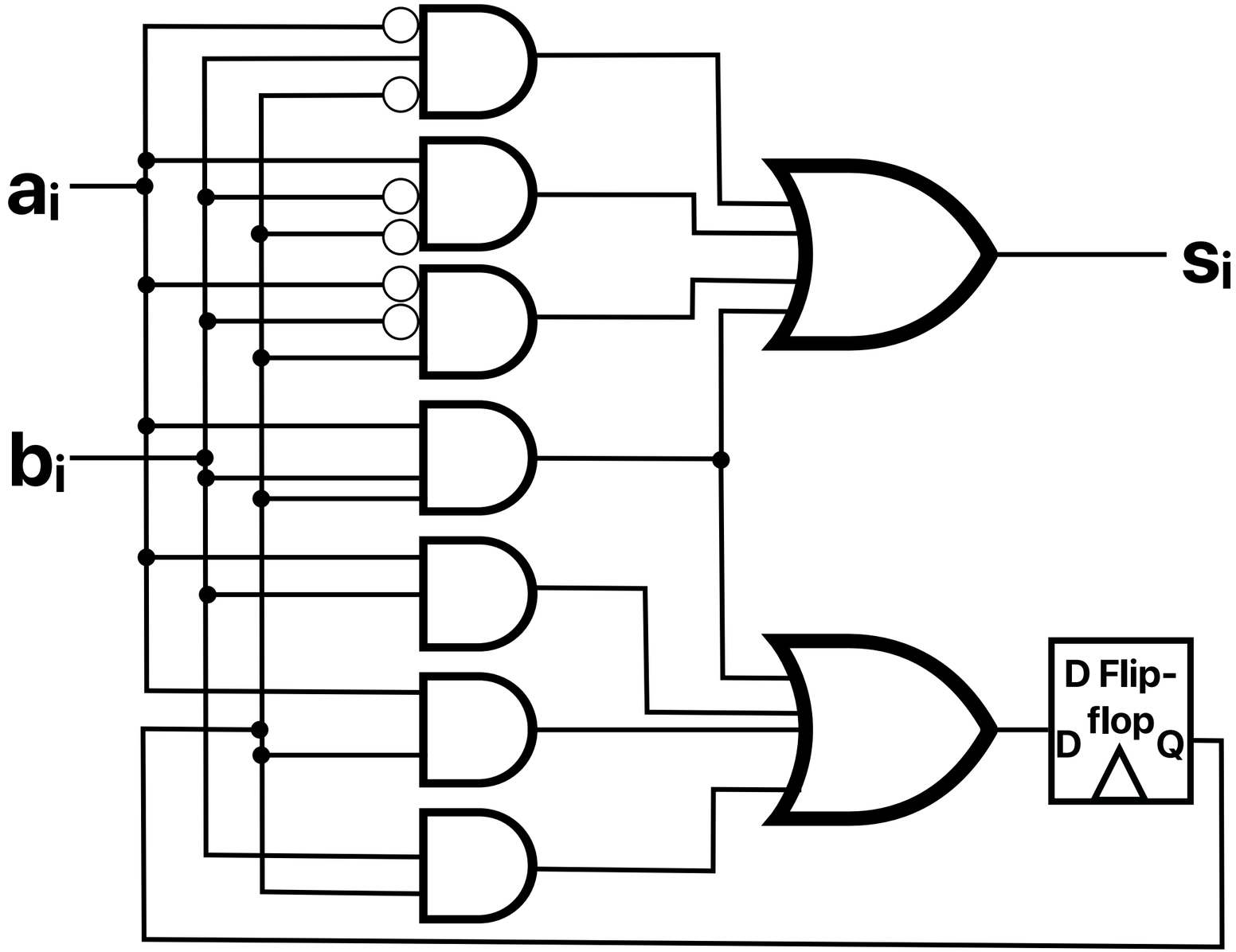
The basic idea





Excitation Table of Serial Adder

a_i	b_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Area/Delay of adders

- Consider the following adders?
  ① 32-bit CLA made with 8 4-bit CLA adders
  ② 32-bit CRA made with 32 full adders
  ③ 32-bit serial adders made with 4-bit CLA adders
  ④ 32-bit serial adders made with 1-bit full adders

Each CLA — 2-gate delay — 8*2+1 ~ 17

Each carry — 2-gate delay — 64

Each CLA — (3-gate delay + 2-gate delay)*8 cycles — 5*8+1 = 41

Each CLA — (2-gate delay + 2-gate delay)*32 cycles — 4*32 = 128

A. Area: (1) > (2) > (3) > (4) Delay: (1) < (2) < (3) < (4)
B. Area: (1) > (3) > (2) > (4) Delay: (1) < (3) < (2) < (4)
C. Area: (1) > (3) > (4) > (2) Delay: (1) < (3) < (4) < (2)
D. Area: (1) > (2) > (3) > (4) Delay: (1) < (3) < (2) < (4)
E. Area: (1) > (3) > (2) > (4) Delay: (1) < (3) < (4) < (2)

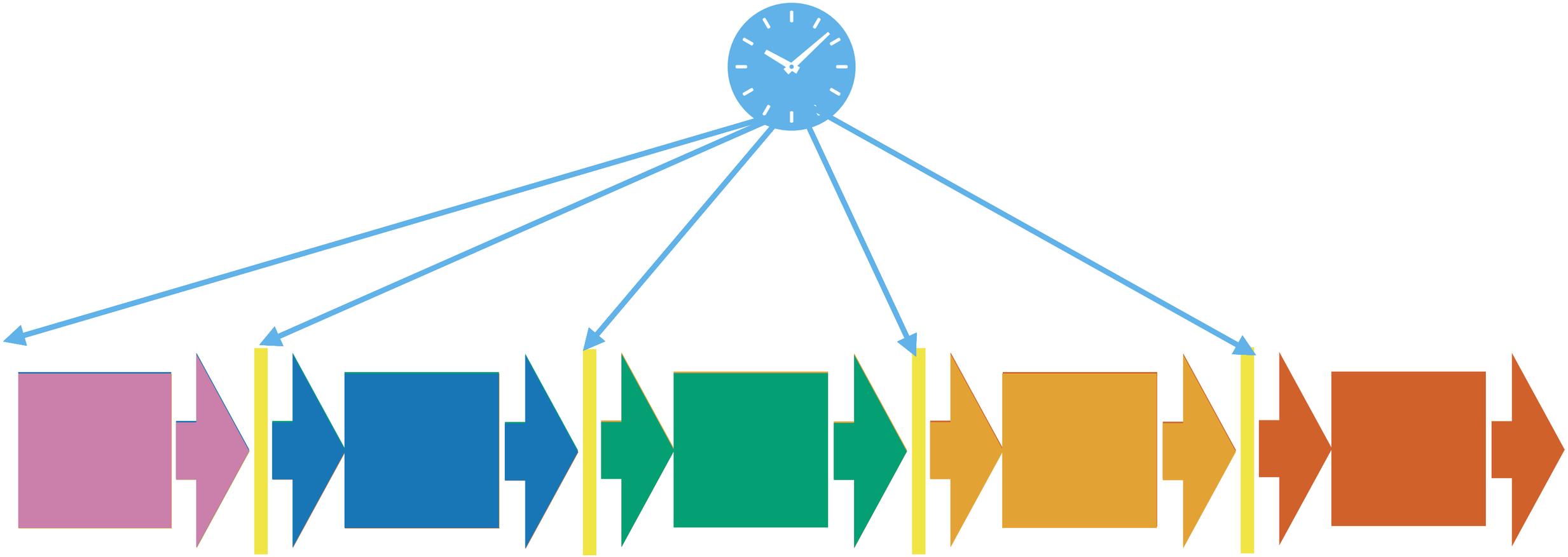
Frequency

- Consider the following adders. Assume each gate delay is 1ns and the delay in a register is 2ns. Please rank their maximum operating frequencies
  - ① 32-bit CLA made with 8 4-bit CLA adders $\frac{1}{17ns} = 58.8MHz$
  - ② 32-bit CRA made with 32 full adders $\frac{1}{64ns} = 15.6MHz$
  - ③ 32-bit serial adders made with 4-bit CLA adders $\frac{1}{5ns} = 200MHz$
  - ④ 32-bit serial adders made with 1-bit full adders $\frac{1}{4ns} = 250MHz$
  - A. (1) > (2) > (3) > (4)
  - B. (2) > (1) > (4) > (3)
  - C. (2) > (1) > (3) > (4)
  - D. (4) > (3) > (2) > (1)
  - E. (4) > (3) > (1) > (2)

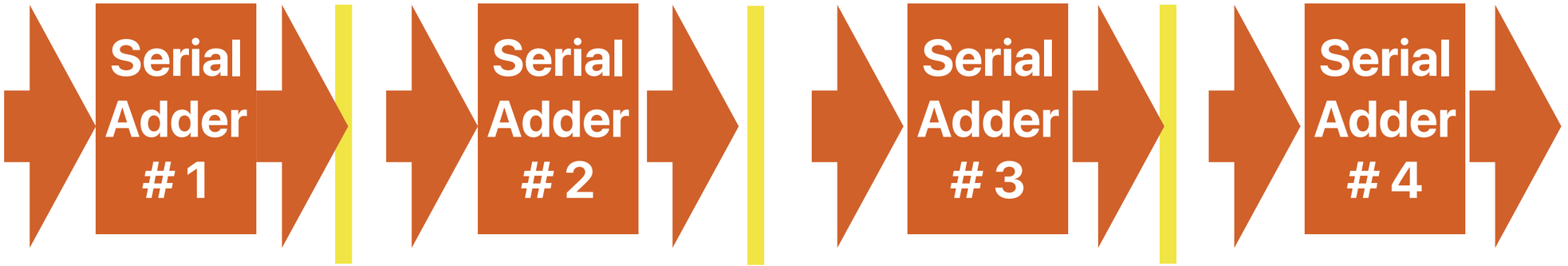
Pipelining

- Different parts of the hardware works on different requests/ commands simultaneously
- A clock signal controls and synchronize the beginning and the end of each part/**stage** of the work
- A **pipeline register** between different parts of the hardware to keep intermediate results necessary for the upcoming work
  - Register is basically an array of flip-flops!

Pipelining

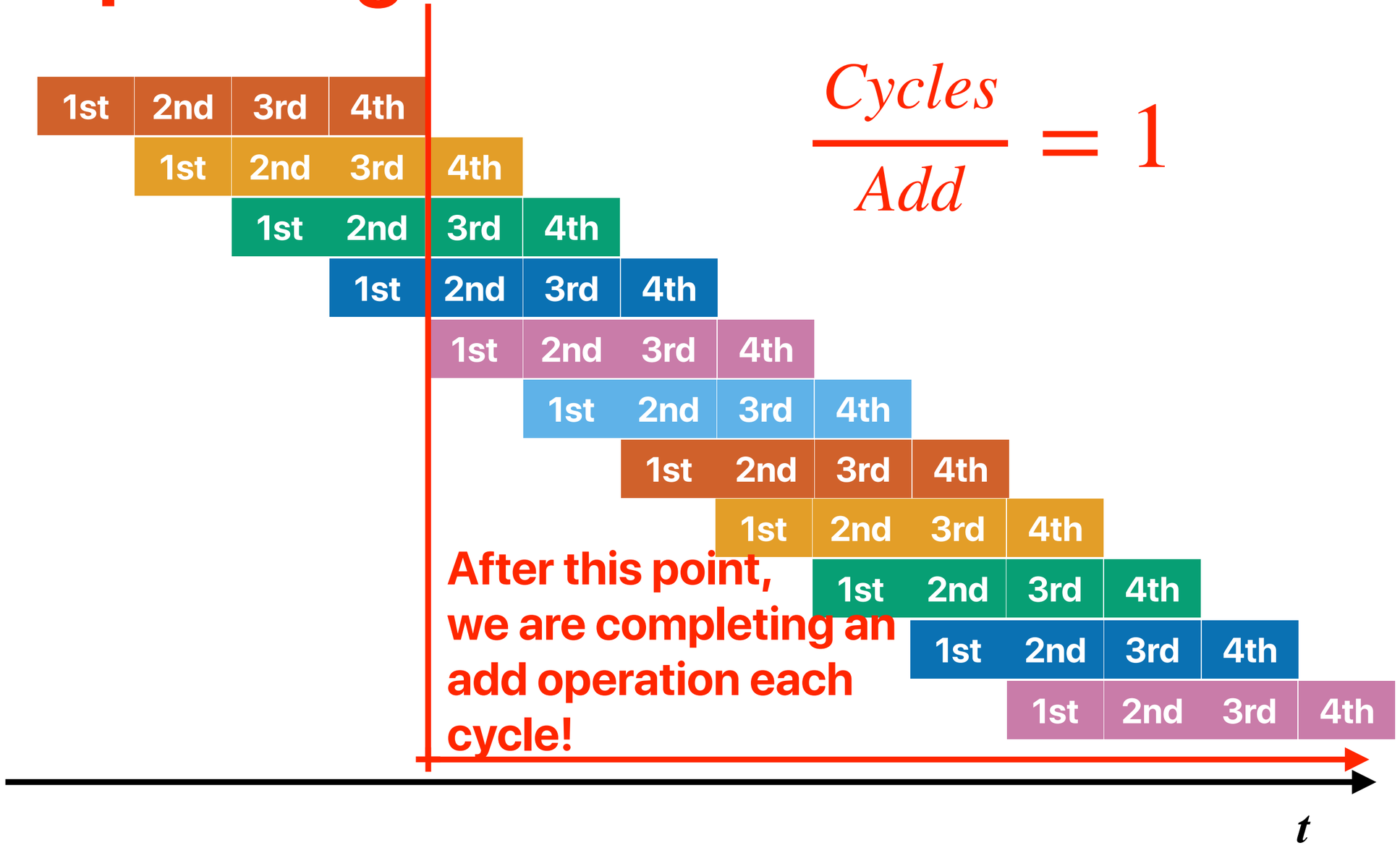


Pipelining a 4-bit serial adder



Pipelining a 4-bit serial adder

```
add a, b
add c, d
add e, f
add g, h
add i, j
add k, l
add m, n
add o, p
add q, r
add s, t
add u, v
```



$$\frac{Cycles}{Add} = 1$$

Throughput

- Consider the following adders. Assume each gate delay is 1ns and the delay in a register is also 1ns. And we are processing millions of add operations. Please rank their throughputs.
  - ① 32-bit CLA made with 8 4-bit CLA adders
  - ② 32-bit CRA made with 32 full adders
  - ③ 8-stage, pipelined 32-bit serial adders made with 4-bit CLA adders
  - ④ 32-stage, pipelined 32-bit serial adders made with 1-bit full adders
  - A. (1) > (2) > (3) > (4)
  - B. (2) > (1) > (4) > (3)
  - C. (3) > (4) > (2) > (1)
  - D. (4) > (3) > (2) > (1)
  - E. (4) > (3) > (1) > (2)

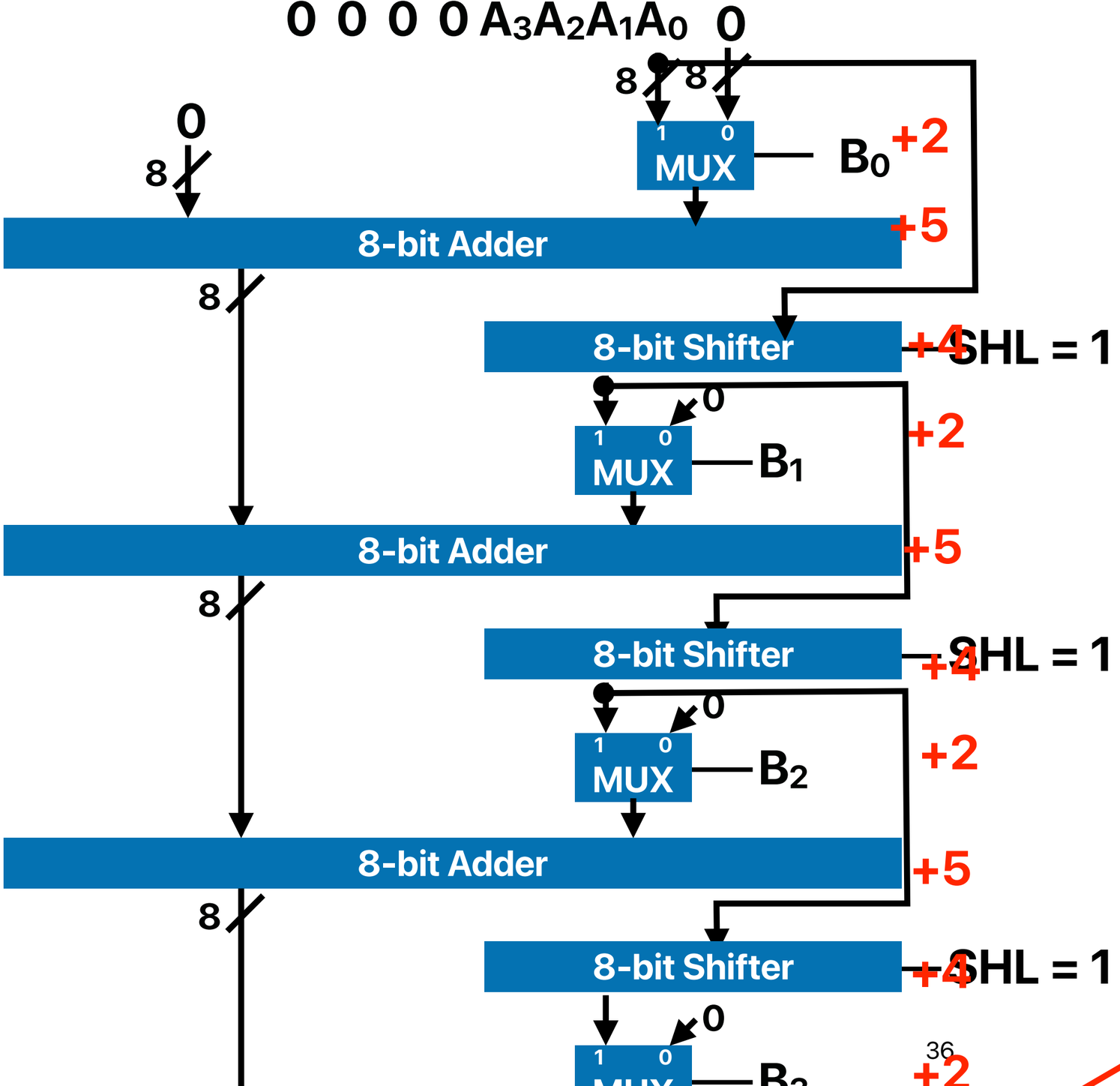
Latency/Delay v.s. Bandwidth/Throughput

- Latency — the amount of time to finish an operation
  - access time
  - response time
- Throughput — the amount of work can be done within a given period of time
  - bandwidth (MB/Sec, GB/Sec, Mbps, Gbps)
  - IOPs
  - MFLOPs

Latency/Delay v.s. Throughput

	Toyota Prius	100 Gb Network
	•100 miles (161 km) from UCSD •75 MPH on highway! •Max load: 374 kg = 2,770 hard drives (2TB per drive)	•100 miles (161 km) from UCSD •Lightspeed! — $3*10^8$m/sec •Max load:4 lanes operating at 25GHz
bandwidth	290GB/sec	100 Gb/s or 12.5GB/sec
latency	3.5 hours	2 Peta-byte over 167772 seconds = 1.94 Days
response time	You see nothing in the first 3.5 hours	You can start watching the movie as soon as you get a frame!

Multiplier

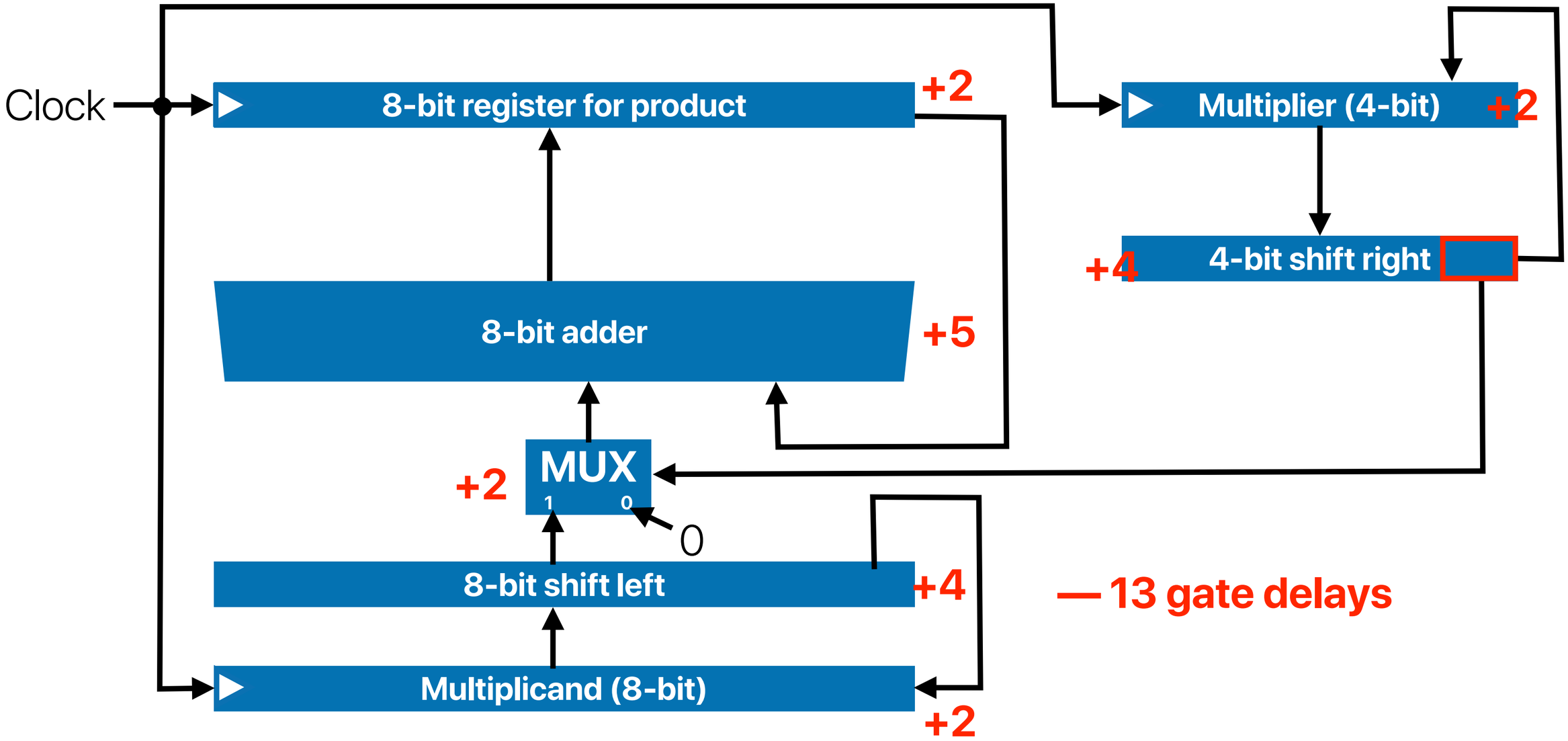


Shift and add

$$
\begin{array}{rrrrrrrr}
 & & & & a_3 & a_2 & a_1 & a_0 \\
 & & & & \times\ b_3 & b_2 & b_1 & b_0 \\
\hline
 & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & 0 \\
 & & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & 0 & 0 \\
 & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & 0 & 0 & 0 \\
\hline
p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

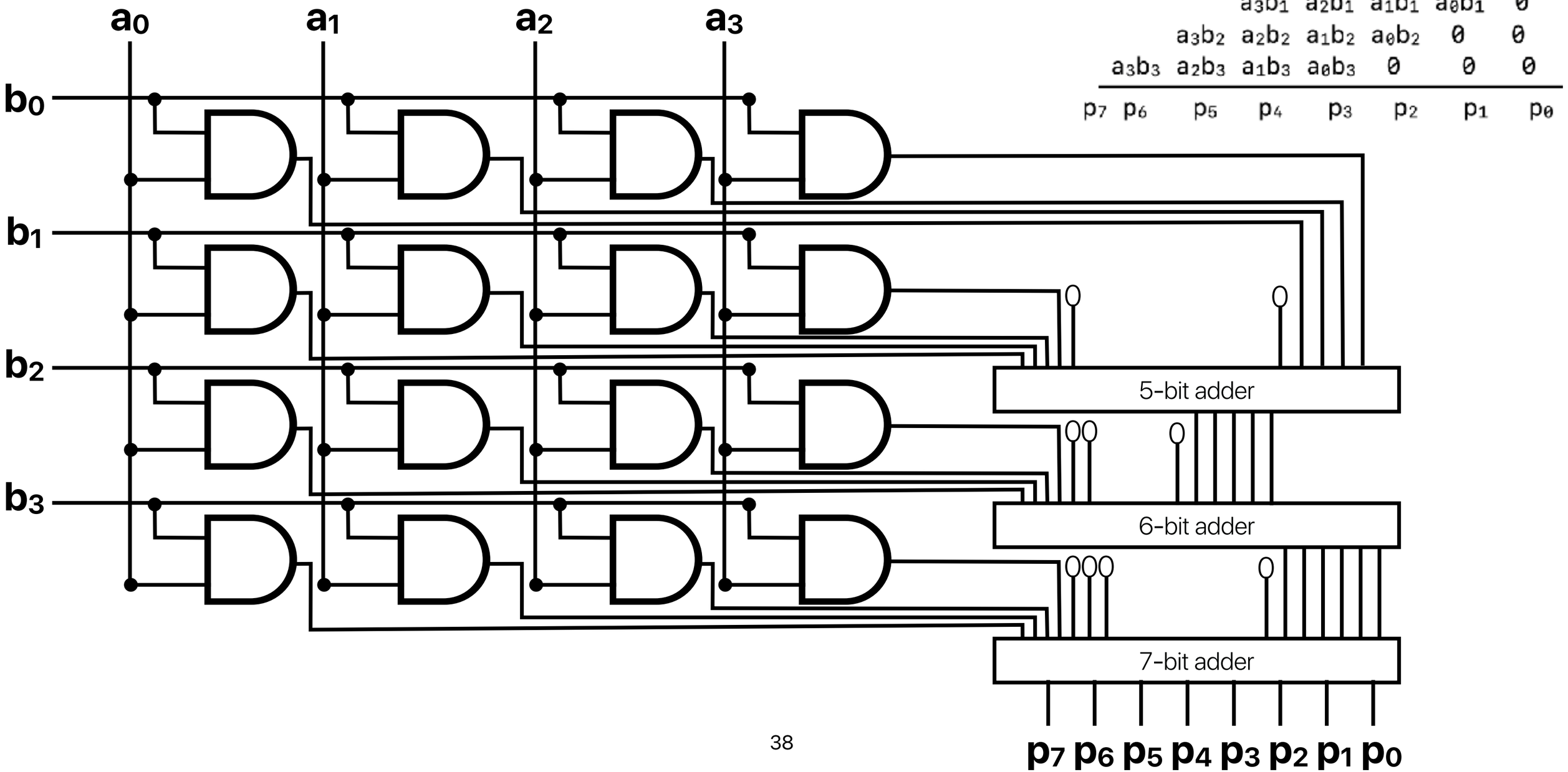
— 40 gate delays

4-bit serial shift-and-add multiplier



Array style

			a_3	a_2	a_1	a_0	
			× b_3	b_2	b_1	b_0	
			a_3b_0	a_2b_0	a_1b_0	a_0b_0	
		a_3b_1	a_2b_1	a_1b_1	a_0b_1	0	
	a_3b_2	a_2b_2	a_1b_2	a_0b_2	0	0	
a_3b_3	a_2b_3	a_1b_3	a_0b_3	0	0	0	
p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0



Gate-delays of 32-bit array-style multipliers

- What's the estimated gate-delay of a 32-bit multiplier? (Assume adders are composed of 4-bit CLAs)
  - A. 0 — 100
  - B. 100 — 500
  - C. 500 — 1000
  - D. 1000 — 1500
  - E. > 1500

Each n-bit adder is roundup(n/4)*2+1

We need 33-64 bit adders

33 - 36 -bit adders —> (9*2+1) gate delays *4

37 - 40 -bit adders —> (10*2+1) gate delays *4

41 - 44 -bit adders —> (11*2+1) gate delays *4

45 - 48 -bit adders —> (12*2+1) gate delays *4

49 - 52 -bit adders —> (13*2+1) gate delays *4

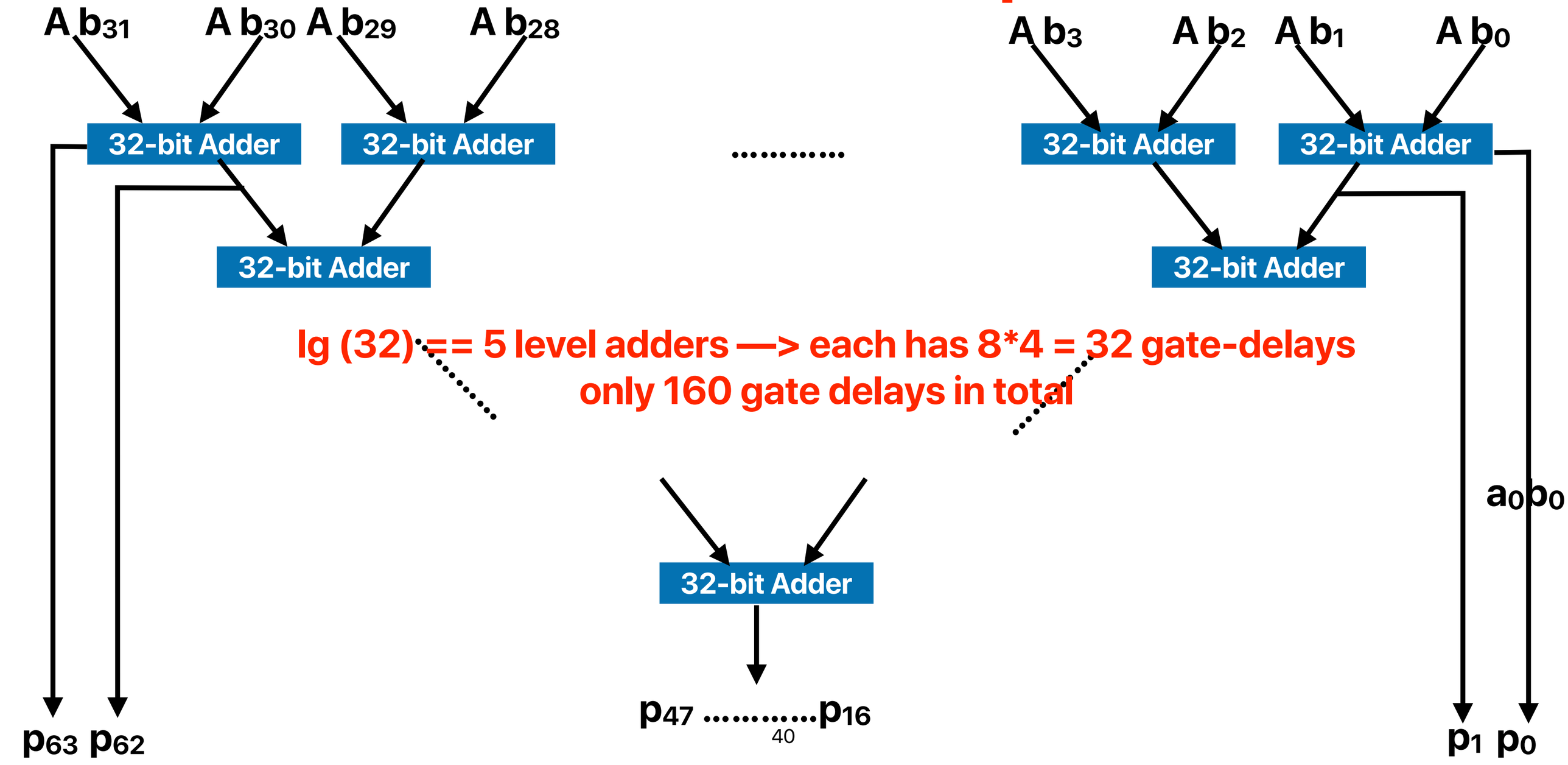
53 - 56 -bit adders —> (14*2+1) gate delays *4

57 - 60 -bit adders —> (15*2+1) gate delays *4

61 - 64 -bit adders —> (16*2+1) gate delays *4

4*2*(9+10+11+12+13+14+15+16+1) = 808

Parallel-tree Multiplier



Latency of multipliers

- Consider the following multipliers and assume each gate delay is 1ns and the delay in a register is 2ns. If all circuits can operate their maximum frequency, please identify the multiplier with shortest end-to-end latency in generating the result for multiplying two 32-bit numbers
  A. 32 32-bit shift and add multipliers — 39*64 = 2496 gate delays
  B. 32-bit array-style multipliers — 808 gate delays
  C. Pipelined 4-bit serial shift-and-add multiplier — 41*64 = 2624 gate delays

Throughput of multipliers

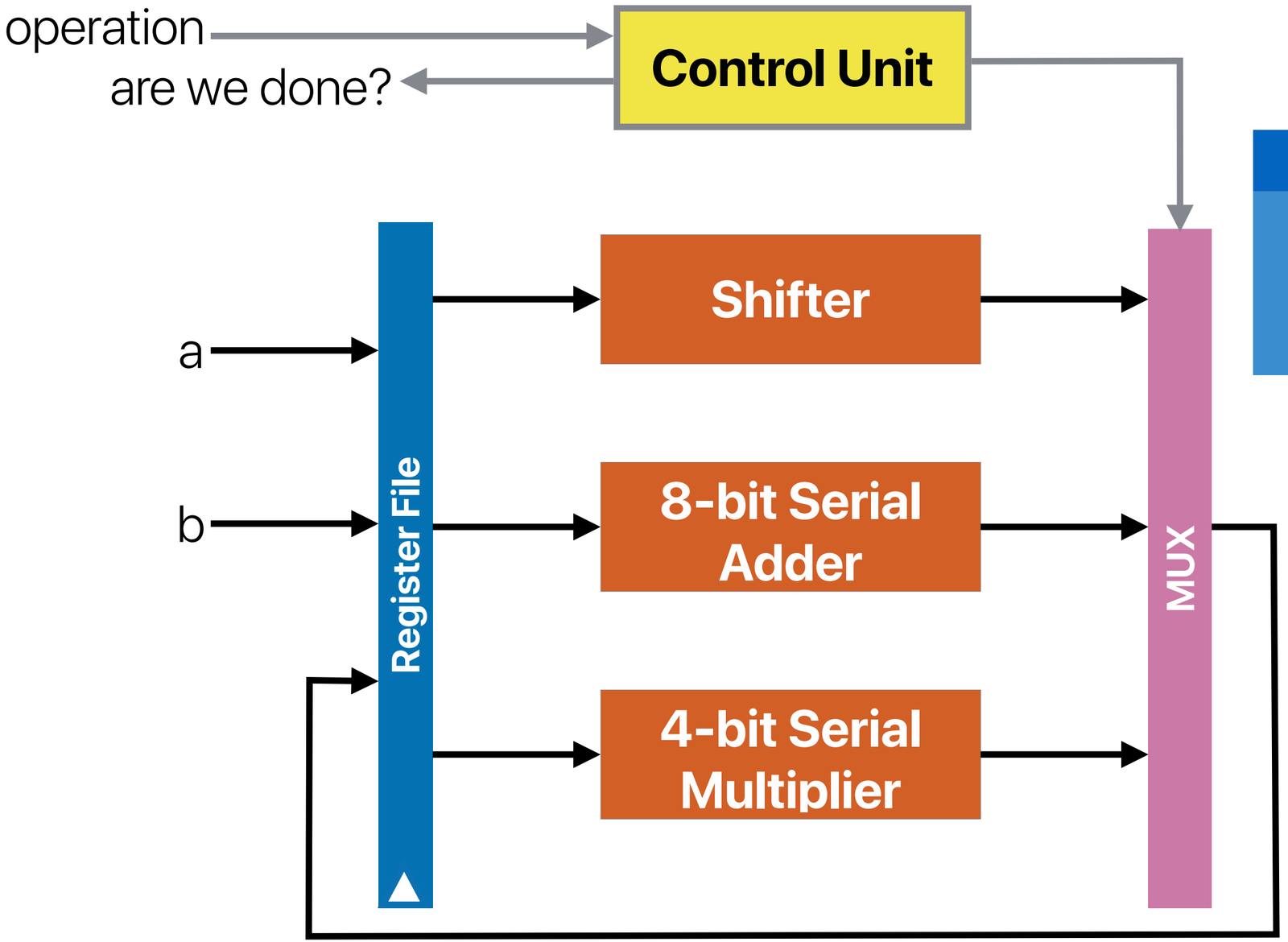
- Consider the following multipliers and assume each gate delay is 1ns and the delay in a register is 2ns. If all circuits can operate their maximum frequency, please identify the multiplier with shortest end-to-end latency in generating the result for multiplying two million pairs of 32-bit numbers
  A. 32-bit shift and add multipliers
  B. 32-bit array-style multipliers
  C. Pipelined 32-bit serial shift-and-add multiplier

Put everything all together

Let’s put all things together!

- We have learned all datapath components for an ALU!
  - Register
  - Shifter
  - Adders
  - Multiplier
- Processor has only one clock generator
  - Each datapath component has a different latency
  - We have make some of the above “serial”

Overview of the simple "processor"



Control Path

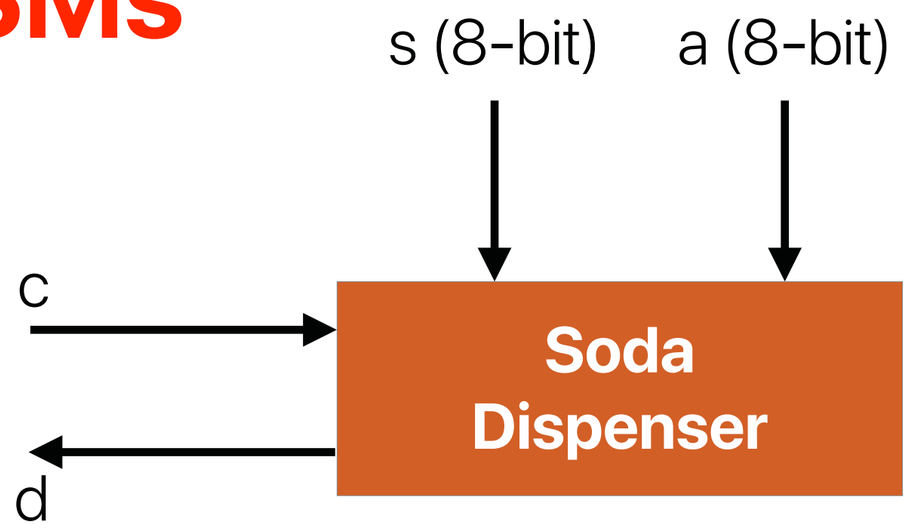
	cycles	mux
shift	1	0
add	4	1
mul	32	2

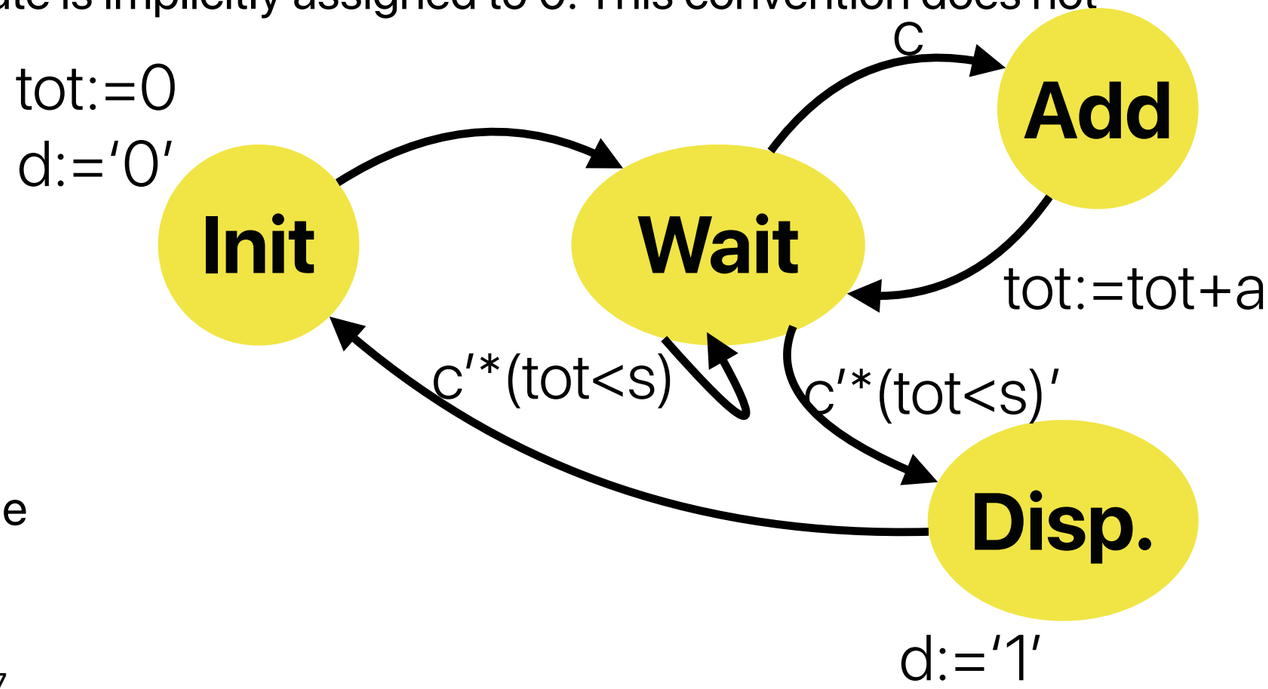
Datapath

HLSM — High-Level State Machine

Benefits of HLSMs

- High-level state machine (HLSM) extends FSM with:
  - Multi-bit input/output
  - Local storage
  - Arithmetic operations
- Conventions
  - Each transition is implicitly ANDed with a rising edge of the clock
  - Any bit output not explicitly assigned a value in a state is implicitly assigned to 0. This convention does not apply for multibit outputs
  - Every HLSM multibit output is registered
  - Numbers:
    - Single-bit: '0' (single quotes)
    - Integer: 0 (no quotes)
    - Multi-bit: "0000" (double quotes)
  - == for comparison equal
  - Multi-bit outputs must be registered via local storage
  - – // precedes a comment





RTL(Register Transfer Level) Design

RTL Design Process

- Step 1: Capture a high-level state machine
  - Describe the system's desired behavior as a high-level state machine. The state machine consists of states and transitions. The state machine is high level because the transition conditions and the state actions are more than just Boolean operations on single-bit input and outputs
  - Recommendations:
    - Always list all inputs, outputs and local registers on top of your HLSM diagram
    - Clearly specify the size in bits of each of them
    - On states: update the value of registers, update of outputs
    - On transitions: express conditions in terms of the HLSM inputs or state of the internal values and arithmetic operations between them.

RTL Design Process

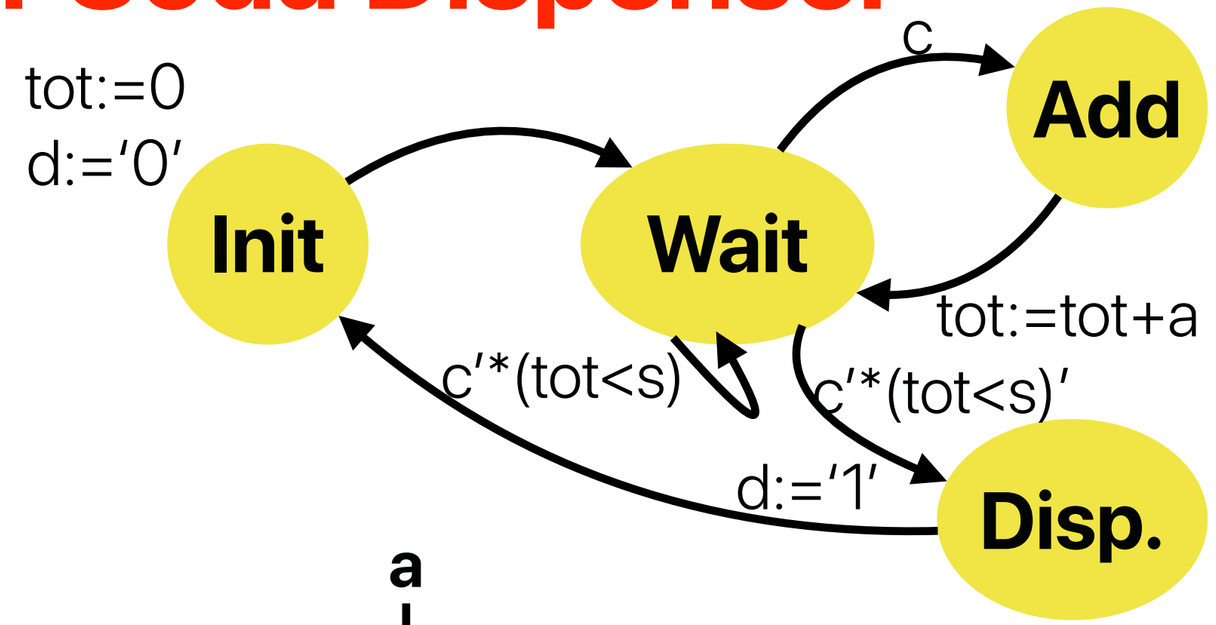
- Step 2: Convert it to a circuit
  - Create a datapath
    - Create a datapath to carry out the data operations of the high level state machine
    - Elements of your datapaths can be registers, adders, comparators, multipliers, dividers, etc.
  - Connect the datapath to a controller
    - Connect the datapath to a controller block.
    - Connect the external control inputs and outputs to the controller block.
    - Clearly label all control signals that are exchanged between the datapath and the controller
  - Derive the controller's FSM
    - Convert the high-level state machine to a finite state machine (FSM) for the controller, by replacing data operations with setting and reading of control signals to and from the datapath
- Final Step Implement the FSM as a state register and logic

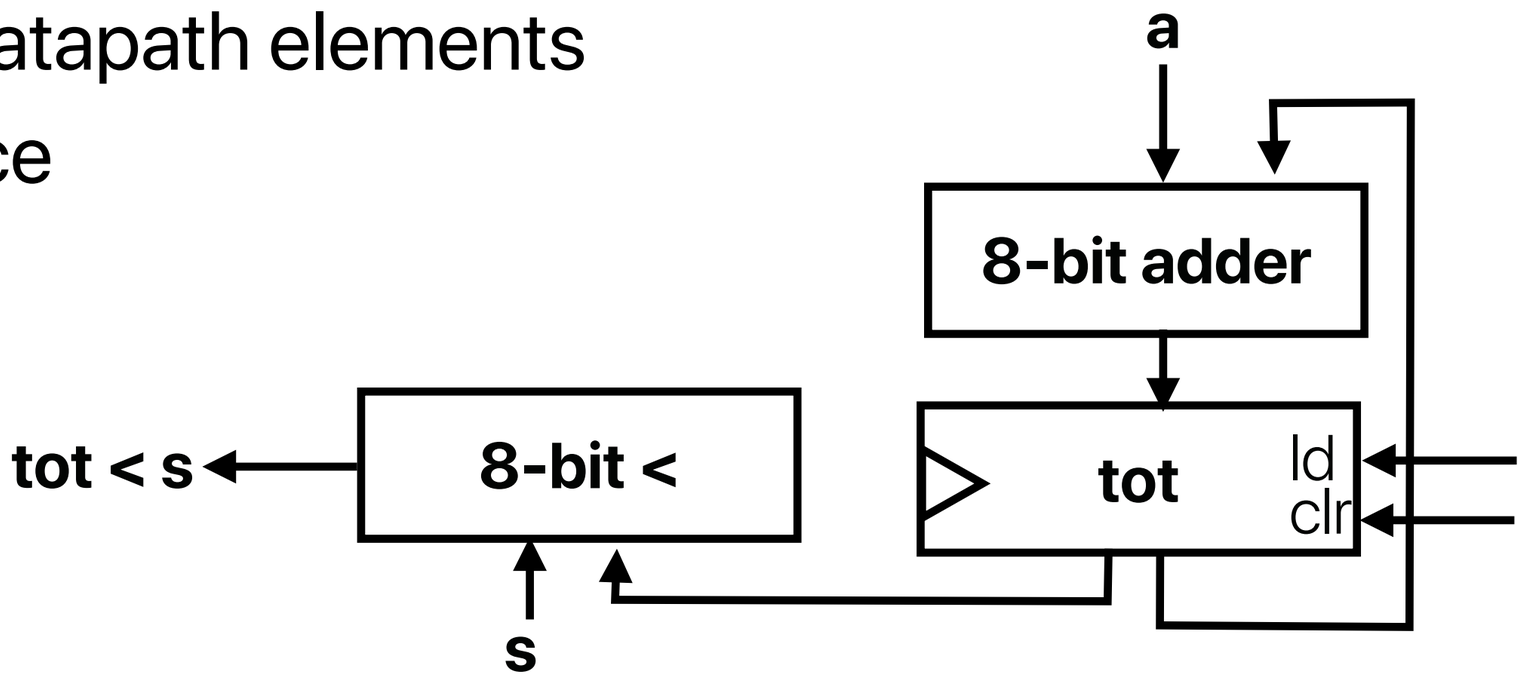
RTL Design Summary

- Capture the behavior with HLSM
- Convertit to a circuit
  - High-level architecture (datapath and control path)
  - Datapath capable of HLSM's data operations
  - Design controller to control the datapath

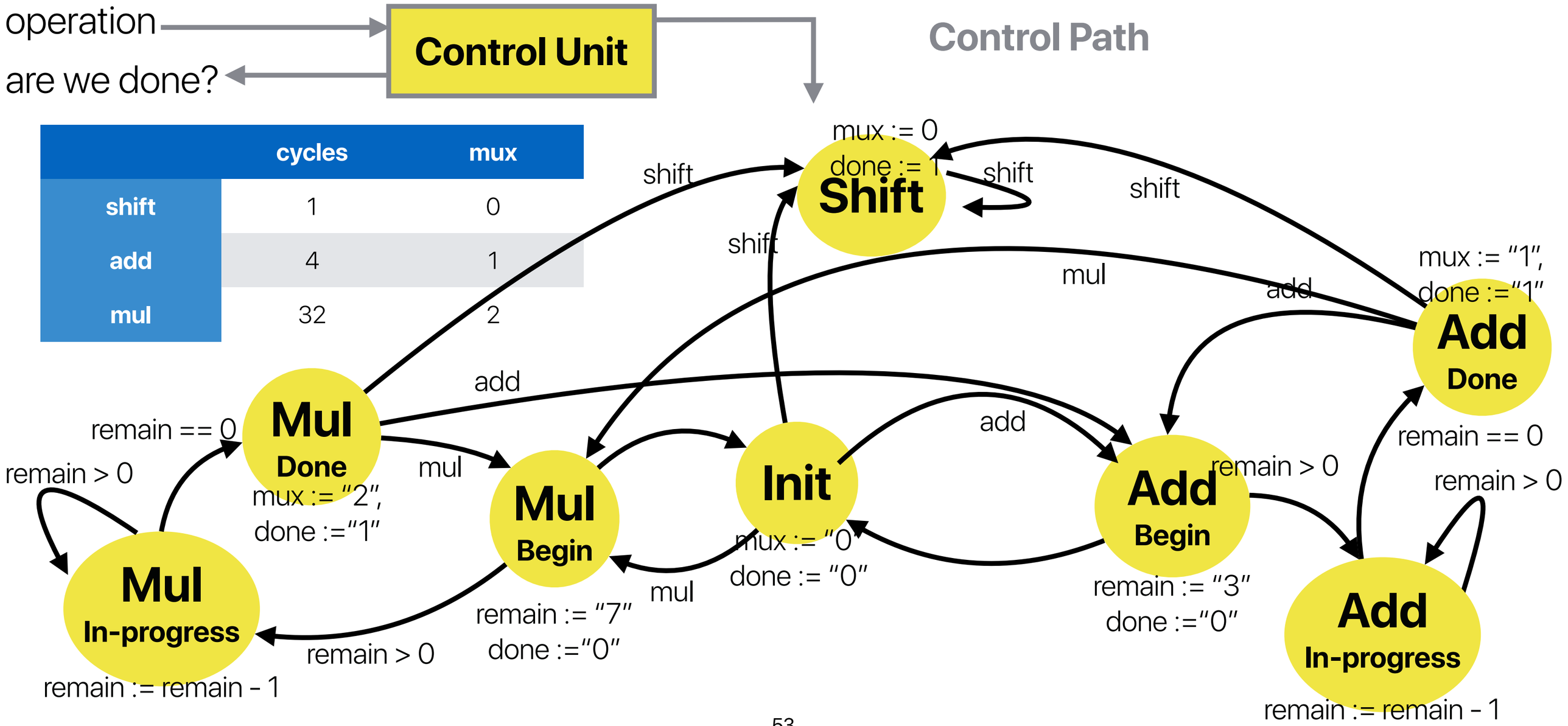
Create Datapath for Soda Dispenser

- Register: tot
- Comparator: to compare tot and s
- Adder: to update tot = tot + a
- Connect datapath elements
- I/O interface





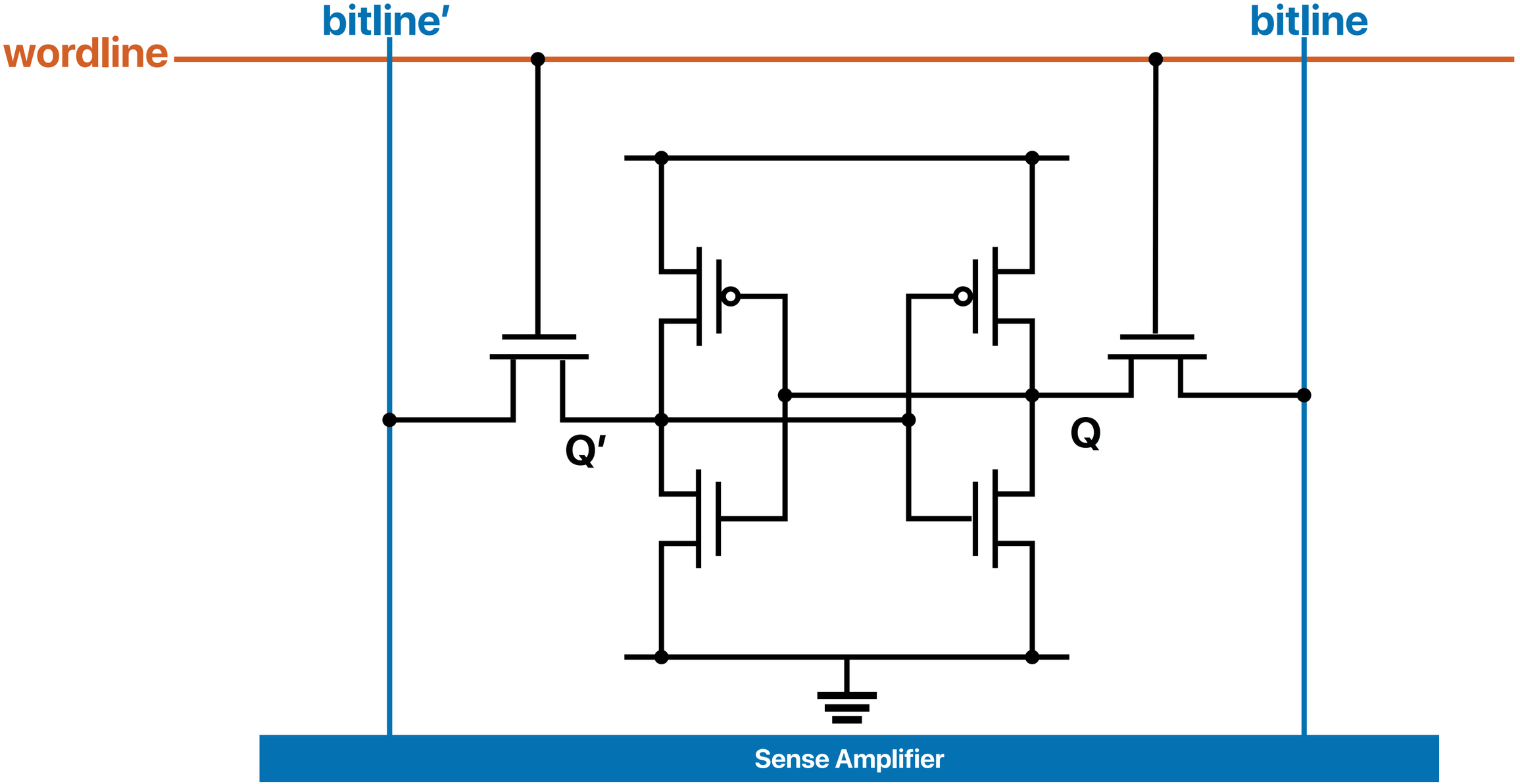
The HLSM for the processor's control



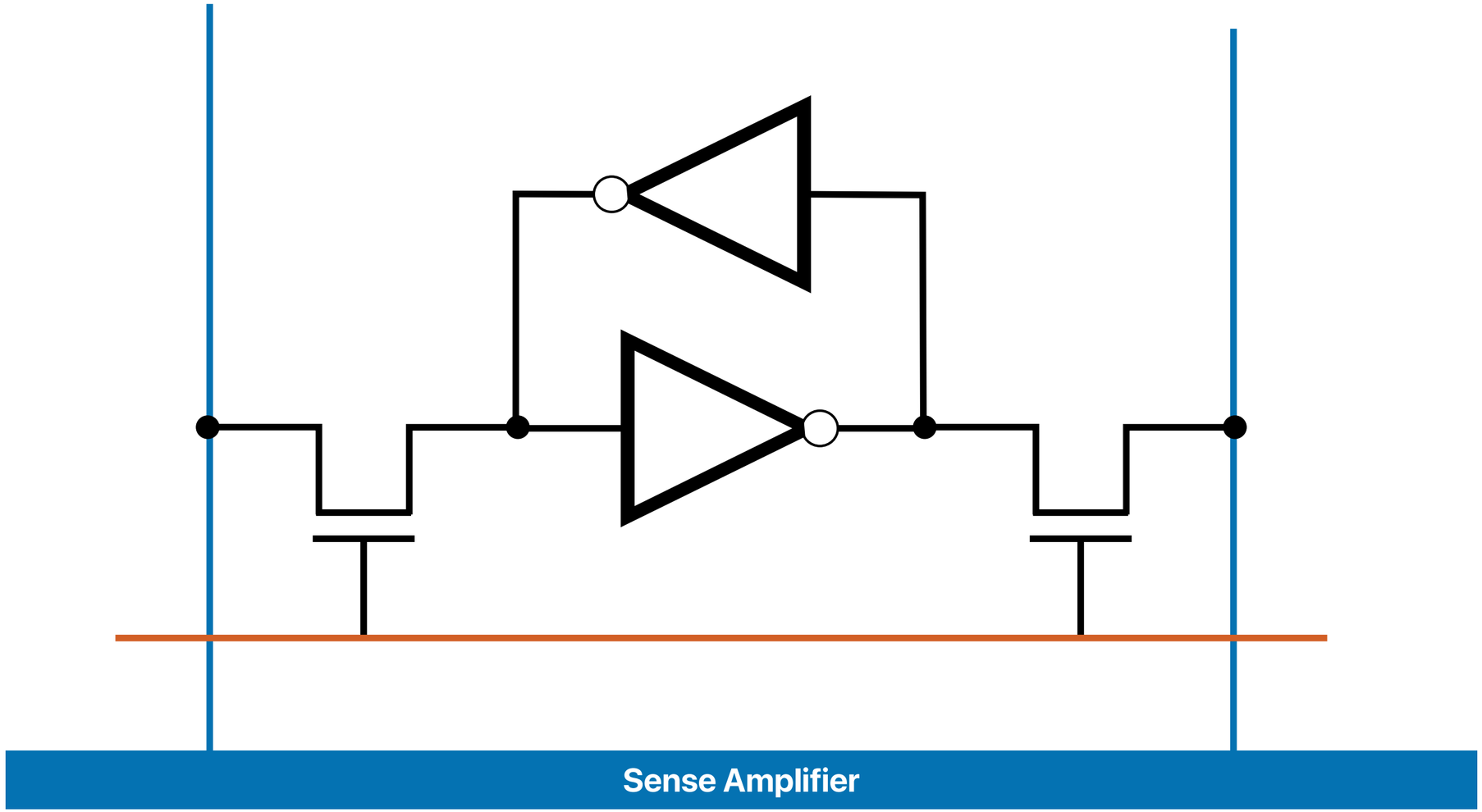
Memory technologies

Static Random Access Memory (SRAM)

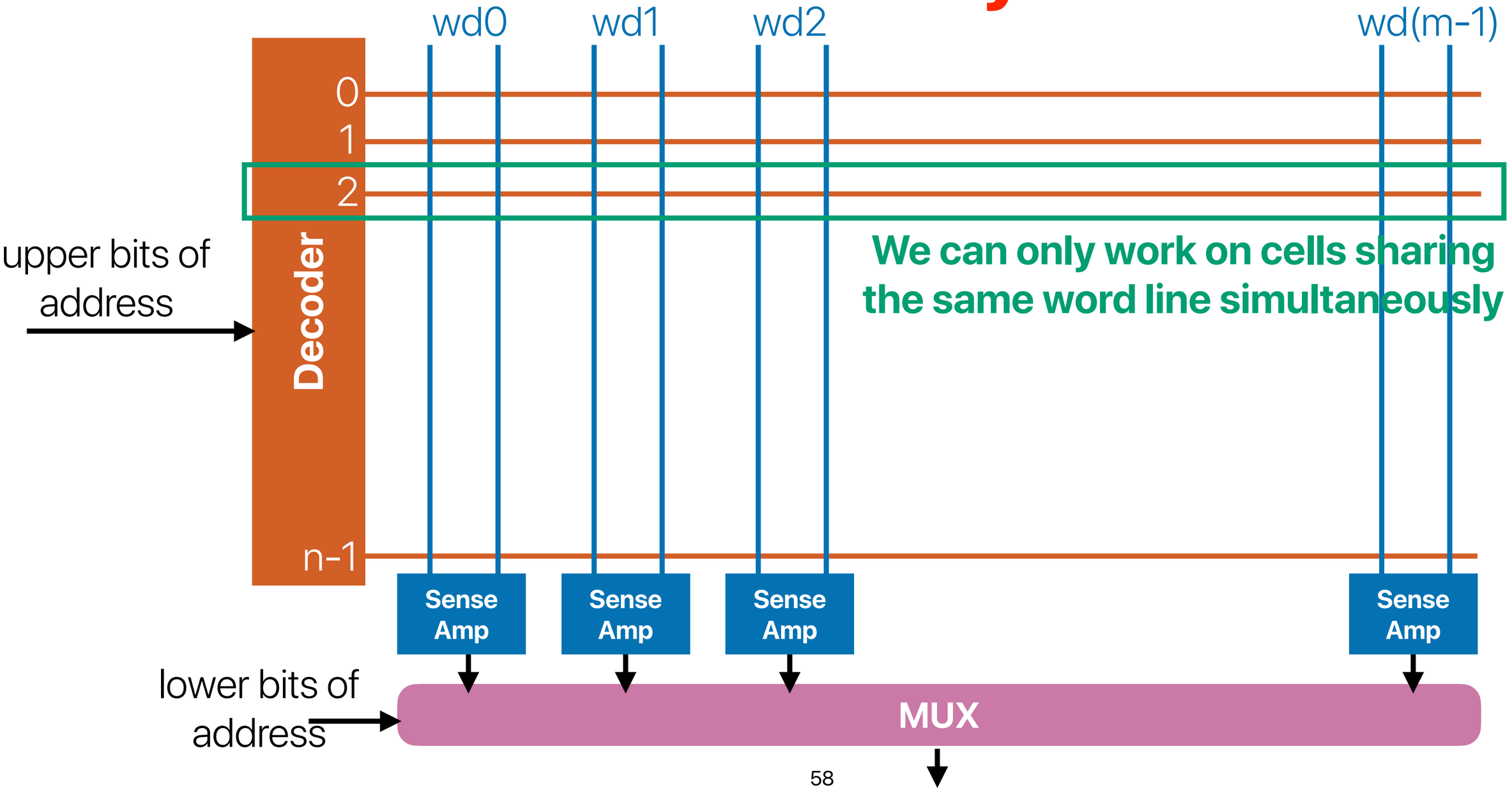
A Classical 6-T SRAM Cell



A Classical 6-T SRAM Cell

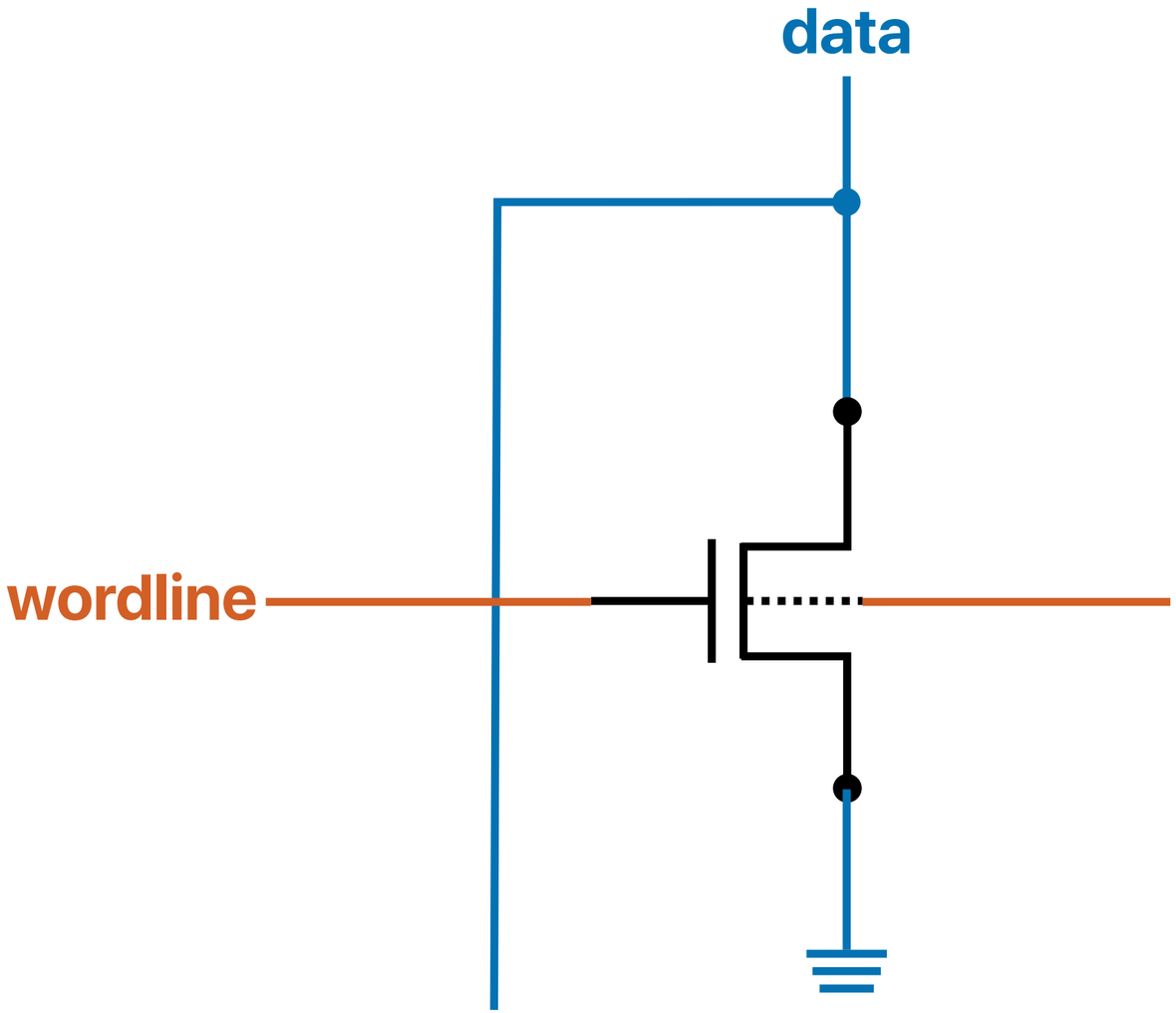


SRAM array


wd0
wd1
wd2
wd(m-1)
0
1
2
n-1
Decoder
upper bits of address
We can only work on cells sharing the same word line simultaneously
Sense Amp
Sense Amp
Sense Amp
Sense Amp
lower bits of address
MUX

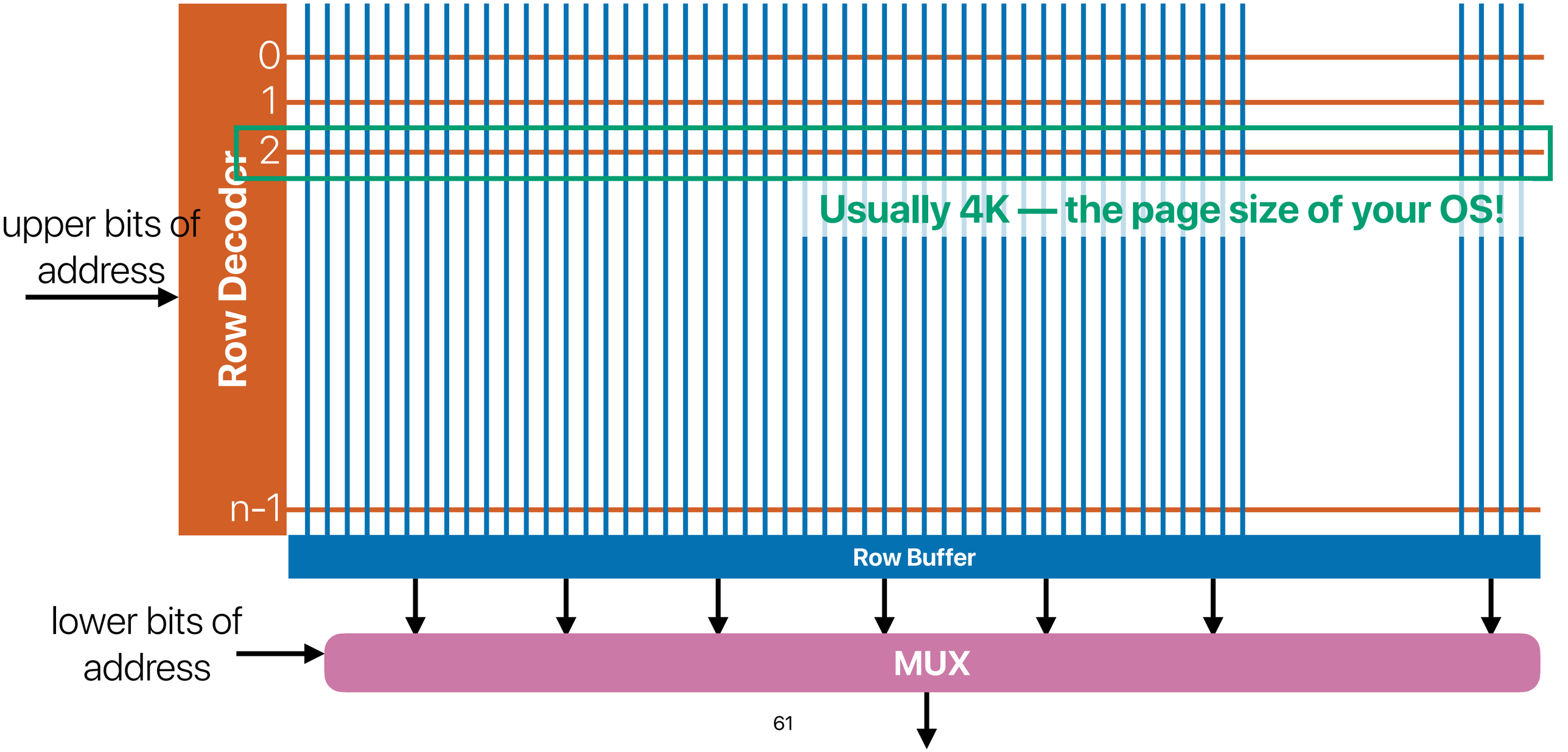
Dynamic Random Access Memory (DRAM)

An DRAM cell



- 1 transistor (rather than 6)
- Relies on large capacitor to store bit
  - Write: transistor conducts, data voltage level gets stored on top plate of capacitor
  - Read: look at the value of d
- Problem: Capacitor discharges over time
  - Must “refresh” regularly, by reading d and then writing it right back

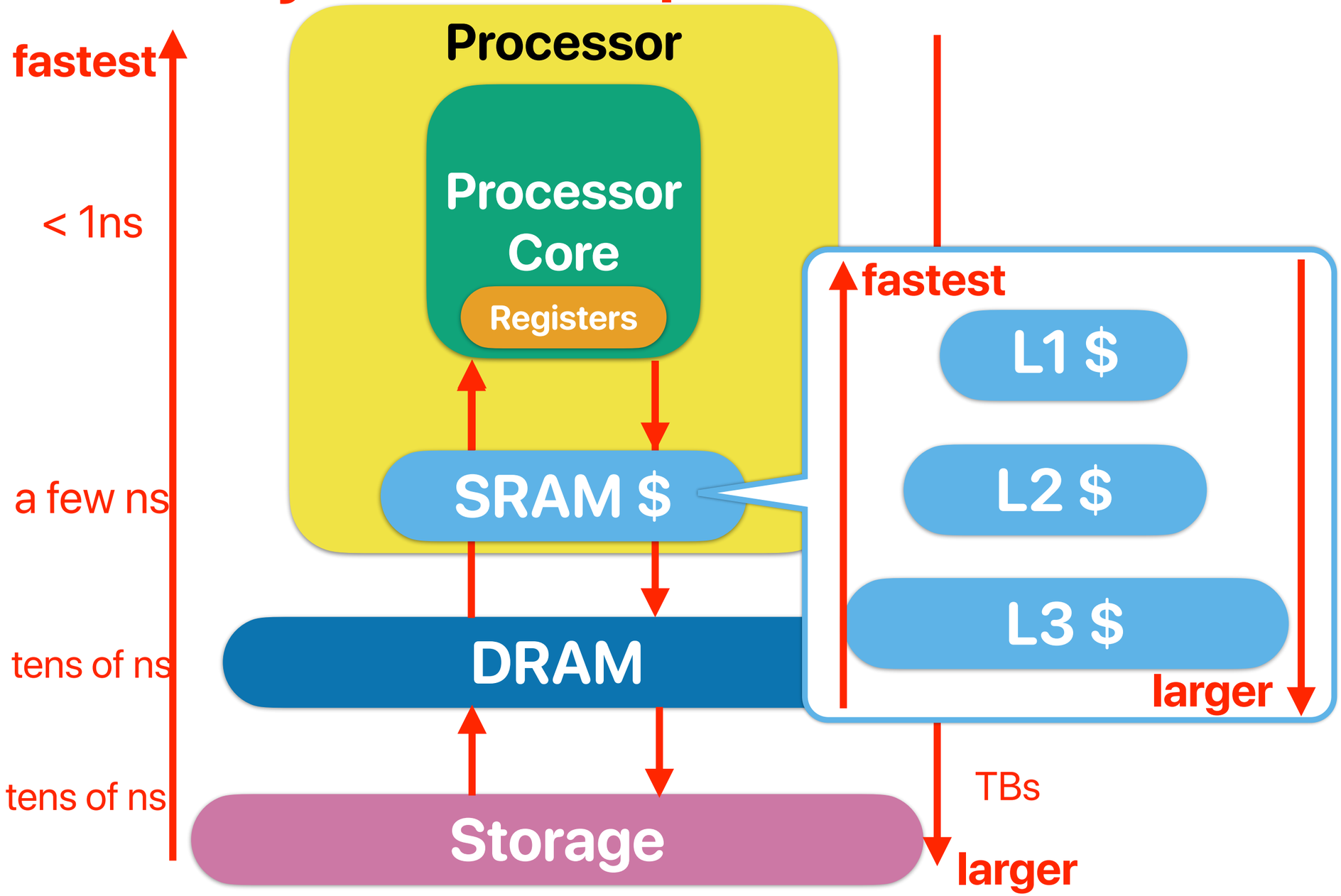
DRAM array


0
1
2
n-1
Row Decoder
upper bits of address
Usually 4K — the page size of your OS!
Row Buffer
lower bits of address
MUX

Recap: Latency of volatile memory

	Size (Transistors per bit)	Latency (ns)
Register	18T	~ 0.1 ns
SRAM	6T	~ 0.5 ns
DRAM	1T	50-100 ns

Memory "hierarchy" in modern processor architectures



Thinking about programming

```
struct student_record
{
    int id;
    double homework;
    double midterm;
    double final;
};

int main(int argc, char **argv)
{
    int i,j;
    double midterm_average=0.0;
    int number_of_records = 10000000;
    struct timeval time_start, time_end;
    struct student_record *records;
    records = (struct
student_record*)malloc(sizeof(struct
student_record)*number_of_records);
    init(number_of_records,records);


    for (j = 0; j < 100; j++)
        for (i = 0; i < number_of_records; i++)
            midterm_average+=records[i].midterm;

    printf("average: %lf\n",midterm_average/
number_of_records);
   free(records);
    return 0;
}
```

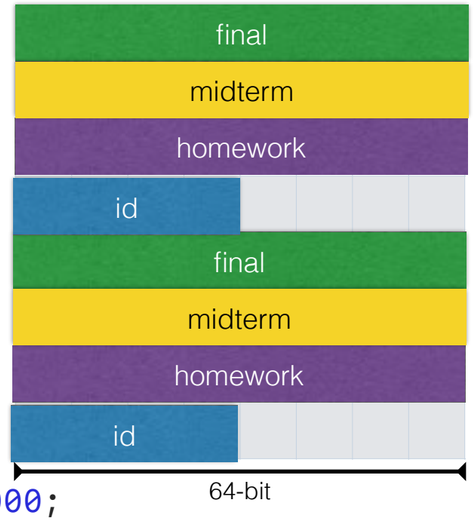
```
int main(int argc, char **argv)
{
    int i,j;
    double midterm_average=0.0;
    int number_of_records = 10000000;
    struct timeval time_start, time_end;
    id = (int*)malloc(sizeof(int)*number_of_records);
    midterm = (double*)malloc(sizeof(double)*number_of_records);
    final = (double*)malloc(sizeof(double)*number_of_records);
    homework = (double*)malloc(sizeof(double)*number_of_records);
    init(number_of_records);

    for (j = 0; j < 100; j++)
        for (i = 0; i < number_of_records; i++)
            midterm_average+=midterm[i];

    free(id);
    free(midterm);
    free(final);
    free(homework);
    return 0;
}
```

More row buffer hits in the DRAM, more SRAM hits

- Which side is faster in executing the for-loop?
  - A. Left
  - B. Right
  - C. About the same

Thinking about programming (2)

```c
struct student_record
{
    int id;
    double homework;
    double midterm;
    double final;
};

int main(int argc, char **argv)
{
    int i,j;
    double midterm_average=0.0;
    int number_of_records = 10000000;
    struct timeval time_start, time_end;
    struct student_record *records;
    records = (struct
student_record*)malloc(sizeof(struct
student_record)*number_of_records);
    init(number_of_records,records);


    for (j = 0; j < 100; j++)
        for (i = 0; i < number_of_records; i++)
            midterm_average+=records[i].midterm;

    printf("average: %lf\n",midterm_average/
number_of_records);
   free(records);
    return 0;
}
```



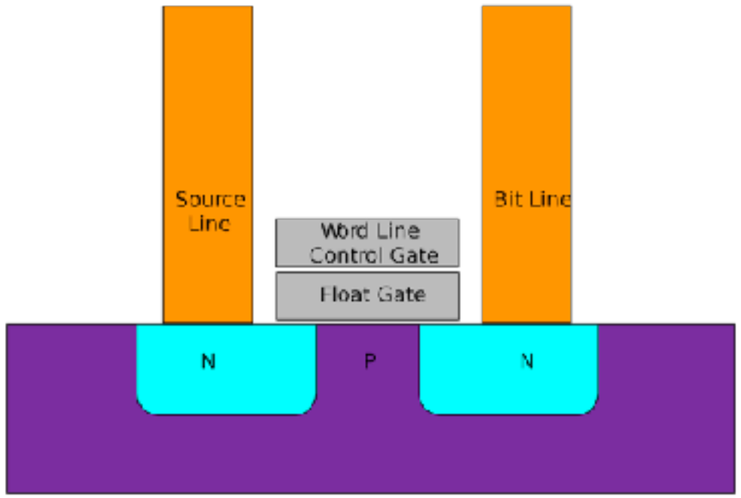

```c
int main(int argc, char **argv)
{
    int i,j;
    double midterm_average=0.0;
    int number_of_records = 10000000;
    struct timeval time_start, time_end;
    id = (int*)malloc(sizeof(int)*number_of_records);
    midterm = (double*)malloc(sizeof(double)*number_of_records);
    final = (double*)malloc(sizeof(double)*number_of_records);
    homework = (double*)malloc(sizeof(double)*number_of_records);
    init(number_of_records);

    for (j = 0; j < 100; j++)
        for (i = 0; i < number_of_records; i++)
            midterm_average+=midterm[i];

    free(id);
    free(midterm);
    free(final);
    free(homework);
    return 0;
}
```

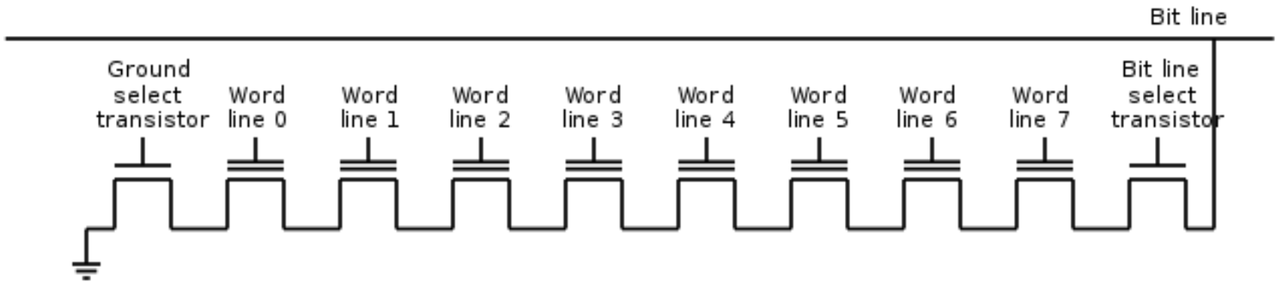
- Which side is consuming less memory?
  A. Left
  B. Right
  C. About the same

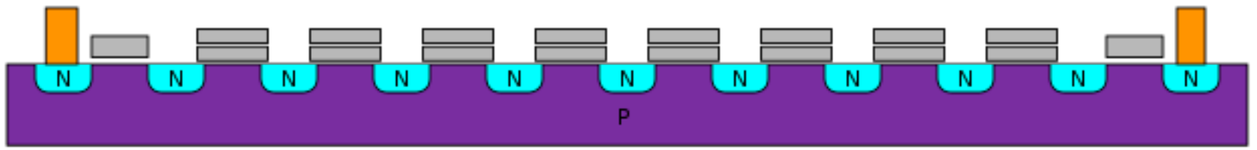
Flash memory

Recap: Flash memory

- Floating gate made by polycrystalline silicon trap electrons
- The voltage level within the floating gate determines the value of the cell
- The floating gates will wear out eventually





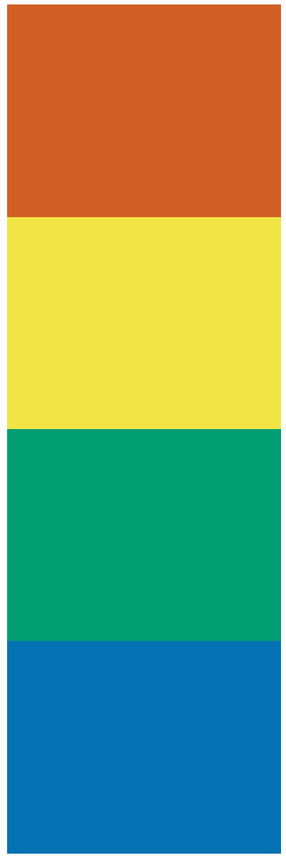


Recap: Types of Flash Chips

2 voltage levels, 1-bit

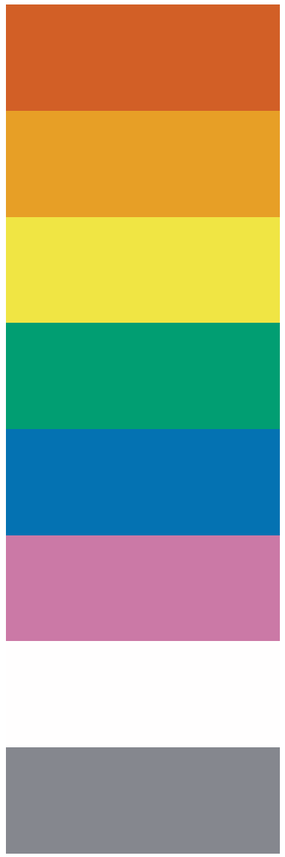
Single-Level Cell (SLC)

4 voltage levels, 2-bit



Multi-Level Cell (MLC)

8 voltage levels, 3-bit



Triple-Level Cell (TLC)

16 voltage levels, 4-bit

Quad-Level Cell (QLC)

Programming in MLC

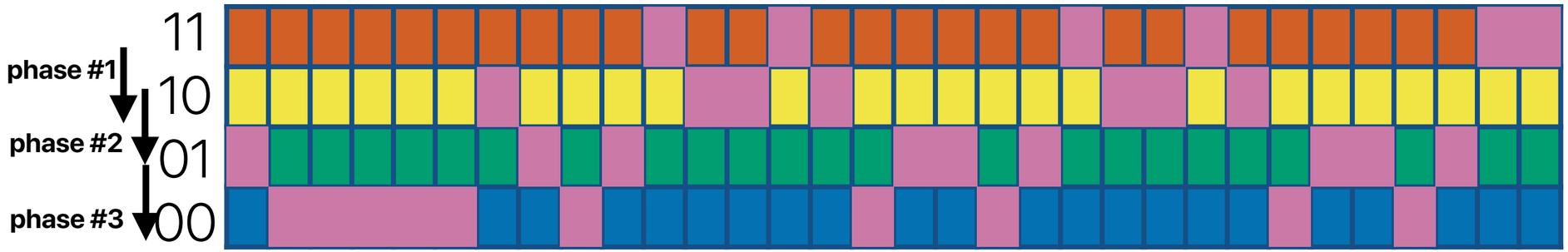
4 voltage levels,
2-bit

Multi-Level Cell (MLC)

3.14000000000000012434497875800

= 0x40091EB851EB851F

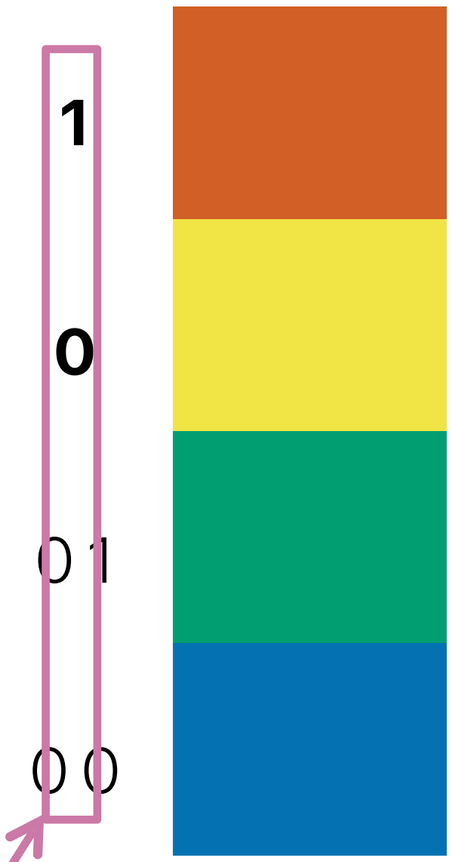
= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



3 Cycles/Phases to finish programming

Optimizing 1st Page Programming in MLC

4 voltage levels,
2-bit



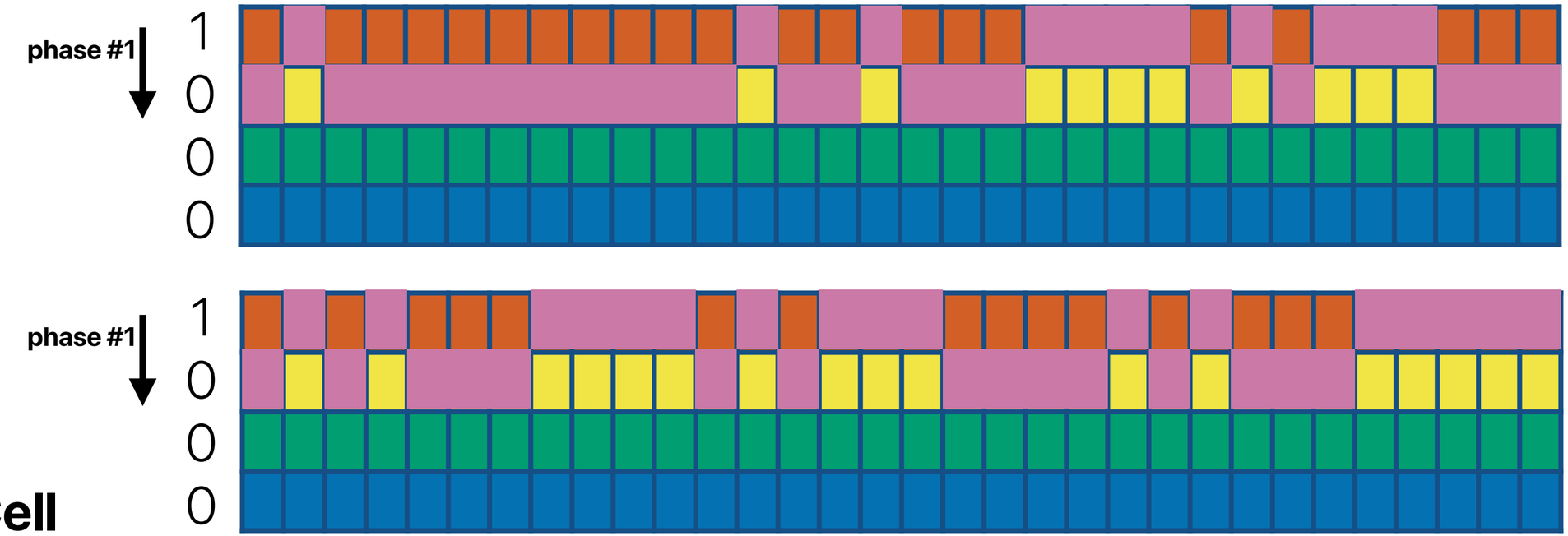
1st page

Multi-Level Cell (MLC)

3.14000000000000012434497875800

= 0x40091EB851EB851F

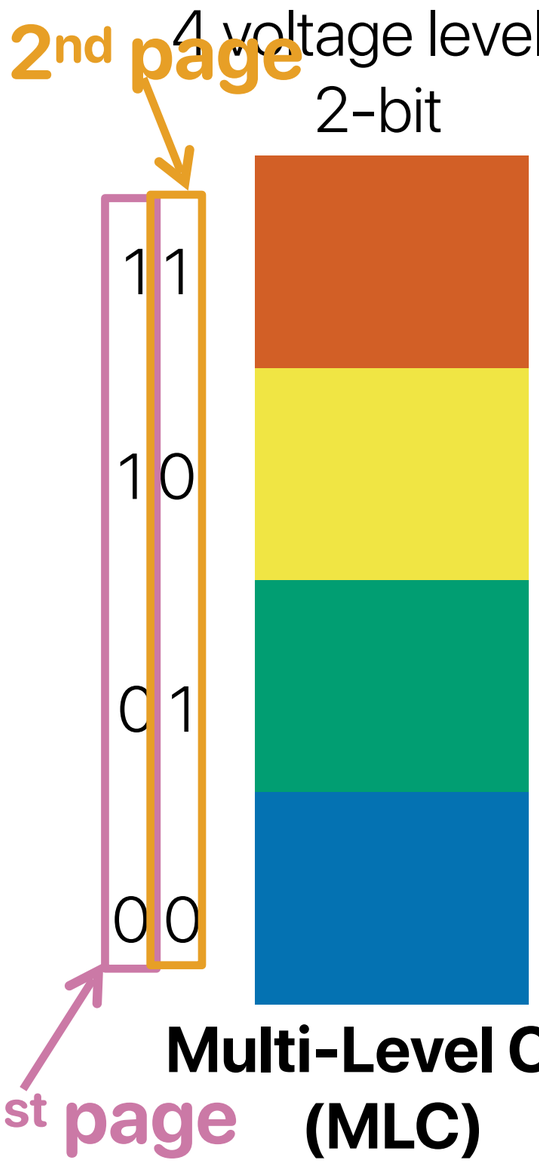
= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



1 Phase to finish programming the first page!

— the phase is shorter now

2nd Page Programming in MLC

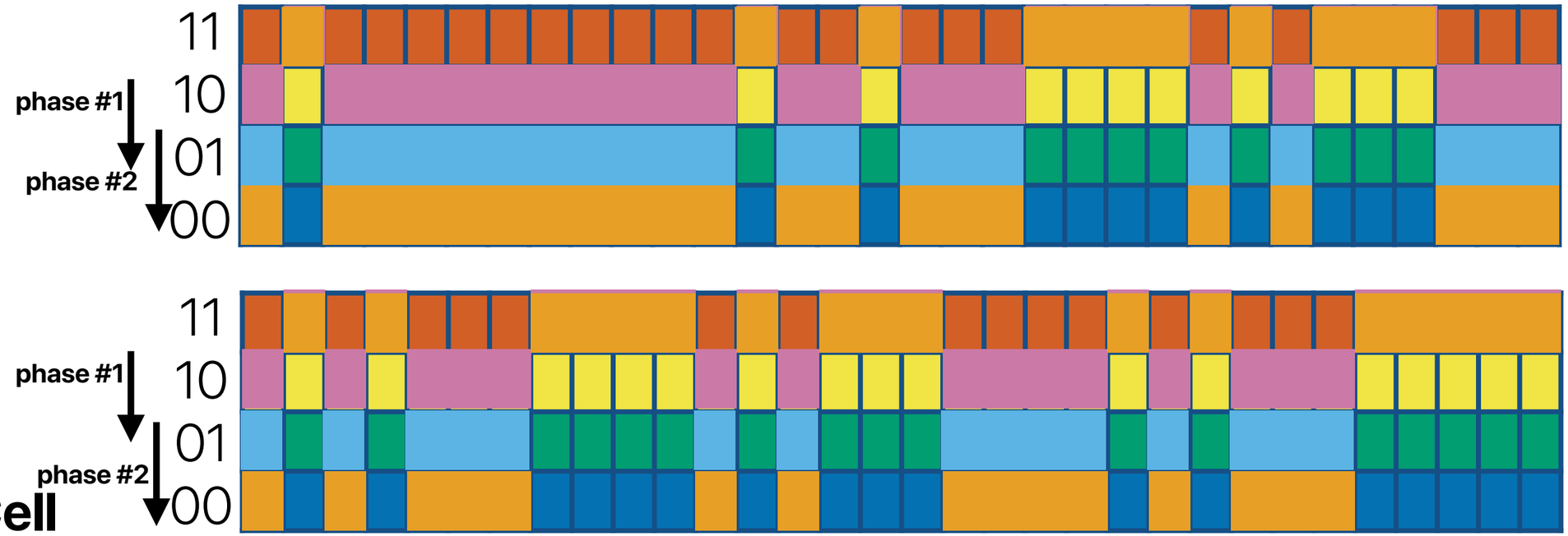


3.140000000000001243449787580

= 0x40091EB851EB851F

= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111

= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



2 Phase to finish programming the second page!

QLC = More Density Per NAND Cell

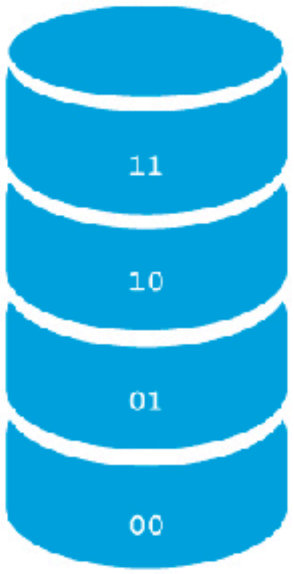
Lower $ per GB

SLC

1 Bit Per Cell

First SSD NAND technology

100K P/E Cycles
(at technology introduction)

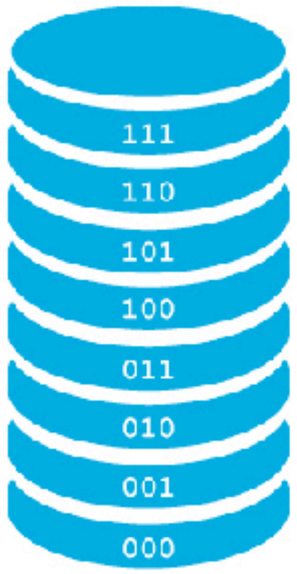


MLC

2 Bits Per Cell

100% increase

10K P/E Cycles

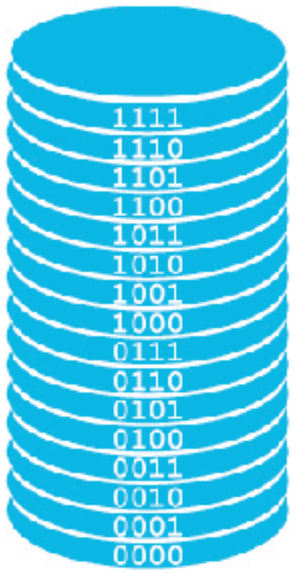


TLC

3 Bits Per Cell

50% increase

3K P/E Cycles



QLC

4 Bits Per Cell

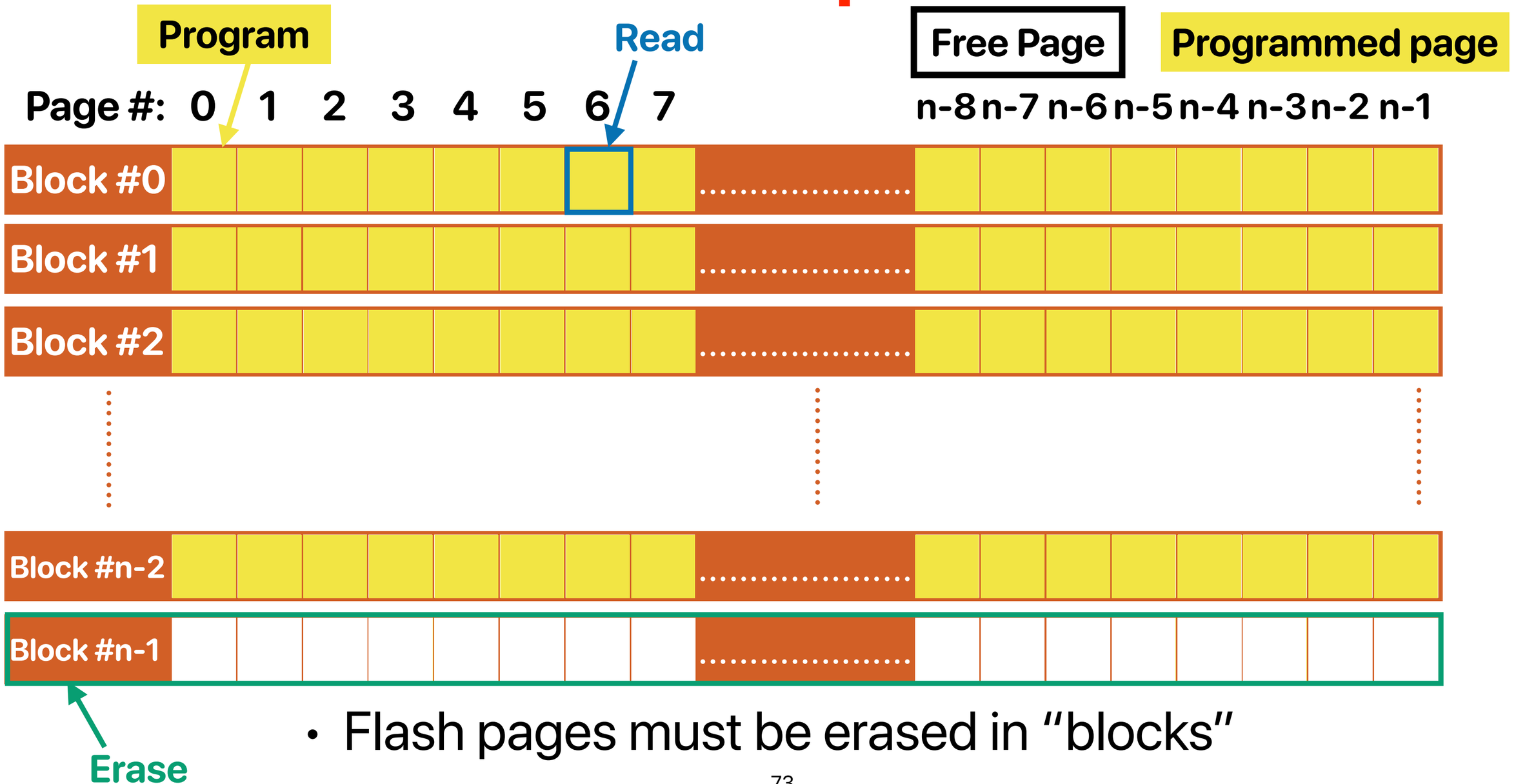
33% increase

1K P/E Cycles

Fewer writes per cell

Micron

Basic flash operations



- Flash pages must be erased in “blocks”

If programmer doesn't know flash "features"

- Software designer should be aware of the characteristics of underlying hardware components

Spotify is writing massive amounts of junk data to storage drives

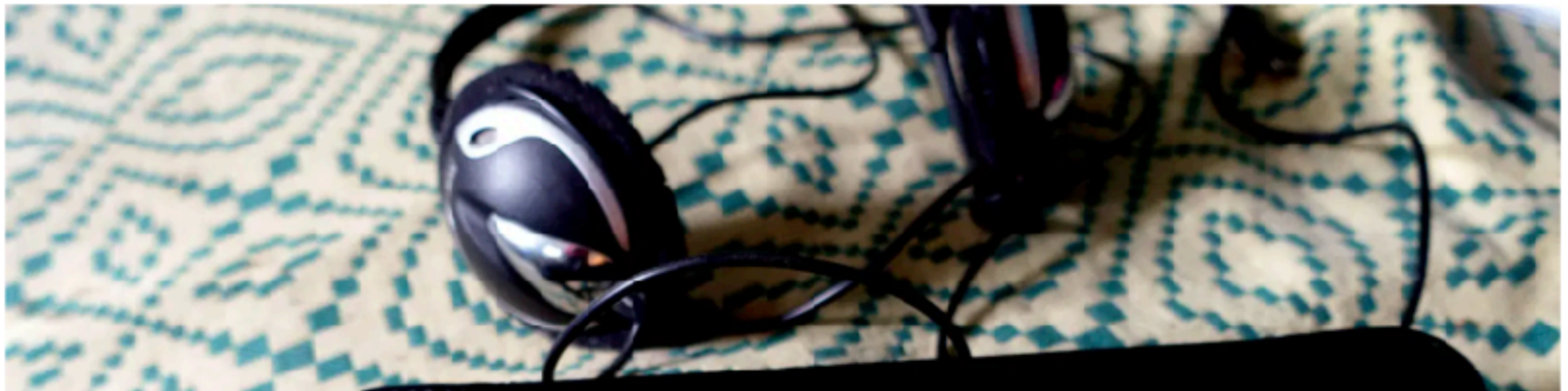
Streaming app used by 40 million writes hundreds of gigabytes per day.

DAN GOODIN - 11/10/2016, 7:00 PM

BGR TECH ENTERTAINMENT DEALS BUSINESS SCIENCE LIFESTYLE

TECH

Spotify has been quietly killing your SSD's life for months



#T

1 Doctors j promisin—the Spotify music streaming app has been assaulting to potentially take years off their expected lifespans. eds of gigabytes being written in an hour aren't ed amounts are measured in terabytes. The overload

2 The mos

sn't storing any songs locally.

Power consumption

Power v.s. Energy

- Power is the direct contributor of “heat”
  - Packaging of the chip
  - Heat dissipation cost
  - Power = $P_{Dynamic} + P_{static}$
- Energy = P * ET
  - The electricity bill and battery life is related to energy!
  - Lower power does not necessary means better battery life if the processor slow down the application too much

Dynamic/Active Power

- The power consumption due to the switching of transistor states
- Dynamic power per transistor

  $P_{dynamic} \sim \alpha \times C \times V^2 \times f \times N$

  - α: average switches per cycle
  - C: capacitance
  - V: voltage
  - f: frequency, usually linear with V
  - N: the number of transistors

Static/Leakage Power

- The power consumption due to leakage — transistors do not turn all the way off during no operation
- Becomes the **dominant** factor in the most advanced process technologies.

$$P_{leakage} \sim N \times V \times e^{-V_t}$$

  - N: number of transistors
  - V: voltage
  - V_t: threshold voltage where transistor conducts (begins to switch)

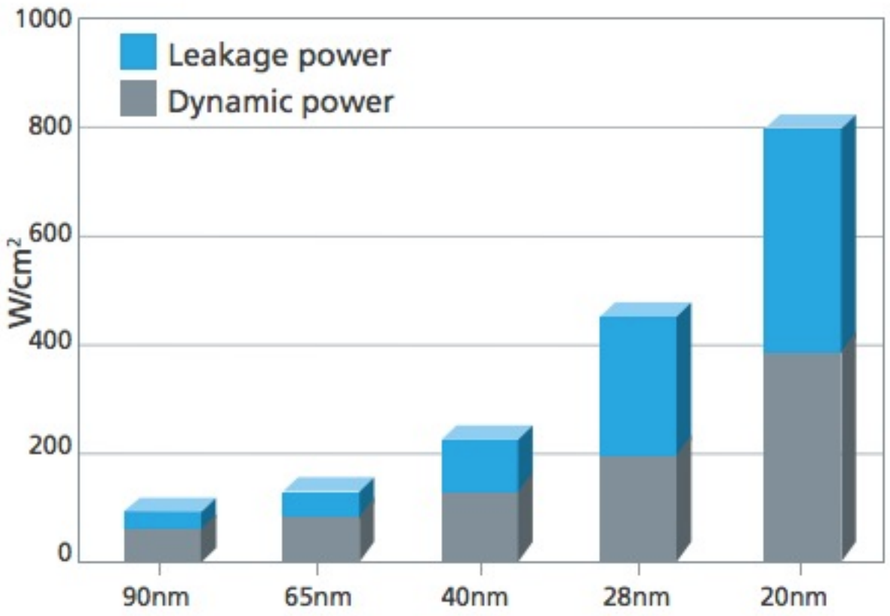


Figure 1: Leakage power becomes a growing problem as demands for more performance and functionality drive chipmakers to nanometer-scale process nodes (Source: IBS).

Dennardian Broken

- Given a scaling factor S

Parameter	Relation	Classical Scaling	Leakage Limited
Power Budget		1	1
Chip Size		1	1
Vdd (Supply Voltage)		1/S	1
Vt (Threshold Voltage)	1/S	1/S	1
tex (oxide thickness)		1/S	1/S
W, L (transistor dimensions)		1/S	1/S
Cgate (gate capacitance)	WL/tox	1/S	1/S
Isat (saturation current)	WVdd/tox	1/S	1
F (device frequency)	Isat/(CgateVdd)	S	S
D (Device/Area)	1/(WL)	S^2	S^2
p (device power)	IsatVdd	$1/S^2$	1
P (chip power)	Dp	1	S^2
U (utilization)	1/P	1	$1/S^2$

Power consumption

Chip

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

=49W

Dennardian Scaling

Chip

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

=50W

Dennardian Broken

Chip

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

=100W!

Power density

Chip

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

$$= \frac{49W}{Chip\ Area}$$

Dennardian Scaling

Chip

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

$$= \frac{50W}{Chip\ Area}$$

Dennardian Broken

Chip

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

$$= \frac{100W}{Chip\ Area}$$

Power density

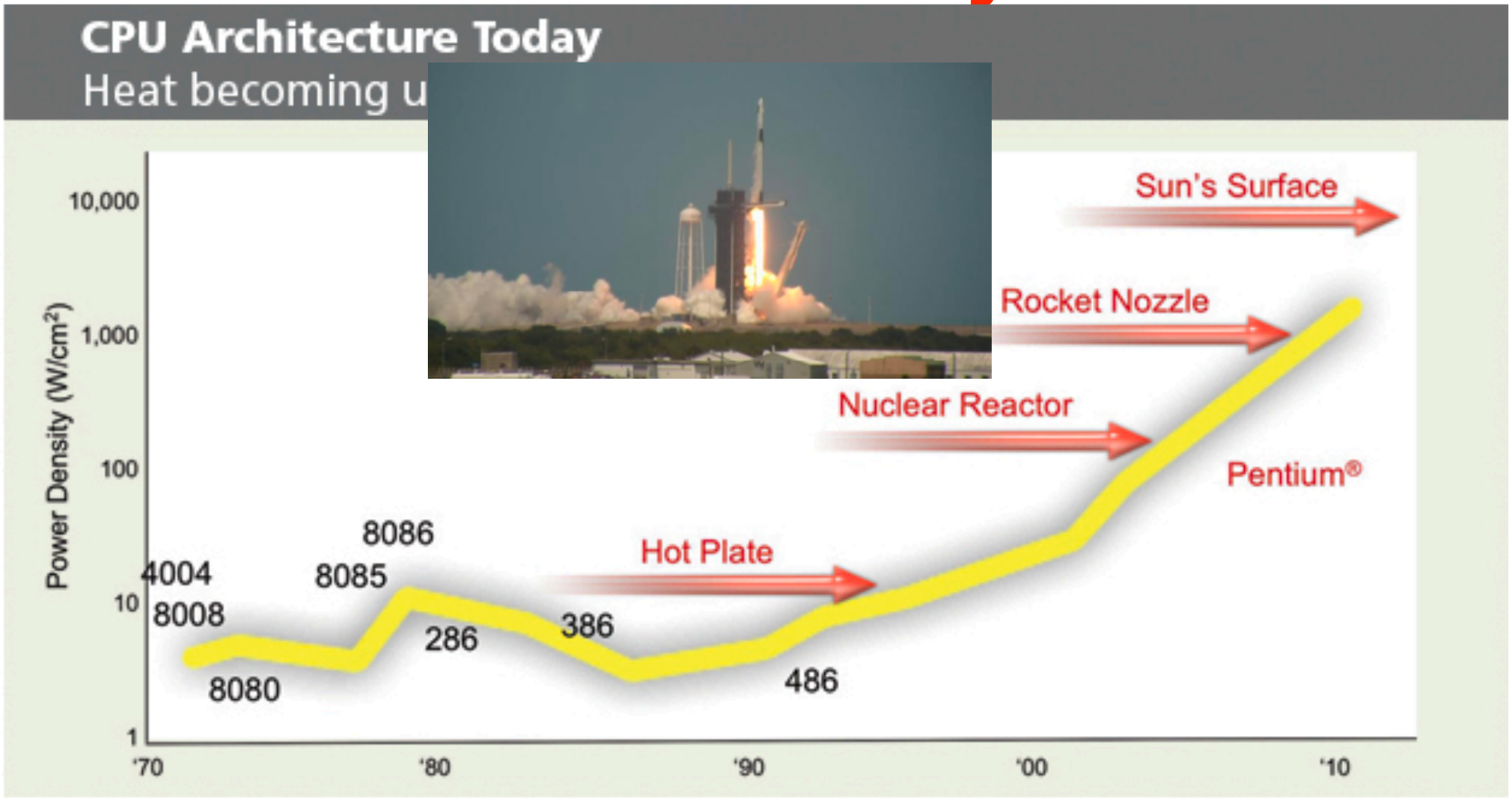


Figure 1. In CPU architecture today, heat is becoming an unmanageable problem. (Courtesy of Pat Gelsinger, Intel Developer Forum, Spring 2004)

https://www.cadalyst.com/hardware/workstation-performance-tomorrow039s-possibilities-viewpoint-column-6351

Power consumption to light on all transistors

If we can only cool down 50W in the same area —

Chip

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

=49W

Dennardian Scaling

Chip

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

=50W

Dennardian Broken

Chip

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

On ~ 50W

Off ~ 0W

Dark!

=100W!

Dark silicon

- Your power consumption goes up as the number of transistors goes up
- Even Moore's Law allows us to put more transistors within the same area —we cannot use them all simultaneously!
- We have no choice to not activate all transistors at the same time!

Trends in the Dark Silicon Era

- Aggressive dynamic voltage/frequency scaling
- Throughout oriented — slower, but more
- Just let it dark — activate part of circuits, but not all
- From general-purpose to domain-specific — ASIC

More cores per chip, slower per core

Products Solutions Support

intel

	Intel® Xeon® Processor E7-8890 v4	Intel® Xeon® Processor E7-8893 v4	Intel® Xeon® Processor E7-8880 v4
Status	Launched	Launched	Launched
Launch Date	Q2'16	Q2'16	Q2'16
Lithography	14 nm	14 nm	14 nm
Performance			
# of Cores	24	4	22
# of Threads	48	8	44
Processor Base Frequency	2.20 GHz	3.20 GHz	2.20 GHz
Max Turbo Frequency	3.40 GHz	3.50 GHz	3.30 GHz
Cache	60 MB	60 MB	55 MB
Bus Speed	9.6 GT/s	9.6 GT/s	9.6 GT/s
# of QPI Links	3	3	3
TDP	165 W	140 W	150 W

The rise of GPU

Real-time AI: Microsoft announces preview of Project Brainwave

AI & MACHINE LEARNING

An in-depth look at Google's first Tensor Processing Unit (TPU)

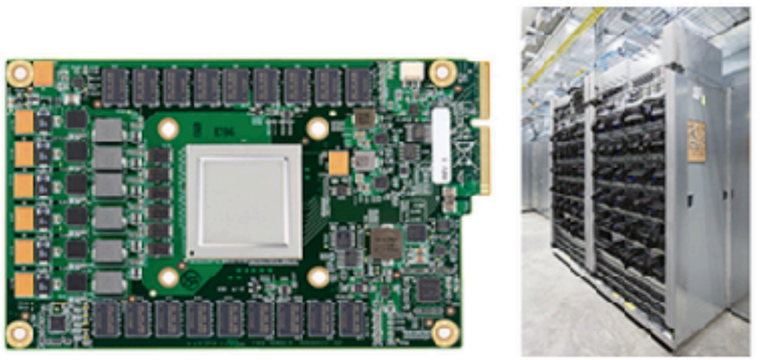
Kaz Sato
Staff Developer Advocate, Google Cloud

Cliff Young
Software Engineer, Google Brain

David Patterson
Distinguished Engineer, Google Brain

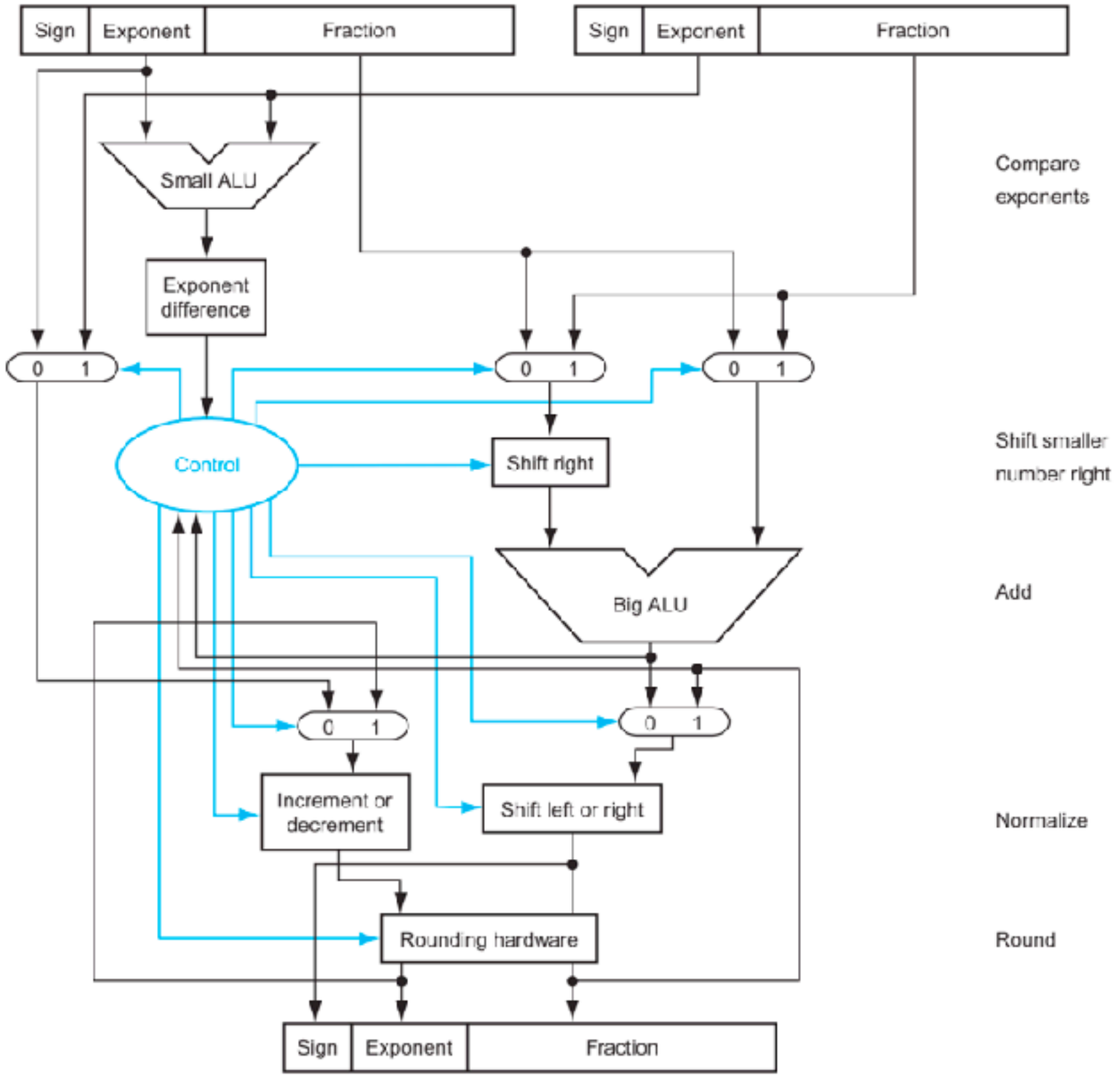
May 12, 2017

There's a common thread that connects Google services such as Google Search, Street View, Google Photos and Google Translate: they all use Google's Tensor Processing Unit, or TPU, to accelerate their neural network computations behind the scenes.



Google's first Tensor Processing Unit (TPU) on a printed circuit board (left); TPUs deployed in a Google datacenter (right)

Recap: Floating point adder



Programming and hardware

```c
#include <stdio.h>

int main(int argc, char **argv)
{
    float a, b, c, d;
    int i = 0;
    a = 1.345;
    b = 1.123;
    c = a + b;
    printf("A: %d\n", c==2.468);

    a = 16777216.0;
    b = 1.0;
    c = a+b+b;
    d = a+2*b;
    printf("B: %d\n", c==d);

    printf("C: %d\n", d == 16777218.0);

    a = 1.0;
    for(i=0;i<200;i++)
        a *= 2.0;
    printf("D: %f\n", a);

    a = a - a;
    printf("E: %f\n", a);
    return 0;
}
```

Data storage matters

Poll close in 1:30

Expected file size

- What's the expected size of "test.a"
  A. 4
  B. 8
  C. 9
  D. 10
  E. None of the above

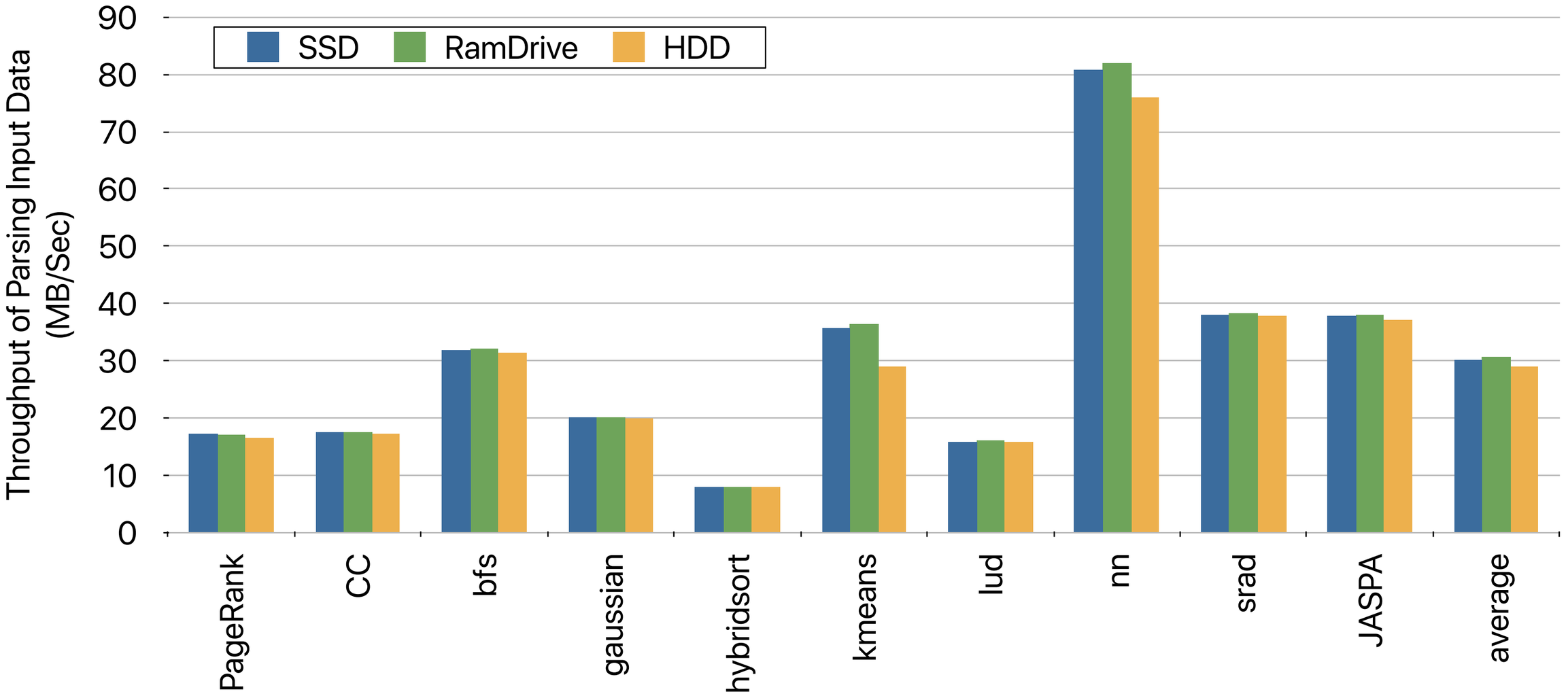
```c
#include <stdio.h>

int main(int argc, char **argv)
{
    FILE *fp;
    int a = 123456789;
    fp = fopen("test.a","w");
    fprintf(fp, "%d", a);
    fclose(fp);
    return 0;
}
```

Expected file size

- What's the expected size of "test.a"
  A. 4
  B. 8
  C. 9
  D. 10
  E. None of the above

```c
#include <stdio.h>

int main(int argc, char **argv)
{
    FILE *fp;
    int a = 123456789;
    fp = fopen("test.a","w");
    fprintf(fp, "%d", a);
    fclose(fp);
    return 0;
}
```

ASCII to Binary doesn't scale with I/O bandwidth



Demo

Let’s take a group photo!

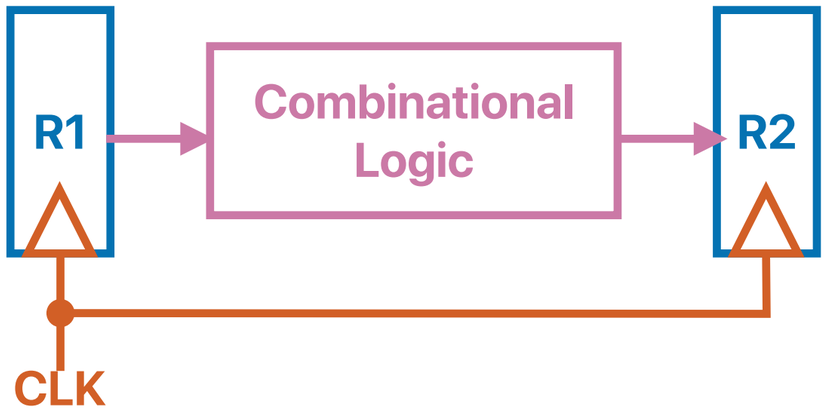
Sample Final

Format of final

- Multiple choices * 50
- (Multiple choices + explanation) * 10
- Free answer/open-ended questions * 6
-

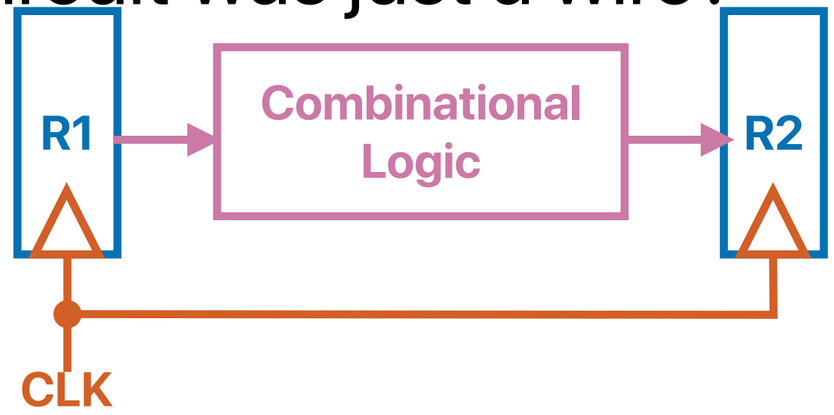
Causes of Timing Issues in Sequential Circuits

- Input to a FF comes from the output of another FF through a combinational circuit
- The FF and combinational circuit have a min & max delay
- Which of the following violations occurs if max delay of R1 is zero & max delay of the combinational circuit is equal to the clock period?
  A. Hold time violation for R2
  B. Setup violation for R2
  C. Hold time violation for R1
  D. Setup violation for R1
  E. None of the above



Causes of Timing Issues in Sequential Circuits (2)

- Input to a FF comes from the output of another FF through a combinational circuit
- The FF and combinational circuit have a min & max delay
- Which of the following violations occurs if min delay of R1 is zero & max delay of the combinational circuit was just a wire?
  A. Hold time violation for R2
  B. Setup violation for R2
  C. Hold time violation for R1
  D. Setup violation for R1
  E. None of the above

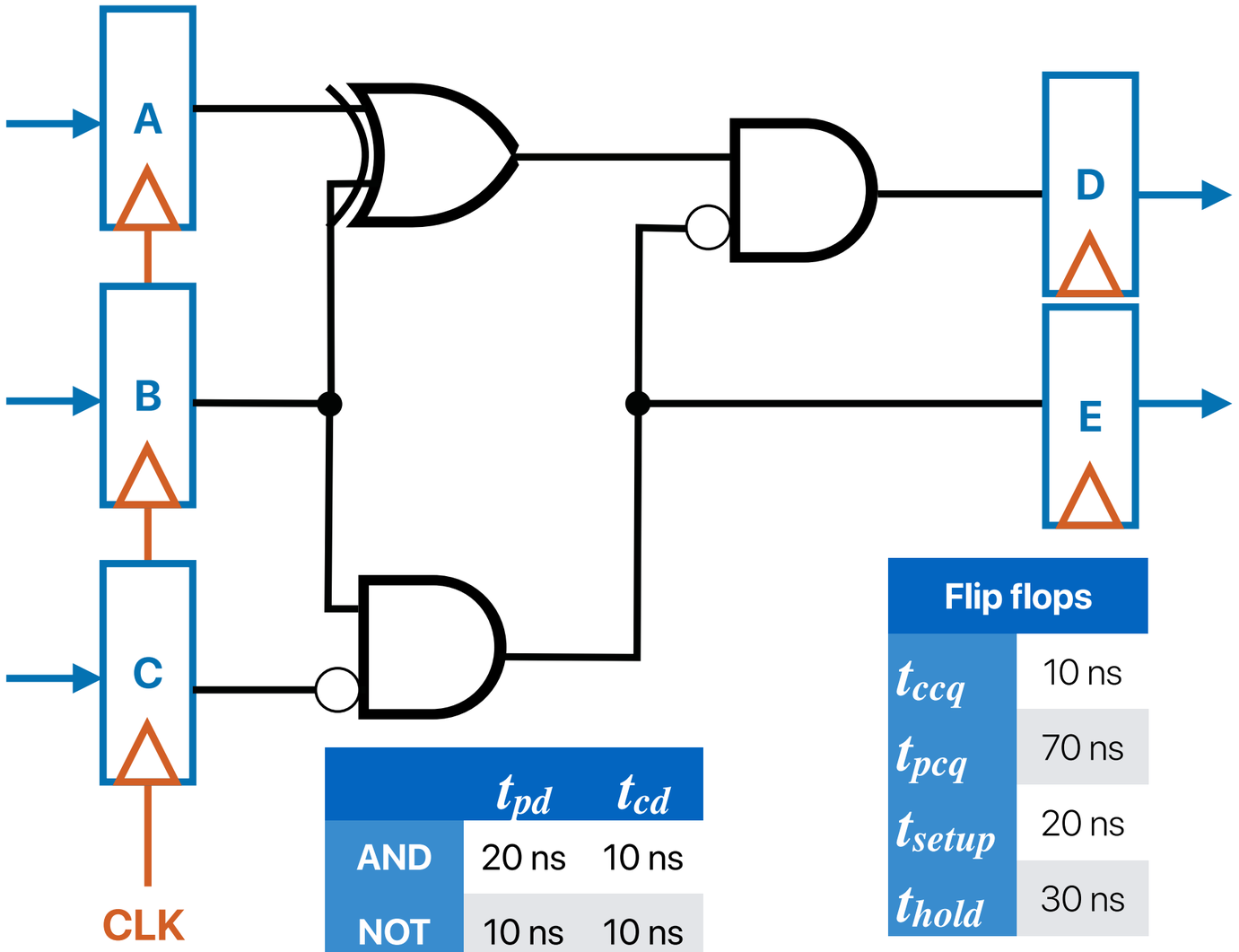


Example: timing constraints

- What's the maximum frequency?
  - A. 1 / 110ns
  - B. 1 / 220ns
  - C. 1 / 200ns
  - D. 1 / 180ns
  - E. None of the above

$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$

$$t_{ccq} + t_{cd} > t_{hold}$$



	t_{pd}	t_{cd}
AND	20 ns	10 ns
NOT	10 ns	10 ns
XOR	110 ns	50 ns

Flip flops	
t_{ccq}	10 ns
t_{pcq}	70 ns
t_{setup}	20 ns
t_{hold}	30 ns

Power & Energy

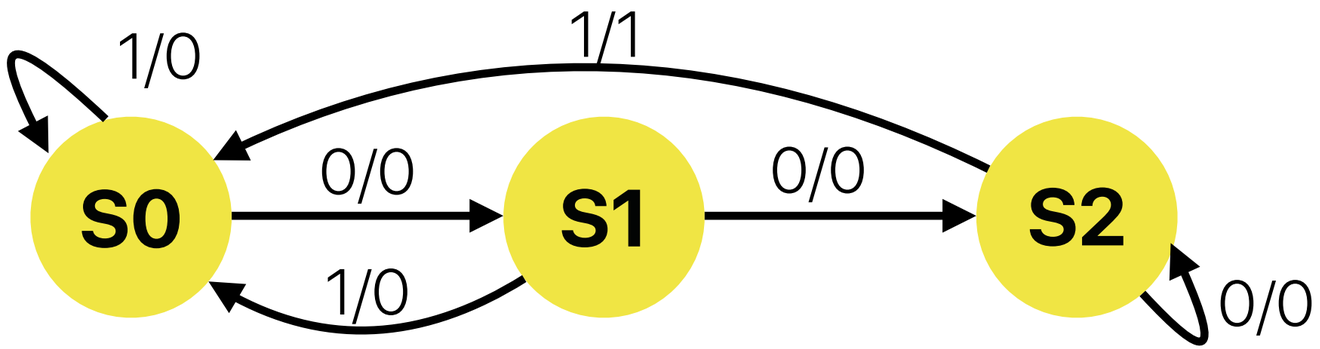
- Regarding power and energy, how many of the following statements are correct?
  - ① Lowering the power consumption helps extending the battery life
  - ② Lowering the power consumption helps reducing the heat generation
  - ③ Lowering the energy consumption helps reducing the electricity bill
  - ④ A CPU with 10% utilization can still consume 33% of the peak power
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

What happens if power doesn’t scale with process technologies?

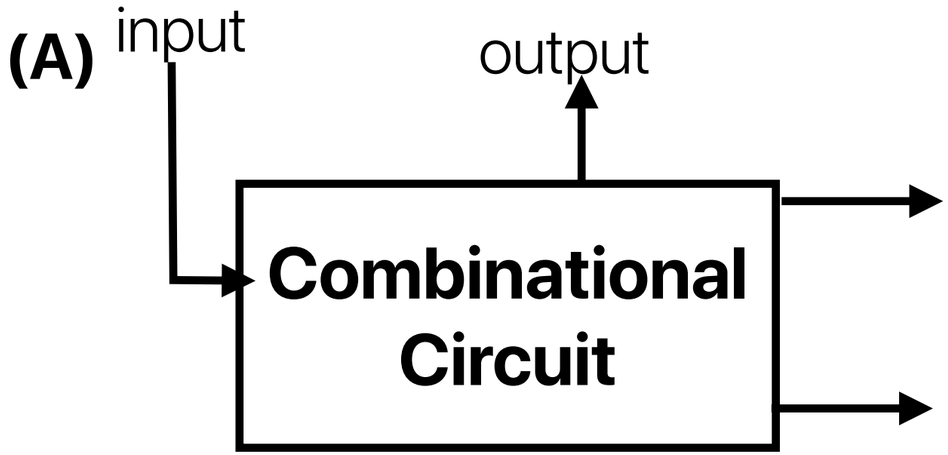
- If we are able to cram more transistors within the same chip area (Moore’s law continues), but the power consumption per transistor remains the same. Right now, if put more transistors in the same area because the technology allows us to. How many of the following statements are true?
  - ① The power consumption per chip will increase
  - ② The power density of the chip will increase
  - ③ Given the same power budget, we may not able to power on all chip area if we maintain the same clock rate
  - ④ Given the same power budget, we may have to lower the clock rate of circuits to power on all chip area

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

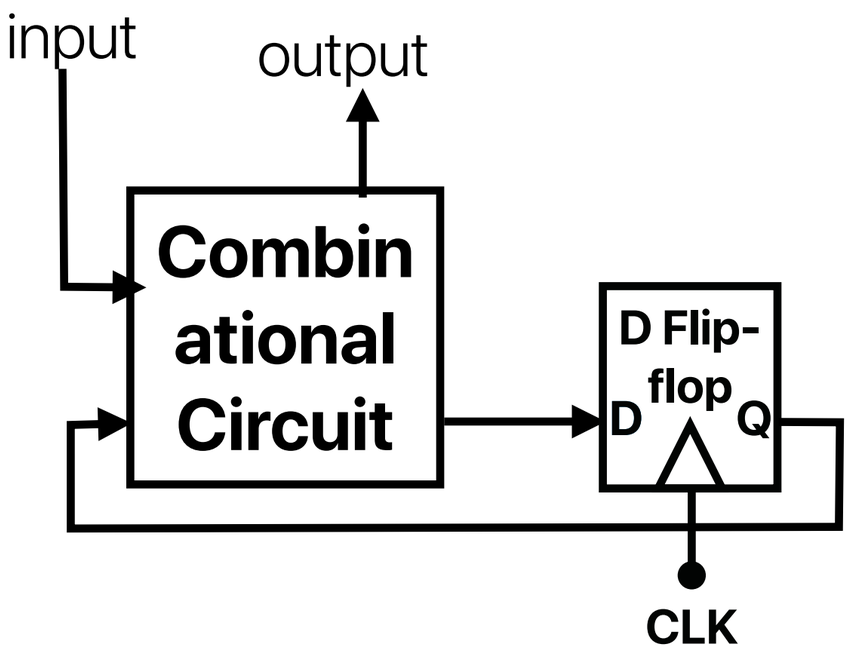
Which is the most likely circuit realization of Life on Mars



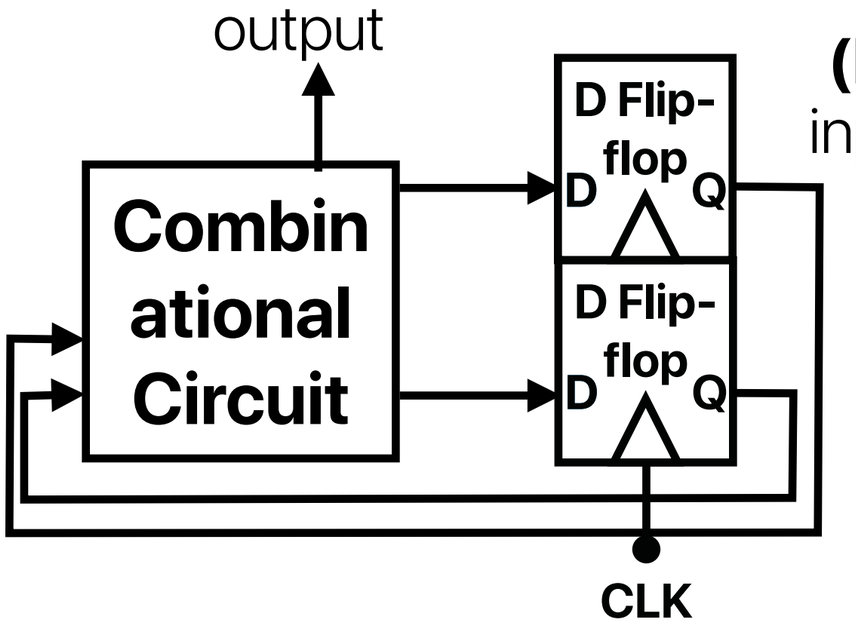
(A)



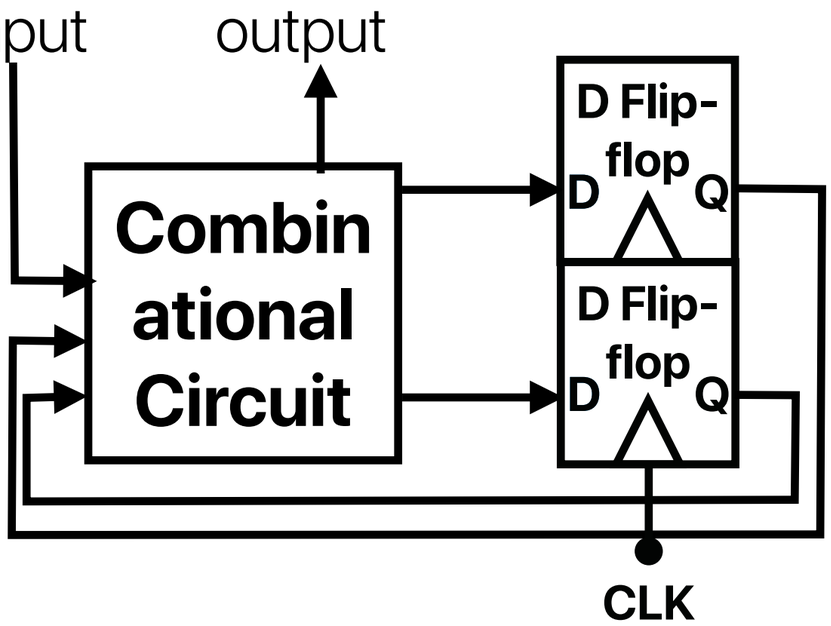
(B)



(C)



(D)



Flash memory characteristics

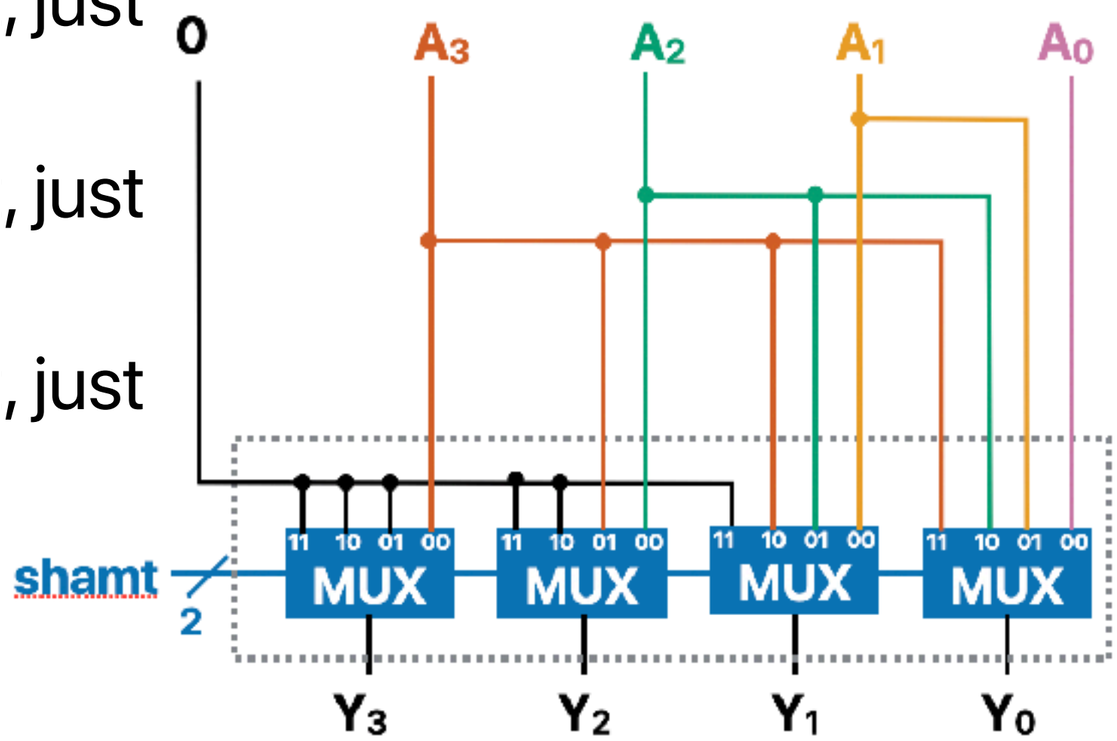
- Regarding the following flash memory characteristics, please identify how many of the following statements are correct
  - ① Flash memory cells can only be programmed with limited times
  - ② The reading latency of flash memory cells can be largely different from programming
  - ③ The latency of programming different flash memory pages can be different
  - ④ The programmed cell cannot be reprogrammed again unless its charge level is refilled to the top-level
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

Frequency

- Consider the following adders. Assume each gate delay is 1ns and the delay in a register is 2ns. Please rank their maximum operating frequencies
  - ① 32-bit CLA made with 8 4-bit CLA adders
  - ② 32-bit CRA made with 32 full adders
  - ③ 32-bit serial adders made with 4-bit CLA adders
  - ④ 32-bit serial adders made with 1-bit full adders
  - A. (1) > (2) > (3) > (4)
  - B. (2) > (1) > (4) > (3)
  - C. (2) > (1) > (3) > (4)
  - D. (4) > (3) > (2) > (1)
  - E. (4) > (3) > (1) > (2)

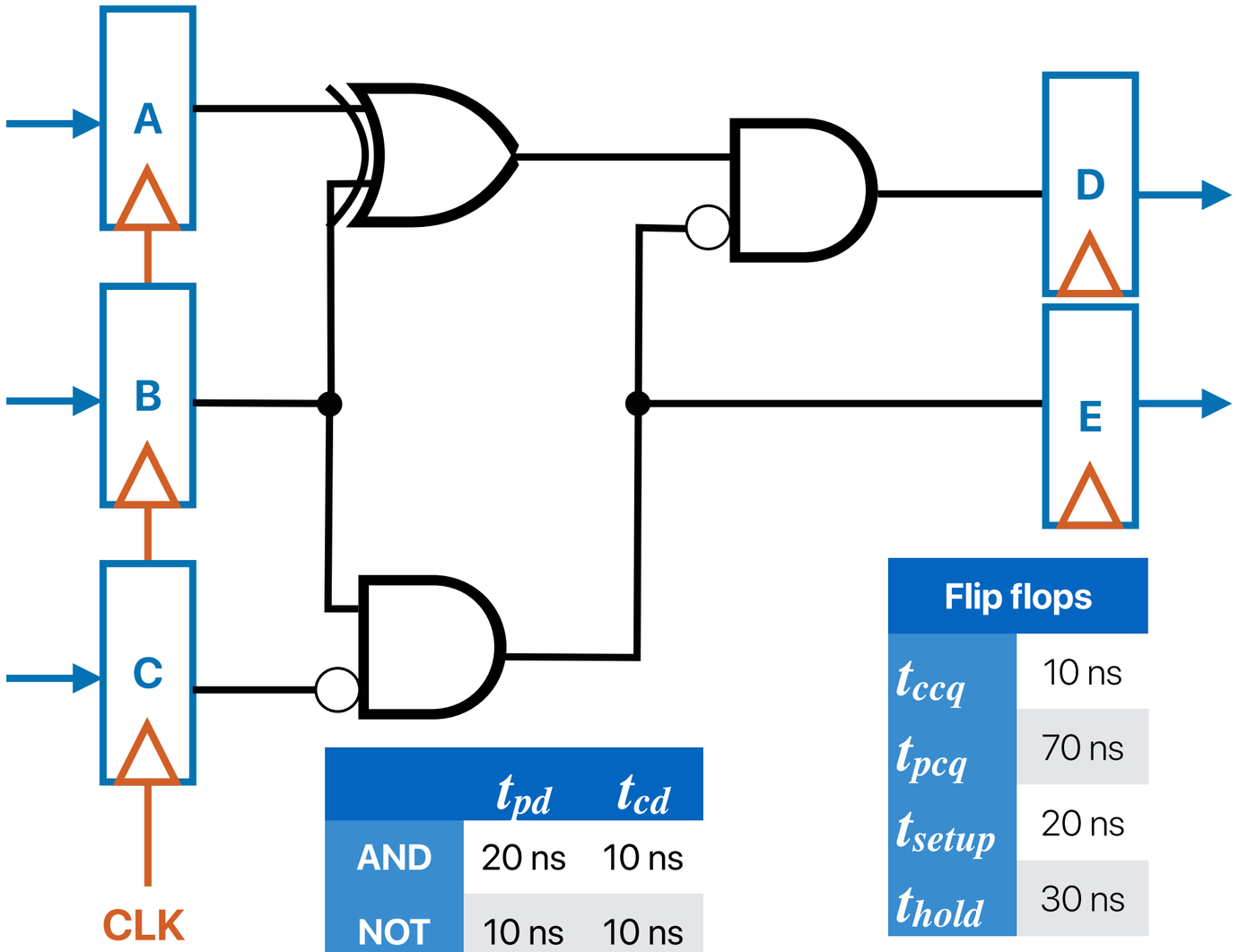
How to support shift left?

- Refer to the shift right logic, what do we need to modify to perform shift left?
  A. We can alter the interpretation of shamt to support shift left
  B. We don't need to modify the circuit, just take a not on every input
  C. We don't need to modify the circuit, just change the order of inputs
  D. We don't need to modify the circuit, just change the order of outputs
  E. None of the above



Example: timing constraints

- Where to place a buffer with t_{cd}= t_{pd}= 25 ns to solve this hold time violation?
  A. After A
  B. After B
  C. After C
  D. Before D
  E. Before E

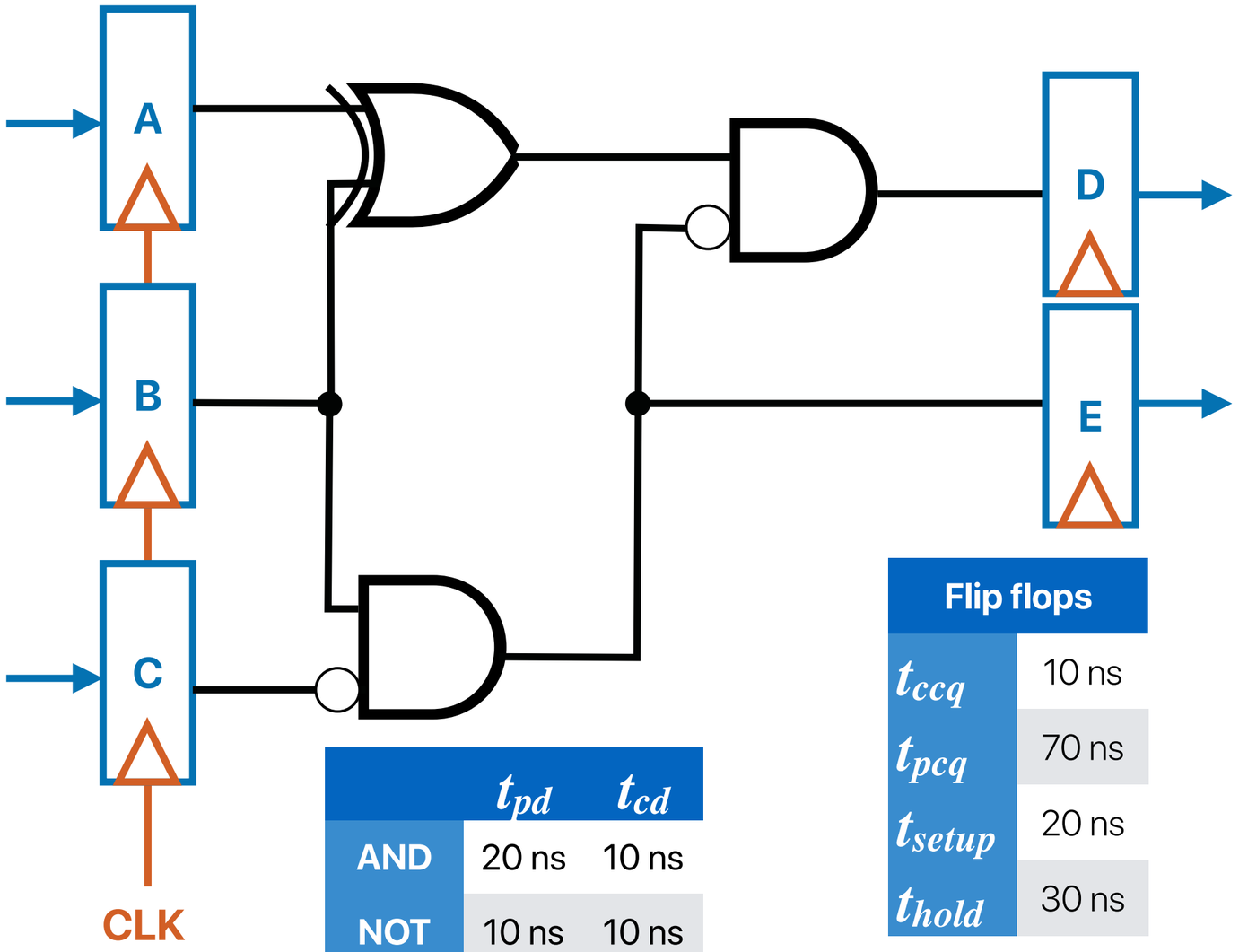


	t_{pd}	t_{cd}
AND	20 ns	10 ns
NOT	10 ns	10 ns
XOR	110 ns	50 ns

Flip flops	
t_{ccq}	10 ns
t_{pcq}	70 ns
t_{setup}	20 ns
t_{hold}	30 ns

Example: timing constraints

- Assume we have a 25 ns buffer placed before E, but now with 20 ns clock skew. Are we having hold time violation?
  A. Yes
  B. No



	t_{pd}	t_{cd}
AND	20 ns	10 ns
NOT	10 ns	10 ns
XOR	110 ns	50 ns

Flip flops	
t_{ccq}	10 ns
t_{pcq}	70 ns
t_{setup}	20 ns
t_{hold}	30 ns

The clock rate of the processor (2)

- If we are designing a processor that supports shift, add, mul operations with the following datapath components, what's the expected processor clock rate?
  - Register — 0.1 ns
  - Shifter — 0.2 ns
  - Serial 4-bit CLA — 0.3 ns
  - Serial 32-bit shift-and-add multiplier — 0.9 ns

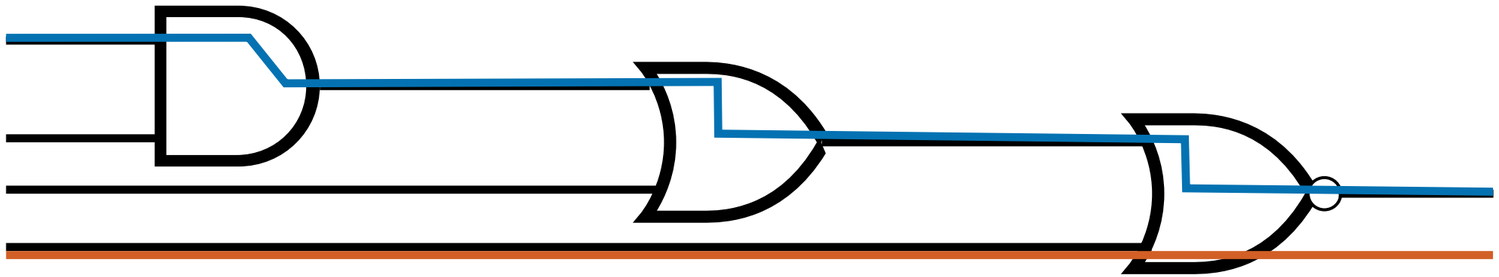
  A. ~ 10 GHz
  B. ~ 5 GHz
  C. ~ 1 GHz
  D. ~ 500 MHz
  E. ~ 50 MHz

How many cycles for an add?

- If we are designing a processor that supports shift, add, mul operations with the following datapath components and the clock rate is set to 1GHz, what's the expected number of cycles we need for a 32-bit add operation?
  - Register — 0.1 ns
  - Shifter — 0.2 ns
  - Serial 8-bit CLA — 0.4 ns
  - Serial 32-bit shift-and-add multiplier — 0.9 ns

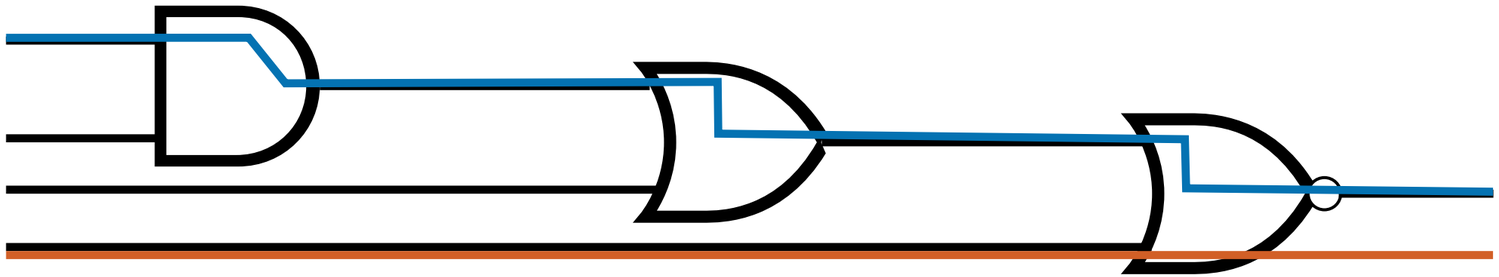
  A. 1
  B. 2
  C. 4
  D. 8
  E. 16

Combinational Logic: Output Timing Constraints



- Which path in the above circuit determines the **contamination** delay of the circuit (assuming the delay of all the gates is the same)?
  A. Blue path
  B. Red path
  C. Both
  D. Neither

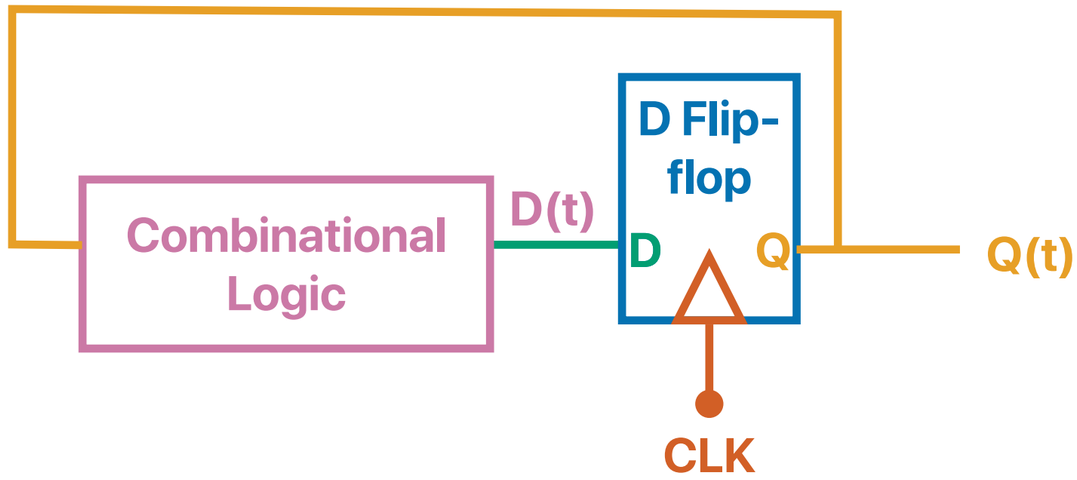
Combinational Logic: Output Timing Constraints



- Which path in the above circuit determines the **propagation** delay of the circuit (assuming the delay of all the gates is the same)?
  A. Blue path
  B. Red path
  C. Both
  D. Neither

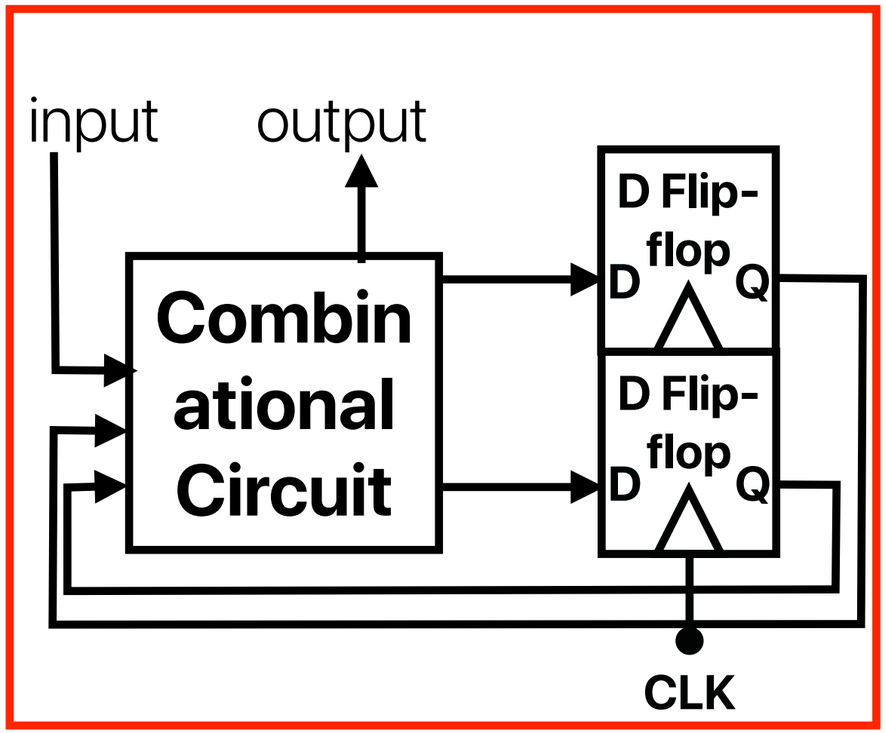
Setup-time violation

- The timing of which of the following signals can cause a setup-time violation?
  A. The input signal D(t)
  B. The output signal Q(t)
  C. Both of the above
  D. None of the above

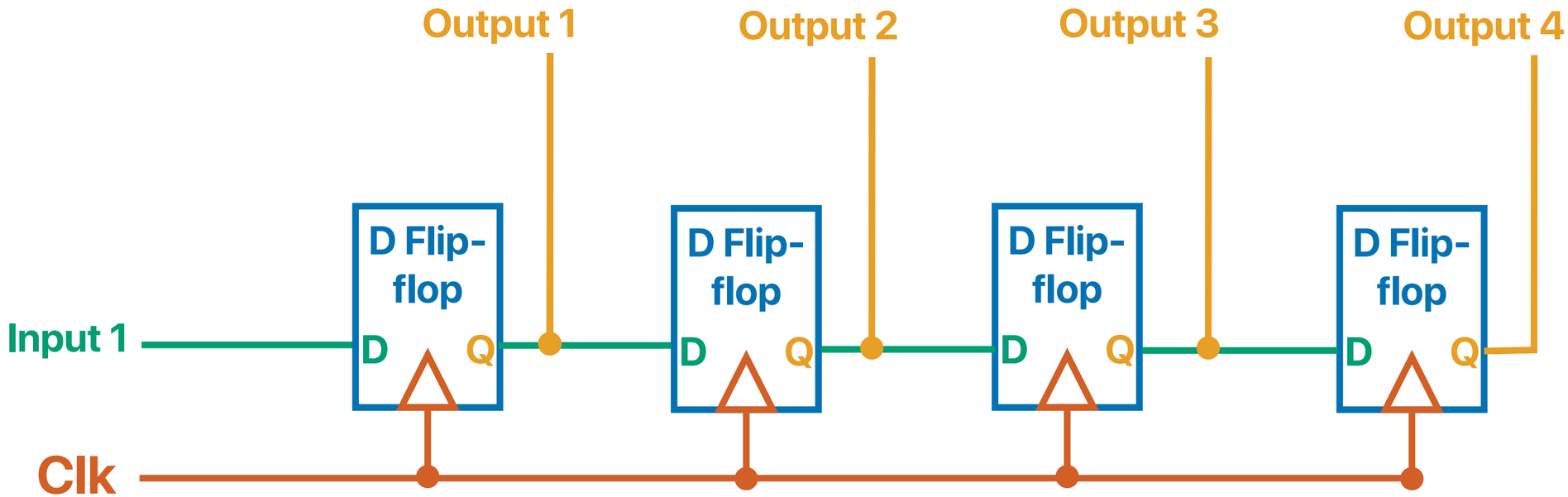


Excitation Table

- Excitation table is basically the truth table describing the combinational circuit that provides inputs for the flip-flops in the sequential circuit. How many rows are there in the excitation table of Life on Mars?
  A. 2
  B. 3
  C. 4
  D. 8
  E. 32



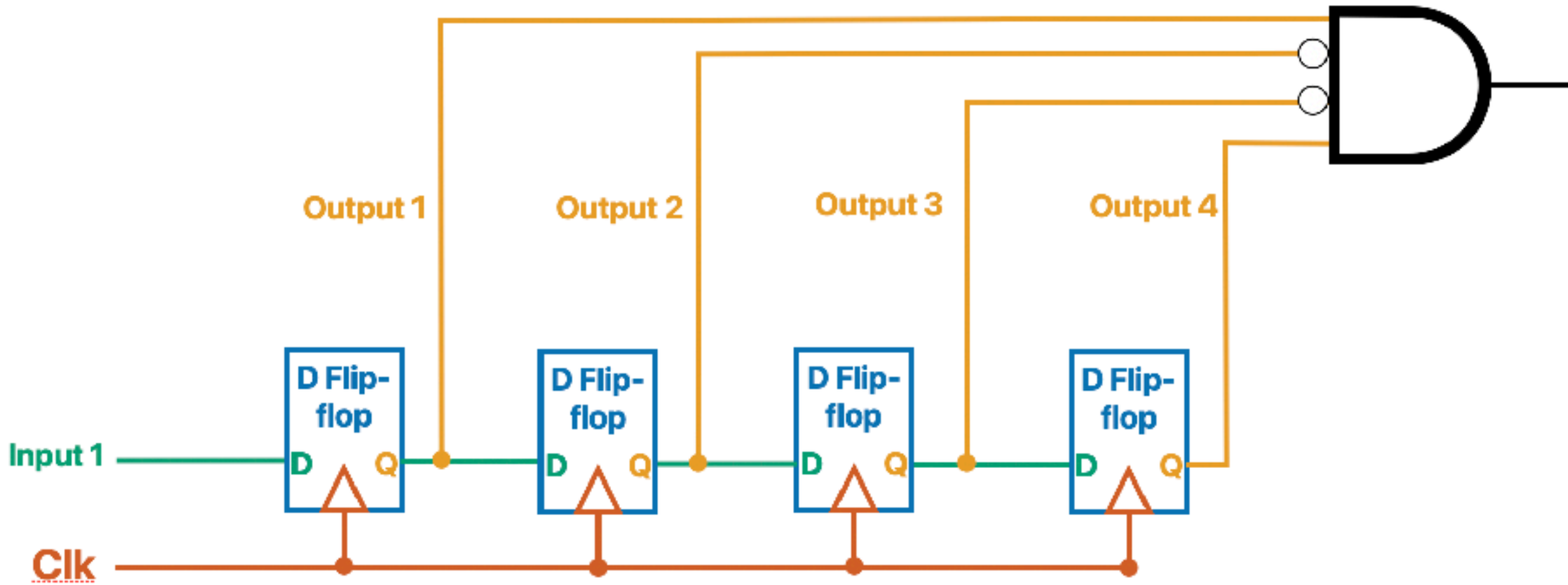
What will we output 4 cycles later?



- For the above D-FF organization, what are we expecting to see in (O1,O2,O3,O4) in the beginning of the 5th cycle after receiving (1,0,1,1)?
  A. (1,1,1,1)
  B. (1,0,1,1)
  C. (1,1,0,1)
  D. (0,0,1,0)
  E. (0,1,0,0)

Let's play with the shift register more…

- For the extended shift register, what sequence of input will the let the circuit output "1"?
  A. (1, 1, 1, 1)
  B. (0, 1, 0, 1)
  C. (1, 0, 1, 0)
  D. (0, 1, 1, 0)
  E. (1, 0, 0, 1)



Register v.s. DRAM v.s. SRAM

- Consider the following memory elements
  - ① 64*64-bit Registers
  - ② 512B SRAM
  - ③ 512B DRAM
  - A. Area: (1) > (2) > (3) Delay: (1) < (2) < (3)
  - B. Area: (1) > (3) > (2) Delay: (1) < (3) < (2)
  - C. Area: (3) > (1) > (2) Delay: (1) < (3) < (2)
  - D. Area: (3) > (2) > (1) Delay: (3) < (2) < (1)
  - E. Area: (2) > (3) > (1) Delay: (2) < (3) < (1)

Recap: Why are digital computers more popular now?

- Please identify how many of the following statements explains why digital computers are now more popular than analog computers.
  - ① The cost of building systems with the same functionality is lower by using digital computers.
  - ② Digital computers can express more values than analog computers.
  - ③ Digital signals are less fragile to noise and defective/low-quality components.
  - ④ Digital data are easier to store.

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Let’s practice!

- X, Y are two Boolean variables. Consider the following function:
  X • Y + X
  How many of the following the input values of X and Y can lead to an output of 1
  - ① X = 0, Y = 0
  - ② X = 0, Y = 1
  - ③ X = 1, Y = 0
  - ④ X = 1, Y = 1
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

A Boolean equation is converted to a circuit in what order?

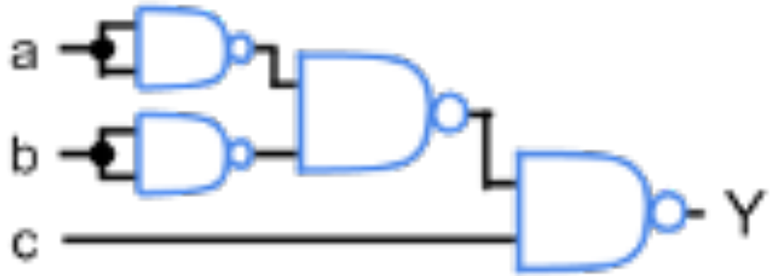
- A Boolean equation is converted to a circuit in what order
  A. Items within parentheses, then NOT, then AND, then OR.
  B. OR, then NOT, then AND, then items within parentheses.
  C. Items within parentheses, then AND, then OR, then NOT.
  D. NOT, then items within parentheses, then AND, then OR.

Boolean Equation from Truth Table

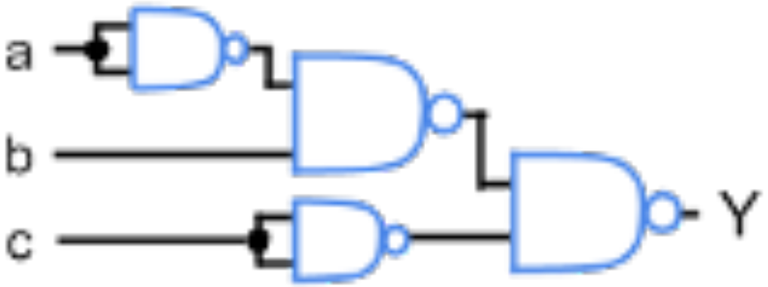
- Which equation best captures the following logic: Bob will pass the class only if doing all of the following: Bob attends all lectures, completes all assignments, passes all exams. Inputs: A = 1 indicates attends all lectures, Z = 1 indicates completes all assignments, E = 1 indicates passes all exams Outputs: P = 1 indicates passes the class
  A. P = A AND Z OR NOT(E)
  B. P = A OR Z OR E
  C. P = A AND Z OR E
  D. P = A AND Z AND E

This equation Y = (a' + b)c is implemented by which circuit?

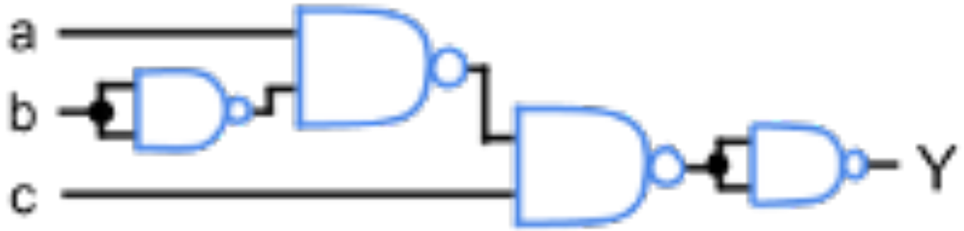
(1)



(2)



(3)



(4)

The sum-of-product form of the full adder

- How many of the following minterms are part of the sum-of-product form of the full adder in generating the output bit?
  - ① A′B′Cin′
  - ② A′BCin′
  - ③ AB′Cin′
  - ④ ABCin′
  - ⑤ A′B′Cin
  - ⑥ A′BCin
  - ⑦ AB′Cin
  - ⑧ ABCin
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

Practicing 2-variable K-map

- What's the simplified function of the following K-map?
  A. A′
  B. A′B
  C. AB′
  D. B
  E. A

A / B	0	1
0	0	0
1	1	1

Valid K-Maps

- How many of the followings are "valid" K-Maps?

(1)

	0,0	0,1	1,1	1,0
0	0	1	0	1
1	1	0	1	0

(2)

	0,1	1,1	1,0	0,0
0	1	0	1	0
1	0	1	0	1

(3)

	1,1	1,0	0,1	0,0
0	0	1	1	0
1	1	0	0	1

(4)

	0,0	0,1	1,0	1,1
0	0	1	1	0
1	1	0	0	1

A. 0
B. 1
C. 2
D. 3
E. 4

Minimum number of SOP terms

- Minimum number of SOP terms to cover the following function?
  A. 1
  B. 2
  C. 3
  D. 4
  E. 5

Input			Output
A	B	C	
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**0**
1	1	1	**0**

Minimum number of SOP terms

- Minimum number of SOP terms to cover the following function?
  A. 1
  B. 2
  C. 3
  D. 4
  E. 5

Input			Output
A	B	C	
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

Minimum SOP terms

- What's the minimum sum-of-products expression of the given truth table?
  A. A′B′C′ + A′BC′+ A′BC + AB′C′
  B. A′B′C + AB + AC
  C. AB′C′ + B′C′
  D. A′B + B′C′
  E. A′C′ + A′B + AB′C′

Input			Output
A	B	C	
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

4-variable K-map

- What's the minimum sum-of-products expression of the given K-map?
  A. B′C′ + A′B′
  B. B′C′D′ + A′B′ + B′C′D′
  C. A′B′CD′ + B′C′
  D. AB′ + A′B′ + A′B′D′
  E. B′C′ + A′C′D′

		A′B′	A′B	AB	AB′
		00	01	11	10
C′D′	00	1	0	0	1
C′D	01	1	0	0	1
CD	11	0	0	0	0
CD′	10	1	1	0	0

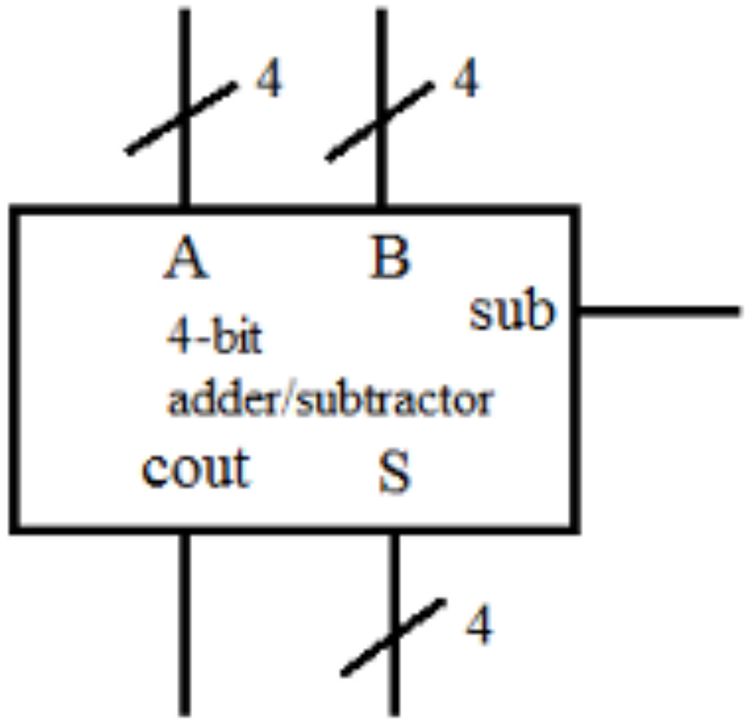
LT?

- What's the minimum SOP presentation of LT?
  A. A′B′D′ + AC′ + BCD
  B. A'B'D + A'C + B′CD
  C. A'B'C'D' + A'BC'D + ABCD + AB′CD′
  D. ABCD + AB′CD′ + A′B′C′D′ + A′BC′D
  E. BC'D' + AC' + ABD'

Input				Output		
A	B	C	D	LT	EQ	GT
0	0	0	0	**0**	**1**	**0**
0	0	0	1	**1**	**0**	**0**
0	0	1	0	**1**	**0**	**0**
0	0	1	1	**1**	**0**	**0**
0	1	0	0	**0**	**0**	**1**
0	1	0	1	**0**	**1**	**0**
0	1	1	0	**1**	**0**	**0**
0	1	1	1	**1**	**0**	**0**
1	0	0	0	**0**	**0**	**1**
1	0	0	1	**0**	**0**	**1**
1	0	1	0	**0**	**1**	**0**
1	0	1	1	**1**	**0**	**0**
1	1	0	0	**0**	**0**	**1**
1	1	0	1	**0**	**0**	**1**
1	1	1	0	**0**	**0**	**1**
1	1	1	1	**0**	**1**	**0**

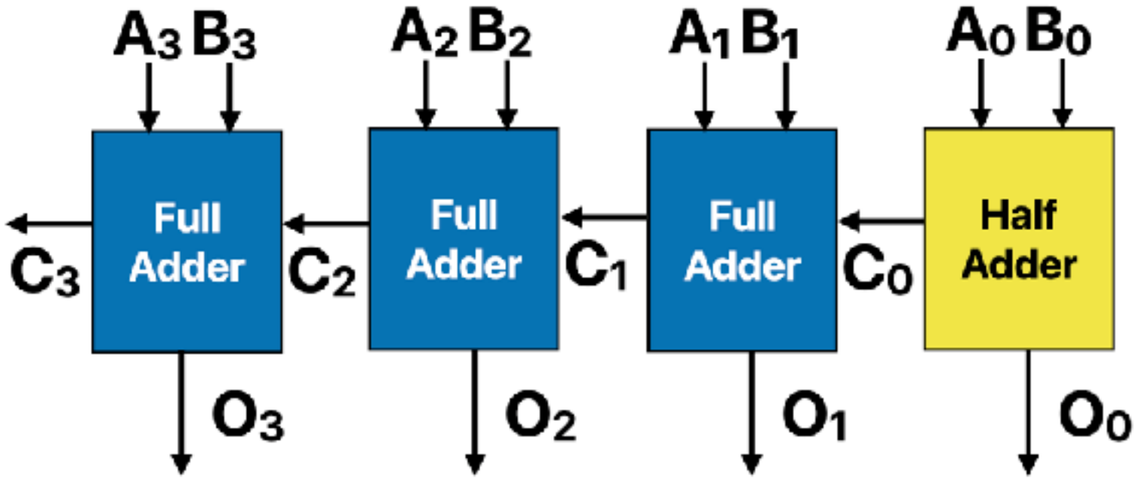
Input/output of a design

- A 4-bit adder/subtractor has inputs A = 0100, and B = 0010. What value of sub outputs sum S = 0110 and cout = 0000?
  A. 0
  B. 1
  C. 0000
  D. 1111



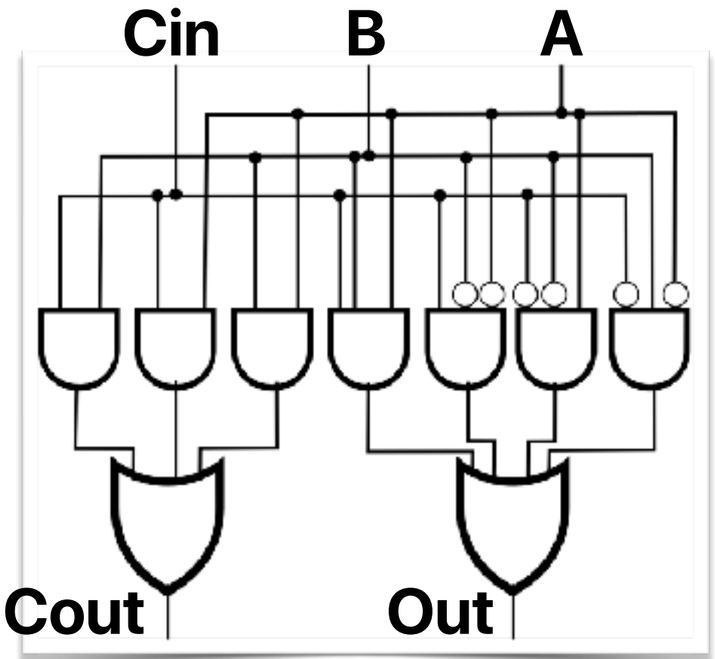
If we want to support subtraction?

- If we would like to extend the 4-bit adder that we've built before to support "A-B" with 2's complement, how many of the followings should we add at least?
  - ① Provide an option to use bitwise NOT A
  - ② Provide an option to use bitwise NOT B
  - ③ Provide an option to use bitwise A XOR B
  - ④ Provide an option to add 0 to the input of the half adder
  - ⑤ Provide an option to add 1 to the input of the half adder
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5



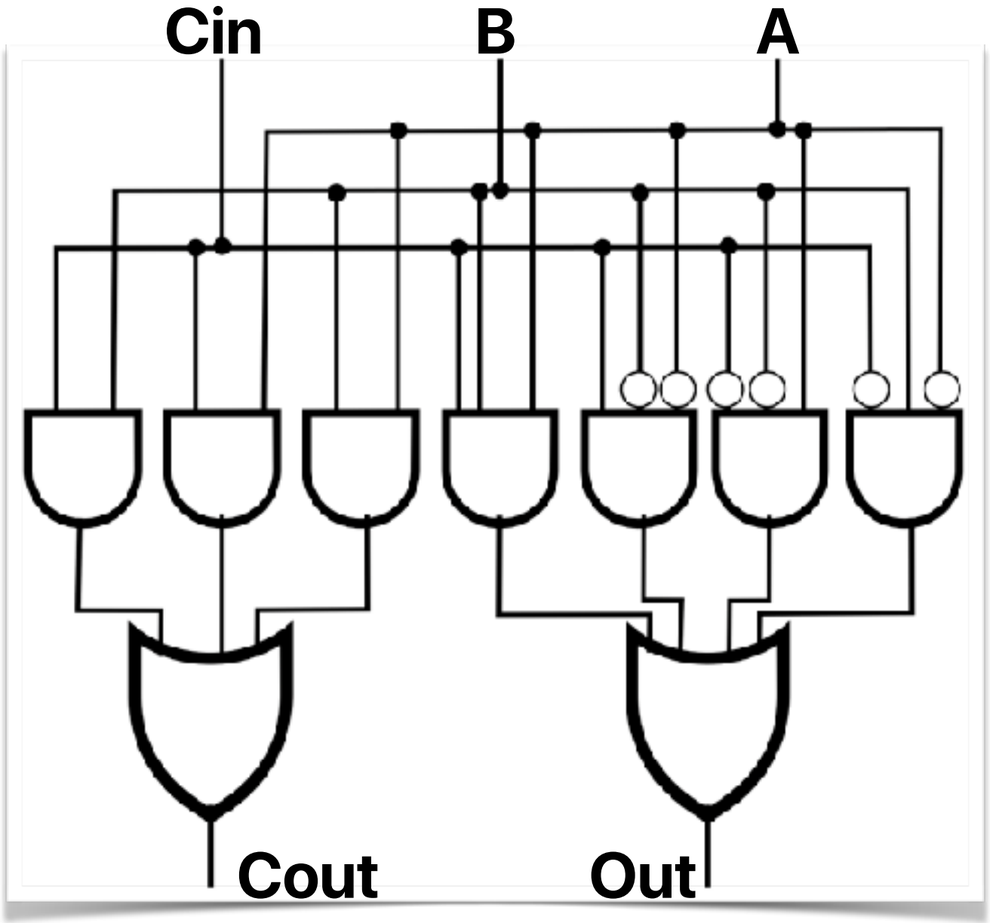
How efficient is the adder?

- One approach estimates transistors, assuming every gate input requires 2 transistors, and ignoring inverters for simplicity. A 2-input gate requires 2 inputs · 2 trans/input = 4 transistors. A 3-input gate requires 3 · 2 = 6 transistors. A 4-input gate: 8 transistors. Wires also contribute to size, but ignoring wires as above is a common approximation.
- Considering the shown 1-bit full adder and use it to build a 32-bit adder, how many transistor do we need?
  A. 1152
  B. 1600
  C. 1664
  D. 1792
  E. 1984



How efficient is the adder?

- Considering the shown 1-bit full adder and use it to build a 32-bit adder, how many gate-delays are we suffering to getting the final output?
  A. 2
  B. 32
  C. 64
  D. 128
  E. 288



CLA's gate delay

- What's the gate-delay of a 4-bit CLA?
  A. 2
  B. 4
  C. 6
  D. 8
  E. 10

$$G_i = A_i B_i$$

$$P_i = A_i \text{ XOR } B_i$$

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2$$
$$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$C_4 = G_3 + P_3 C_3$$
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$

CLA's size

- How many transistors do we need to implement a 4-bit CLA logic?
  A. 38
  B. 64
  C. 88
  D. 116
  E. 128

$$G_i = A_i B_i$$

$$P_i = A_i \text{ XOR } B_i$$

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2$$
$$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$C_4 = G_3 + P_3 C_3$$
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$
$$+ P_3 P_2 P_1 P_0 C_0$$

How big is the 4-bit 4:1 MUX?

- How many estimated transistors are there in the 4-bit 4:1 MUX?
  A. 48
  B. 64
  C. 80
  D. 128
  E. 192

Gate delay of 8:1 MUX

- What's the estimated gate delay of an 8:1 MUX?
  A. 1
  B. 2
  C. 4
  D. 8
  E. 16

16-1 MUX

- How many AND gates does a 16x1 mux require?
  A. 2
  B. 4
  C. 8
  D. 16

IEEE 754 format

- Realign the number into 1.**F** * 2^{e}
- Exponent stores **e** + 127
- Fraction only stores **F**
- Convert the following number
  1 1000 0010 0100 0000 0000 0000 0000 000
  A. - 1.010 * 2^130
  B. -10
  C. 10
  D. 1.010 * 2^130
  E. None of the above

Clock signal

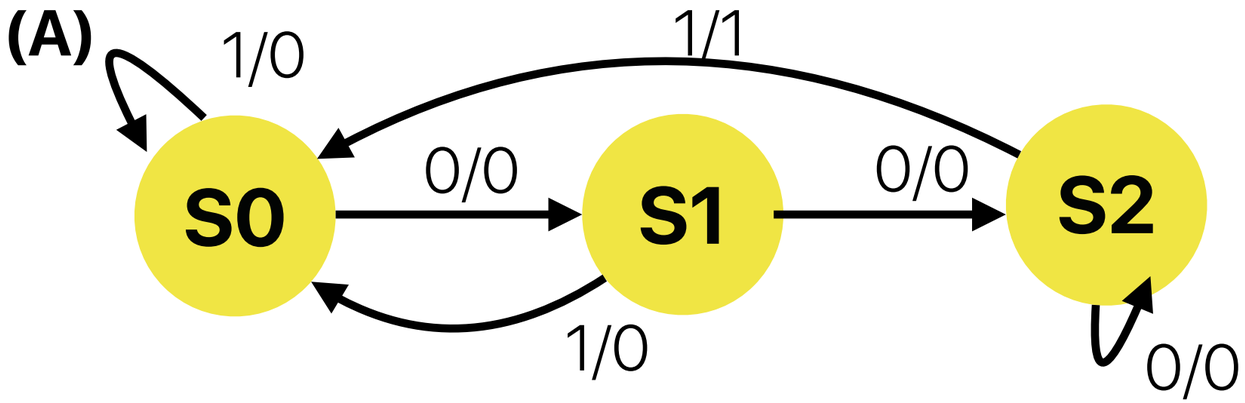
- Regarding the above clock signal, please identify how many of the following statements are correct?
  - ① Clock period of 4ns with 250MHz frequency
  - ② Clock duty cycle 75%
  - ③ Clock period of 1ns with 1GHz frequency
  - ④ The above contains two complete clock cycles.
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

FSM for Life on Mars

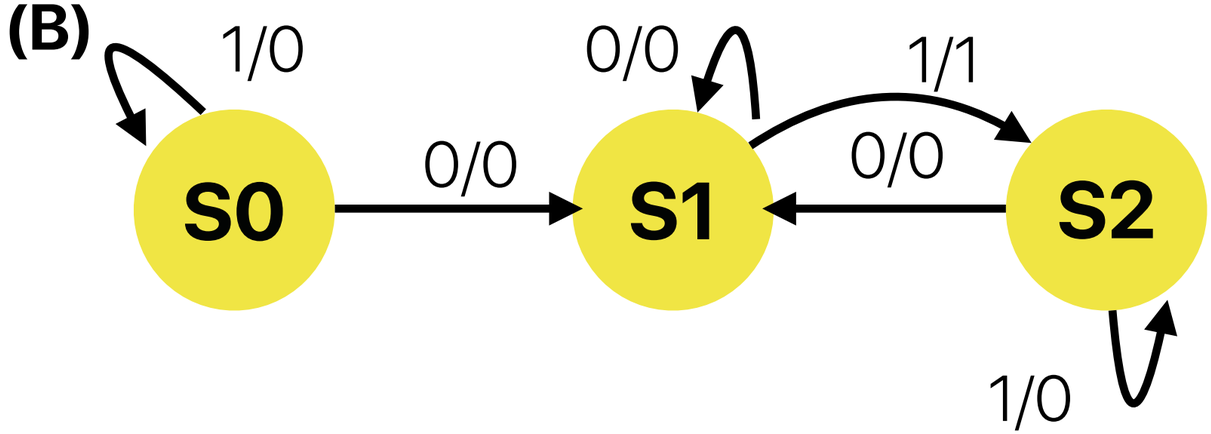
1/0 == Input 1/Output 0

- Which of the following diagrams is a correct FSM for the 001 pattern recognizer on the Mars rover? (If sees "001", output "1")

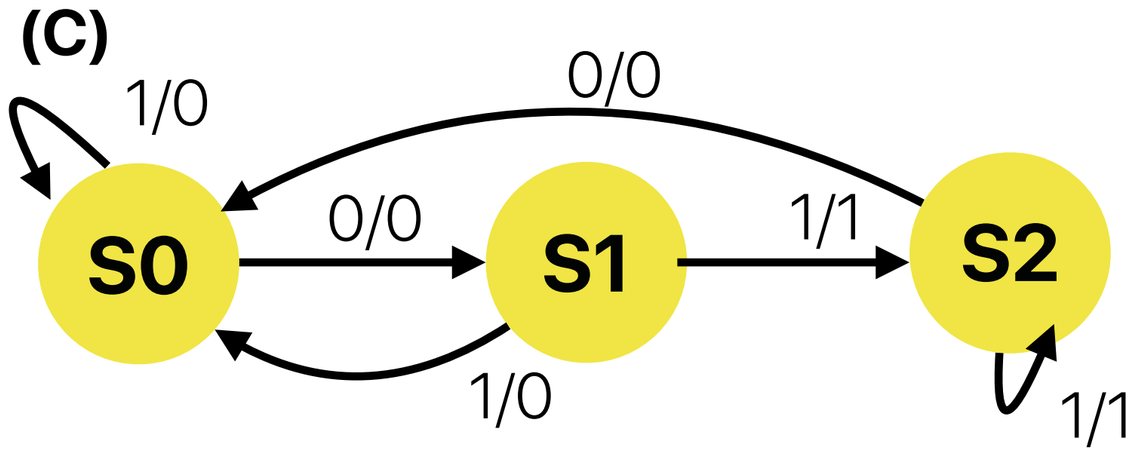
(A)



(B)



(C)

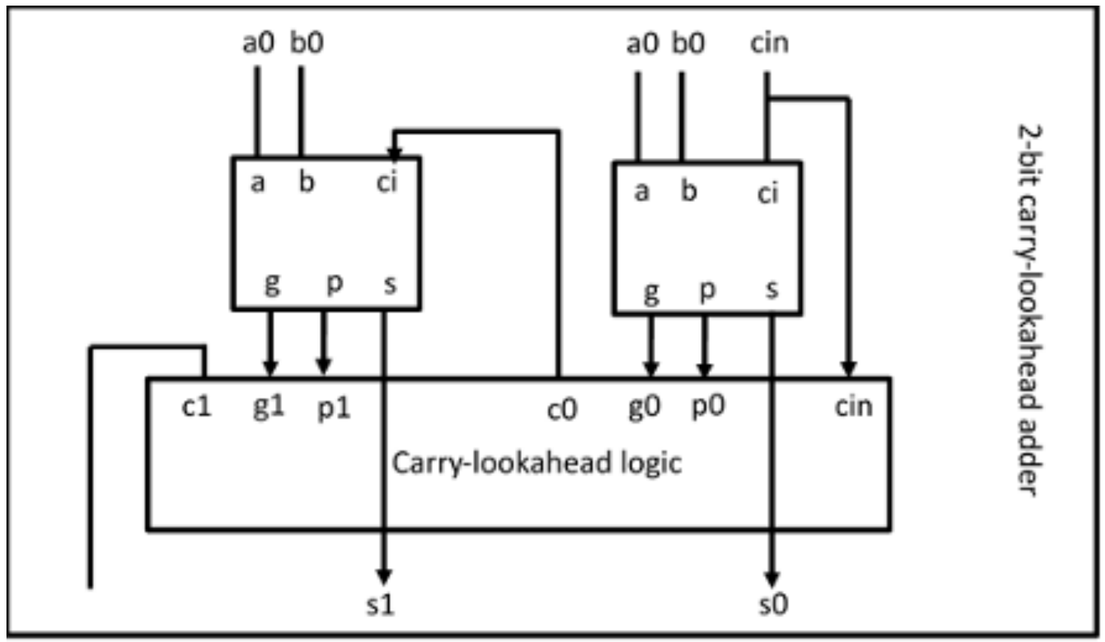


(D) All are correct

(E) None is correct

2-bit CLA

- Which is true about the given 2-bit carry-lookahead adder? Hint: g = ab, p = a + b, and the expression for each digit's carry-out is co = ab + (a + b)ci = g + p·ci.
  A. c0 = 0, when a0 = 1, b0 = 1, and cin = 0
  B. c0 = 1, when a0 = 1, b0 = 1, and cin = 1
  C. c1 = 1, when cin = 1, g0 = 1, p0 = 1, g1 = 0, and p1 = 0
  D. c1 = 0, when cin = 0, g0 = 1, p0 = 1, g1 = 1, and p1 = 1



What's after shift?

- Assume we have a data type that stores 8-bit unsigned integer (e.g., unsigned char in C). How many of the following C statements and their execution results are correct?

	Statement	C = ?
I	`c = 3; c = c >> 2;`	1
II	`c = 255; c = c << 2;`	252
III	`c = 256; c = c >> 2;`	64
IV	`c = 128; c = c << 1;`	1

A. 0
B. 1
C. 2
D. 3
E. 4

The advantage of floating/fixed point

- Regarding the pros of floating point and fixed point expressions, please identify the correct statement
  A. Fixed point can be express wider range of numbers than floating point numbers, but the hardware design is more complex
  B. Floating point can be express wider range of numbers than floating point numbers, but the hardware design is more complex
  C. Fixed point can be express wider range of numbers than floating point numbers, and the hardware design is simpler
  D. Floating point can be express wider range of numbers than floating point numbers, and the hardware design is simpler

4-variable K-map

- What's the minimum sum-of-products expression of the given K-map?
  A. B′C′ + A′B′
  B. B′C′D′ + A′B′ + B′C′D′
  C. A′B′CD′ + B′C′
  D. AB′ + A′B′ + A′B′D′
  E. B′C′ + A′C′D′

		A′B′	A′B	AB	AB′
		00	01	11	10
C′D′	00	1	0	0	1
C′D	01	1	0	0	1
CD	11	0	0	0	0
CD′	10	1	1	0	0

Valid K-Maps

- How many of the followings are "valid" K-Maps?

(1)

	0,0	0,1	1,1	1,0
0	0	1	0	1
1	1	0	1	0

(2)

	0,1	1,1	1,0	0,0
0	1	0	1	0
1	0	1	0	1

(3)

	1,1	1,0	0,1	0,0
0	0	1	1	0
1	1	0	0	1

(4)

	0,0	0,1	1,0	1,1
0	0	1	1	0
1	1	0	0	1

A. 0
B. 1
C. 2
D. 3
E. 4

Multiple choices + explanations

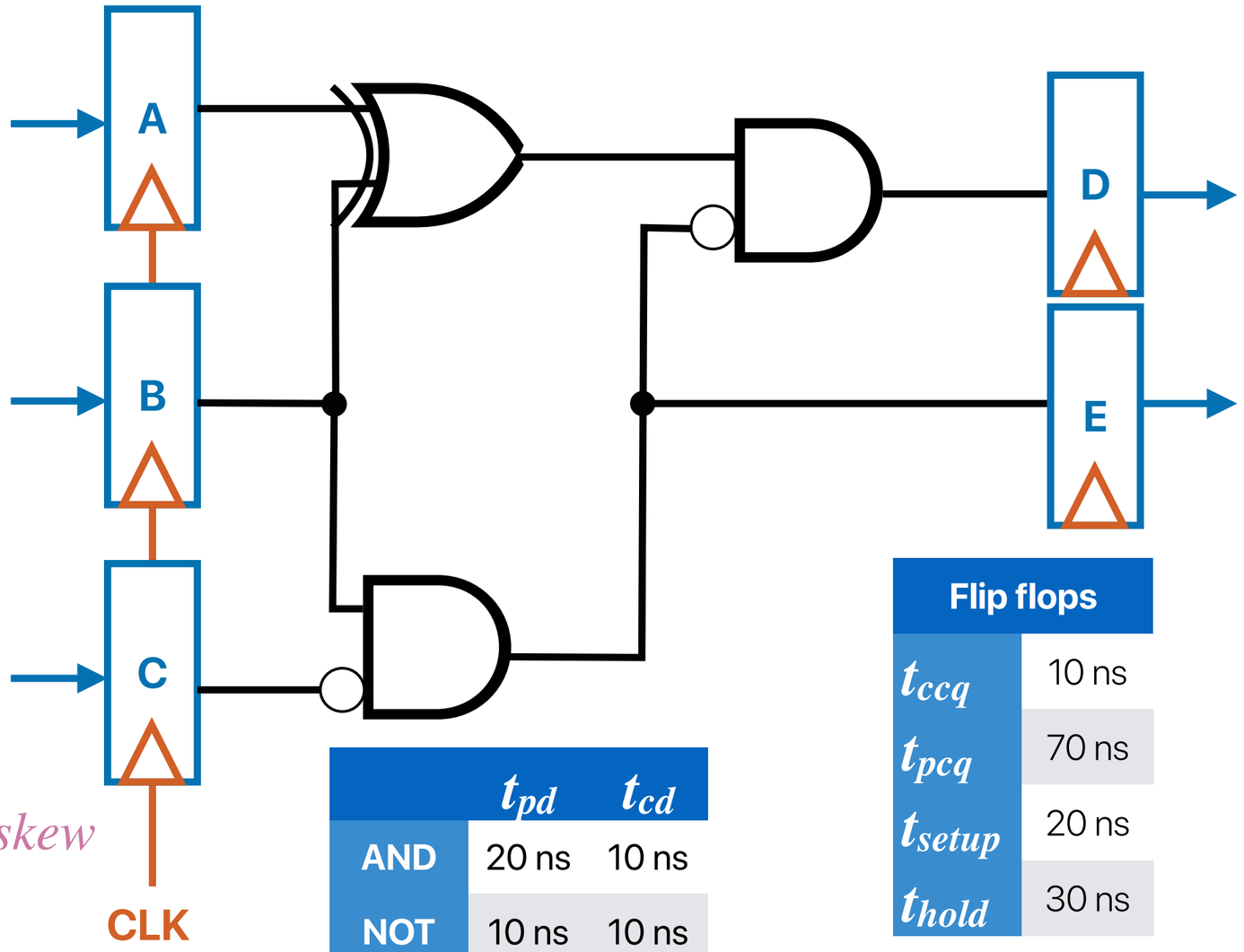
Example: timing constraints

- Does the circuit have a hold violation?
  A. Yes
  B. No

Why?

$$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

$$t_{ccq} + t_{cd} > t_{hold} + t_{skew}$$



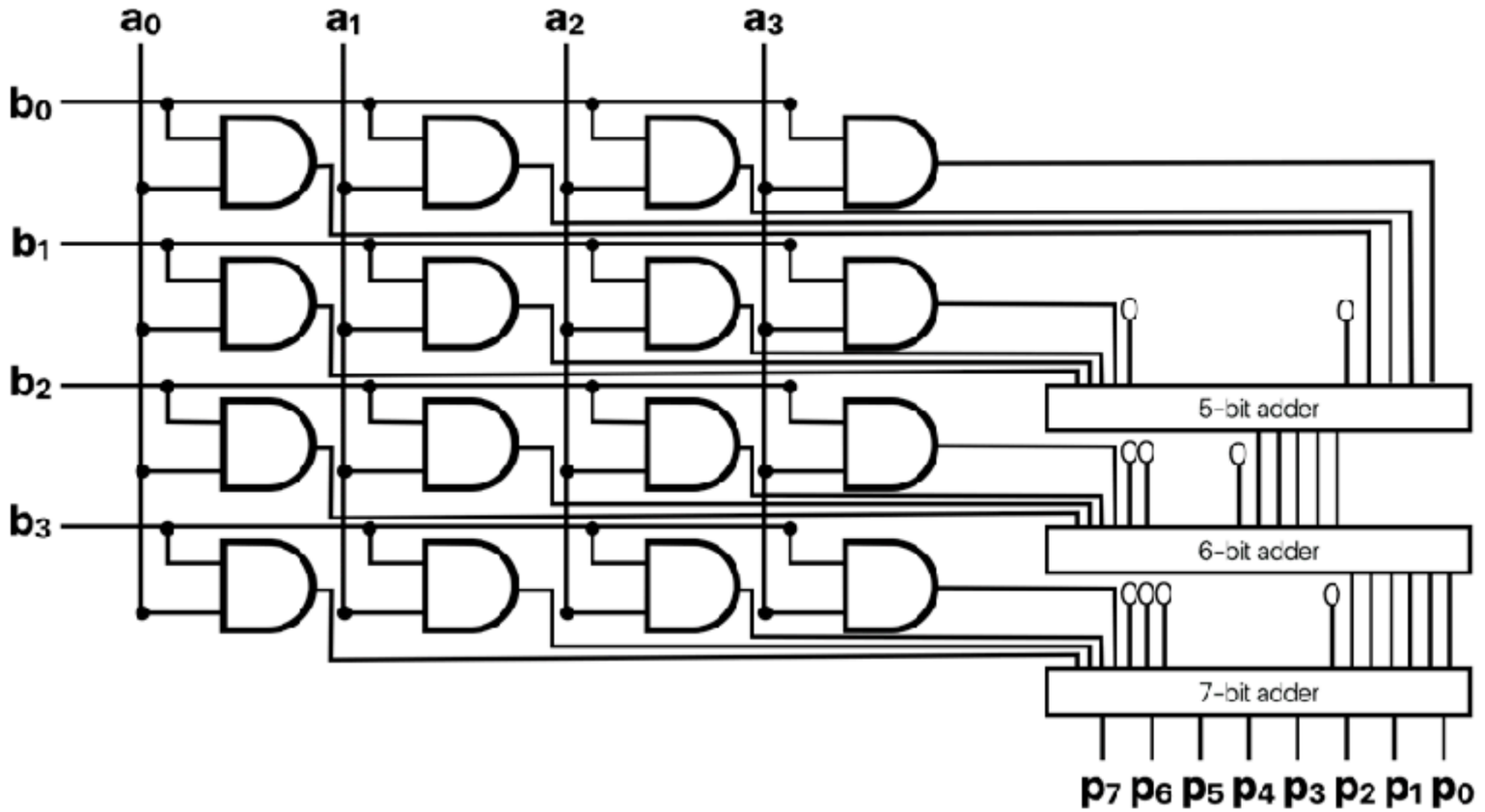
	t_{pd}	t_{cd}
AND	20 ns	10 ns
NOT	10 ns	10 ns
XOR	110 ns	50 ns

Flip flops	
t_{ccq}	10 ns
t_{pcq}	70 ns
t_{setup}	20 ns
t_{hold}	30 ns

Gate-delays of Array-style Multipliers

- What's the estimated gate-delay of the 4-bit multiplier? (Assume adders are composed of 4-bit CLAs)
  - A. 9
  - B. 12
  - C. 13
  - D. 15
  - E. 16

Why?



The clock rate of the processor

- If we are designing a processor that supports shift, add, mul operations with the following datapath components, what's the expected processor clock rate?
  - Register — 0.1 ns
  - Shifter — 0.2 ns
  - 32-bit CLA — 1.6 ns
  - 32-bit hierarchical multiplier — 16 ns

  A. ~ 10 GHz
  B. ~ 5 GHz
  C. ~ 500 MHz
  D. ~ 50 MHz
  E. ~ 5 MHz

Why?

Area/Delay of adders

- Consider the following adders?
  - ① 32-bit CLA made with 8 4-bit CLA adders
  - ② 32-bit CRA made with 32 full adders
  - ③ 32-bit serial adders made with 4-bit CLA adders
  - ④ 32-bit serial adders made with 1-bit full adders

A. Area: (1) > (2) > (3) > (4) Delay: (1) < (2) < (3) < (4)

B. Area: (1) > (3) > (2) > (4) Delay: (1) < (3) < (2) < (4)

C. Area: (1) > (3) > (4) > (2) Delay: (1) < (3) < (4) < (2)

D. Area: (1) > (2) > (3) > (4) Delay: (1) < (3) < (2) < (4)

E. Area: (1) > (3) > (2) > (4) Delay: (1) < (3) < (4) < (2)

Why?

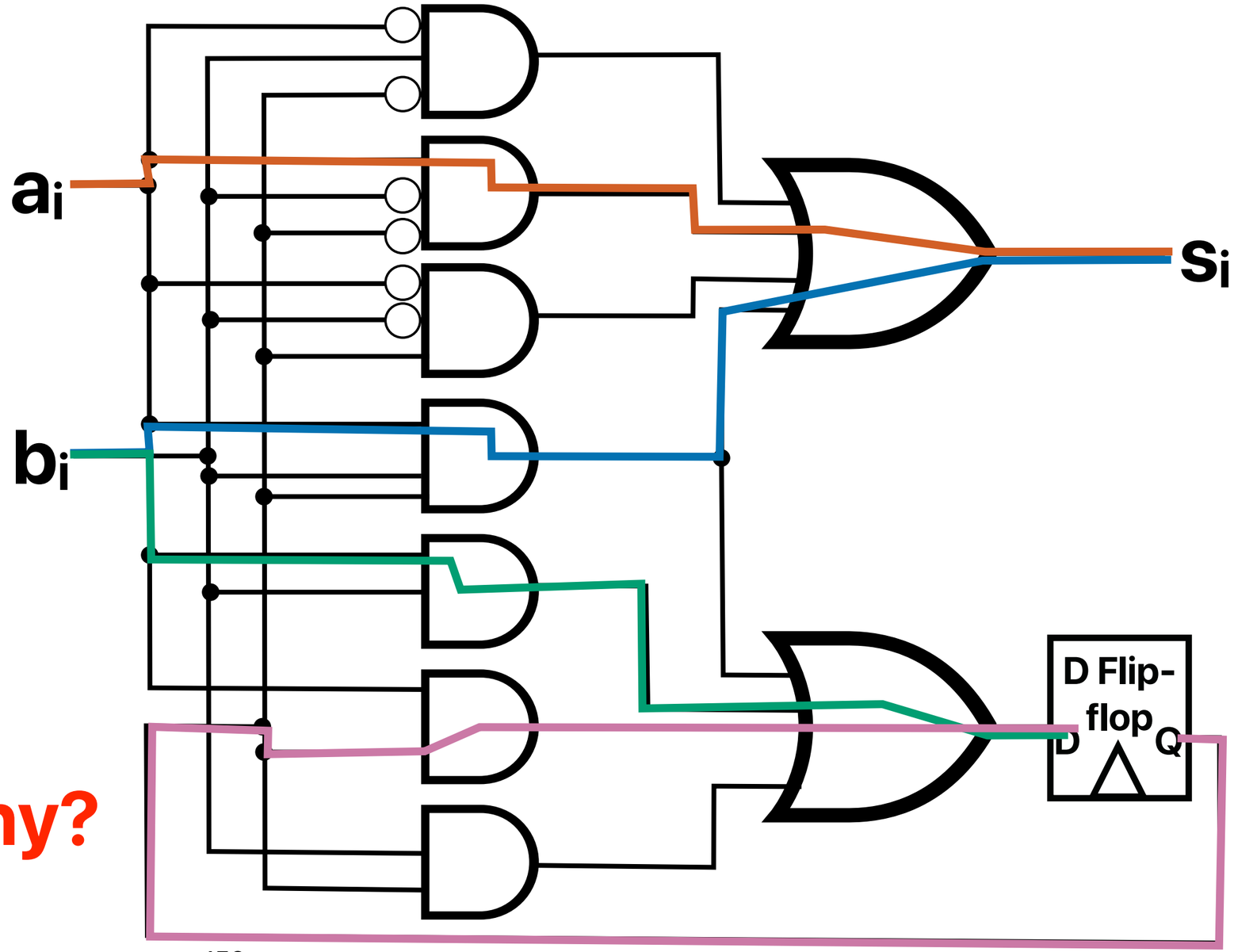
Throughput

- Consider the following adders. Assume each gate delay is 1ns and the delay in a register is also 1ns. And we are processing millions of add operations. Please rank their throughputs.
  - ① 32-bit CLA made with 8 4-bit CLA adders
  - ② 32-bit CRA made with 32 full adders
  - ③ 8-stage, pipelined 32-bit serial adders made with 4-bit CLA adders
  - ④ 32-stage, pipelined 32-bit serial adders made with 1-bit full adders
  - A. (1) > (2) > (3) > (4)
  - B. (2) > (1) > (4) > (3)
  - C. (3) > (4) > (2) > (1)
  - D. (4) > (3) > (2) > (1)
  - E. (4) > (3) > (1) > (2)

Why?

Cycle time of the circuit?

- Assume each gate delay is 1ns and the delay in a register is 2ns, what's the cycle time of the circuit?
  A. 2 ns
  B. 3 ns
  C. 4 ns
  D. 5 ns
  E. 6 ns



Why?

Recap: CLA's size

- How many transistors do we need to implement a 4-bit CLA logic?
  A. 38
  B. 64
  C. 88
  D. 116
  E. 128

Why?

Will the loop end? (last run)

- Consider the following C program.

```c
#include <stdio.h>

int main(int argc, char **argv)
{
    float i=1.0;
    while(i > 0) i+=i;
    printf("We're done! %f\n",i);
    return 0;
}
```

Why?

Please identify the correct statement.

A. The program will finish since i will end up to be +0
B. The program will finish since i will end up to be something < 0
C. The program will not finish since i will always be a positive non-zero number.
D. The program will not finish since i will end up staying at some special FP32 presentation
E. The program will not finish but raise an exception since we will go to NaN first.

Why stuck at 16777216?

- Consider the following C program.

```c
#include <stdio.h>

int main(int argc, char **argv)
{
    float i=1.0;
    while(i > 0) i++;
    printf("We're done! %f\n",i);
    return 0;
}
```

Why i stuck at 16777216.000?

Why?

A. It's a special number in IEEE 754 standard that an adder will treat it differently
B. It's a special number like +Inf/-Inf or +NaN/-NaN with special meaning in the IEEE 754 standard
C. It's just the maximum integer that IEEE 754 standard can represent
D. It's nothing special, but just happened to be the case that 16777216.0+1.0 will produce 16777216.0
E. It's nothing special, but just happened to be the case that 16777216.0 add anything will become 16777216.0

Are we getting the same numbers?

- For the following code, please identify how many statements are correct
  - ① We will see the same output at X and Y
  - ② X will print — 12802.454
  - ③ Y will print — 12802.454
  - ④ Neither X nor Y will print the right result, but X is closer to the right answer
  - ⑤ Neither X nor Y will print the right result, but Y is closer to the right answer
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

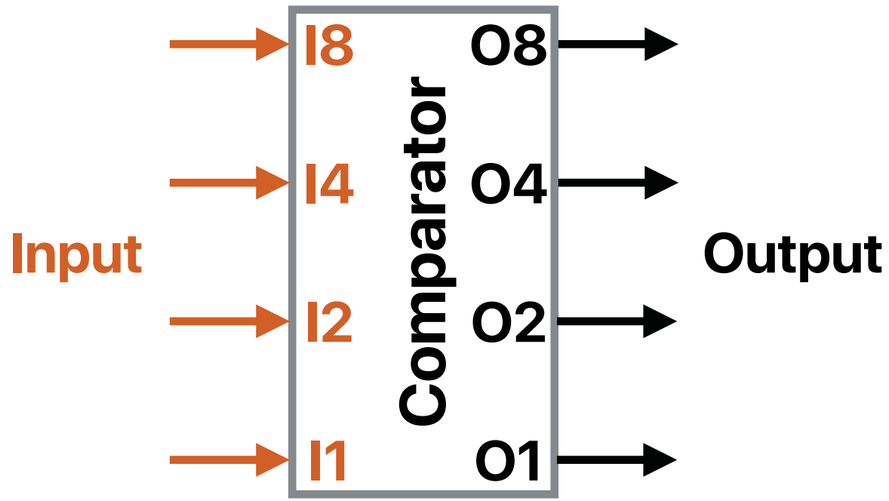
Why?

```c
#include <stdio.h>

int main(int argc, char **argv) {
    float a, b, c;
    a = 1280.245;
    b = 0.0004;
    c = (a + b)*10.0;
    printf("%f\n",c); // X
    c = a*10.0 + b*10.0;
    printf("%f\n",c); // Y
    return 0;
}
```

Free Answer Questions

BCD+1 — Binary coded decimal + 1

- 0x0 — 1
- 0x1 — 2
- 0x2 — 3
- 0x3 — 4
- 0x4 — 5
- 0x5 — 6
- 0x6 — 7
- 0x7 — 8
- 0x8 — 9
- 0x9 — 0
- 0xA — 0xF — Don’t care



Can you write the truth table?
Can you create a K-map?
Can simplify the boolean equation?

What's the output of this? and Why?

```c
#include <stdio.h>

int main(int argc, char **argv)
{
    float a, b, c, d;
    int i = 0;
    a = 1.2;
    b = 1.0;
    c = a + b;
    printf("A: %d\n", c==2.2);

    a = 33554432.0;
    b = 2.0;
    c = a+b;
    printf("B: %d\n", c, d, c==33554434.0);

    a = 1.0;
    for(i=0;i<200;i++)
        a += a;
    printf("C: %f\n", a);

    a = a/0.0;
    printf("D: %f\n", a);
    return 0;
}
```

Which side is faster? Why?

```c
struct student_record
{
    int id;
    double homework;
    double midterm;
    double final;
};

int main(int argc, char **argv)
{
    int i,j;
    double midterm_average=0.0;
    int number_of_records = 10000000;
    struct timeval time_start, time_end;
    struct student_record *records;
    records = (struct
student_record*)malloc(sizeof(struct
student_record)*number_of_records);
    init(number_of_records,records);


    for (j = 0; j < 100; j++)
        for (i = 0; i < number_of_records; i++)
            midterm_average+=records[i].midterm;

    printf("average: %lf\n",midterm_average/
number_of_records);
   free(records);
    return 0;
}
```

```c
int main(int argc, char **argv)
{
    int i,j;
    double midterm_average=0.0;
    int number_of_records = 10000000;
    struct timeval time_start, time_end;
    id = (int*)malloc(sizeof(int)*number_of_records);
    midterm = (double*)malloc(sizeof(double)*number_of_records);
    final = (double*)malloc(sizeof(double)*number_of_records);
    homework = (double*)malloc(sizeof(double)*number_of_records);
    init(number_of_records);

    for (j = 0; j < 100; j++)
        for (i = 0; i < number_of_records; i++)
            midterm_average+=midterm[i];

    free(id);
    free(midterm);
    free(final);
    free(homework);
    return 0;
}
```

Other questions to think about

- Can you design a serial shifter? What's the pros and cons?
- Can you design a pipelined floating point adder? Where would you insert pipeline registers? What do you expect that to change the performance of your processors?
- If Moore's law allows us to keep putting more transistors into the same area, you can add more pipeline stages into the current design. Do you think adding more stages helps improve performance? Why or why not?
- Why ASICs can improve system performance?
- What do you expect PLC (Peta-level cell, 5-bit in a cell) to change the behavior of flash SSD?

Announcement

- iEval — Capture your screenshot, submit through iLearn and you will receive a full credit assignment
- Assignment 6 due tonight
- Lab 6 due **this Friday**
- Please fill out ABET survey through iLearn
- Final exam will be held during the campus scheduled period to avoid conflicts
  - 6/11 11:30am — 2:59:59pm
  - About the same format as midterm, but longer

Electrical
Computer Science
Engineering

120A

つづく