Course Info

- Lab 5 is released, get yourself prepared before going to lab sessions! (FSM preview, FSM background knowledge not a must, just use sum of minterm/Karnaugh map)
- Project 1.2 is available this week, and will be marked in lab sessions. Deadline March 31th. Start early!!!
- HW3 ddl March 18th.
- This week discussion on synchronized digital system (SDS)
- Mid-term next Tuesday 8:00 am-10:00 am, details later at the beginning of second half lecture.

信息科学与技术学院
School of Information Science and Technology

CS 110
Computer Architecture
Combinational & Sequential Circuits

Instructors:

Siting Liu & Chundong Wang

Course website: https://toast-lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2023/index.html

School of Information Science and Technology (SIST)

ShanghaiTech University

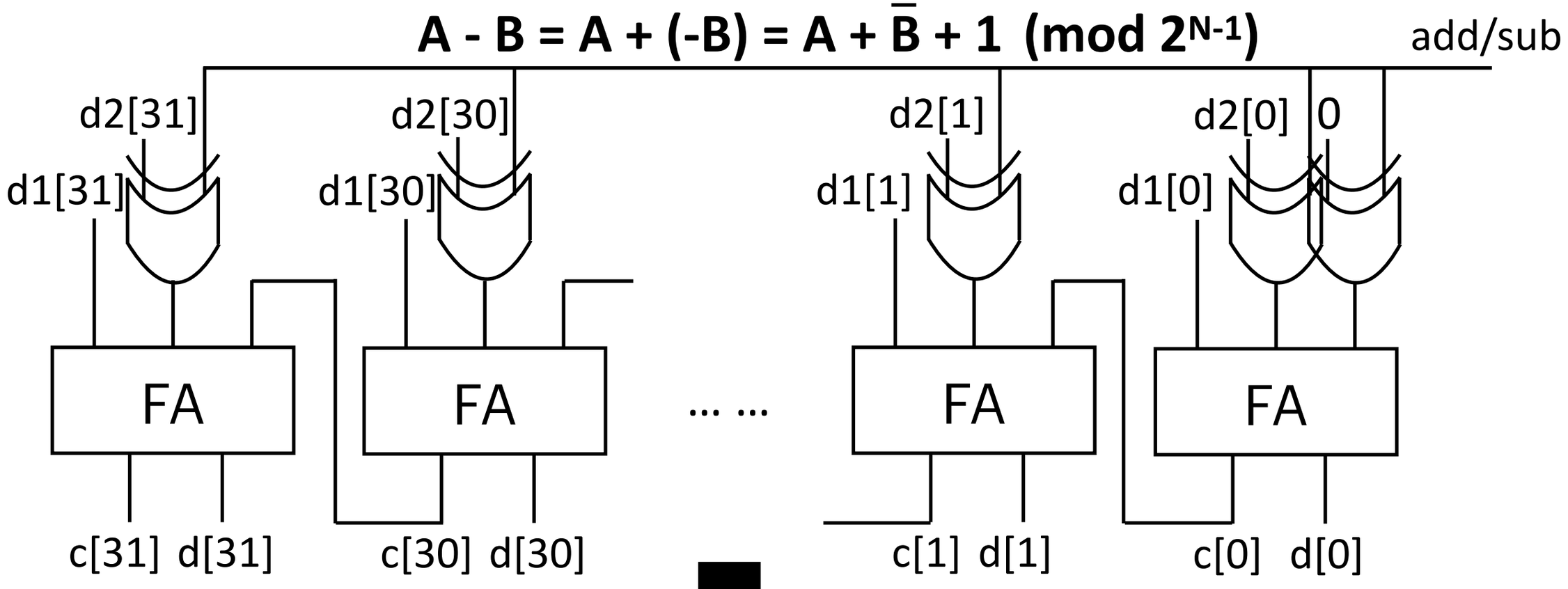
2023/3/5

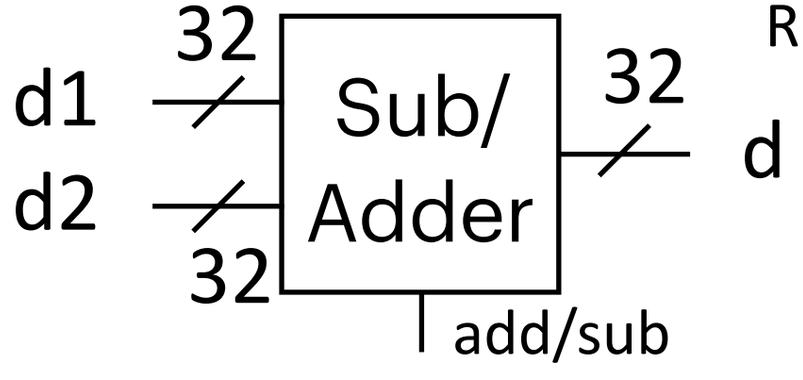
Recap: Combinational Circuits

- Combinational circuits:
  - Outputs decided only by inputs
  - Can be designed by either truth table (Karnaugh map to obtain the logic expression) or structural method (ripple-carry/domino adder)
- ALU design

An Arithmetic & Logic Unit (ALU)

- Arithmetic: Add/**Sub**/Addi
- Logic: And/Or/Xor(i) (bit-wise)

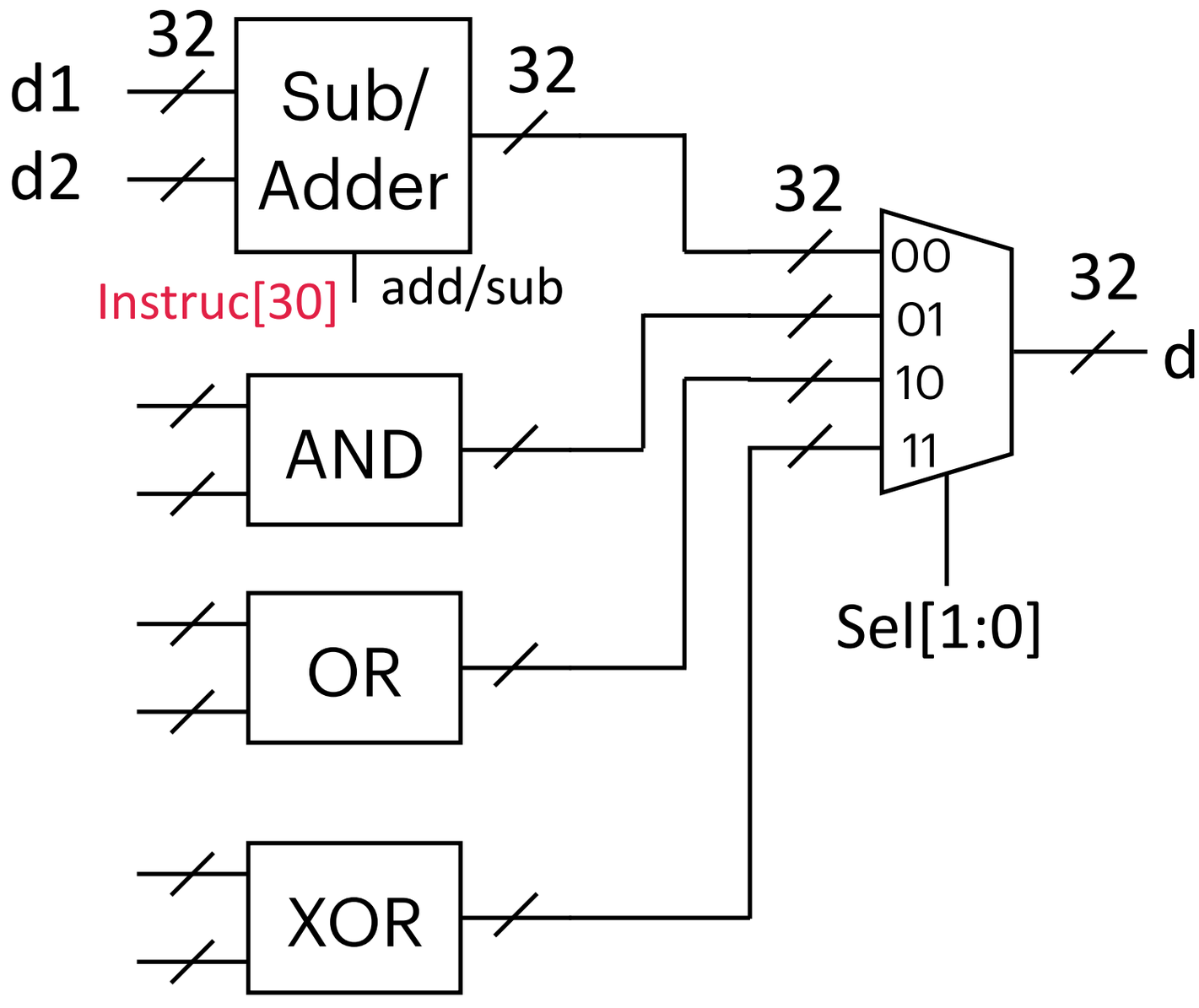




Ripple-carry adder/domino adder

ALU Design

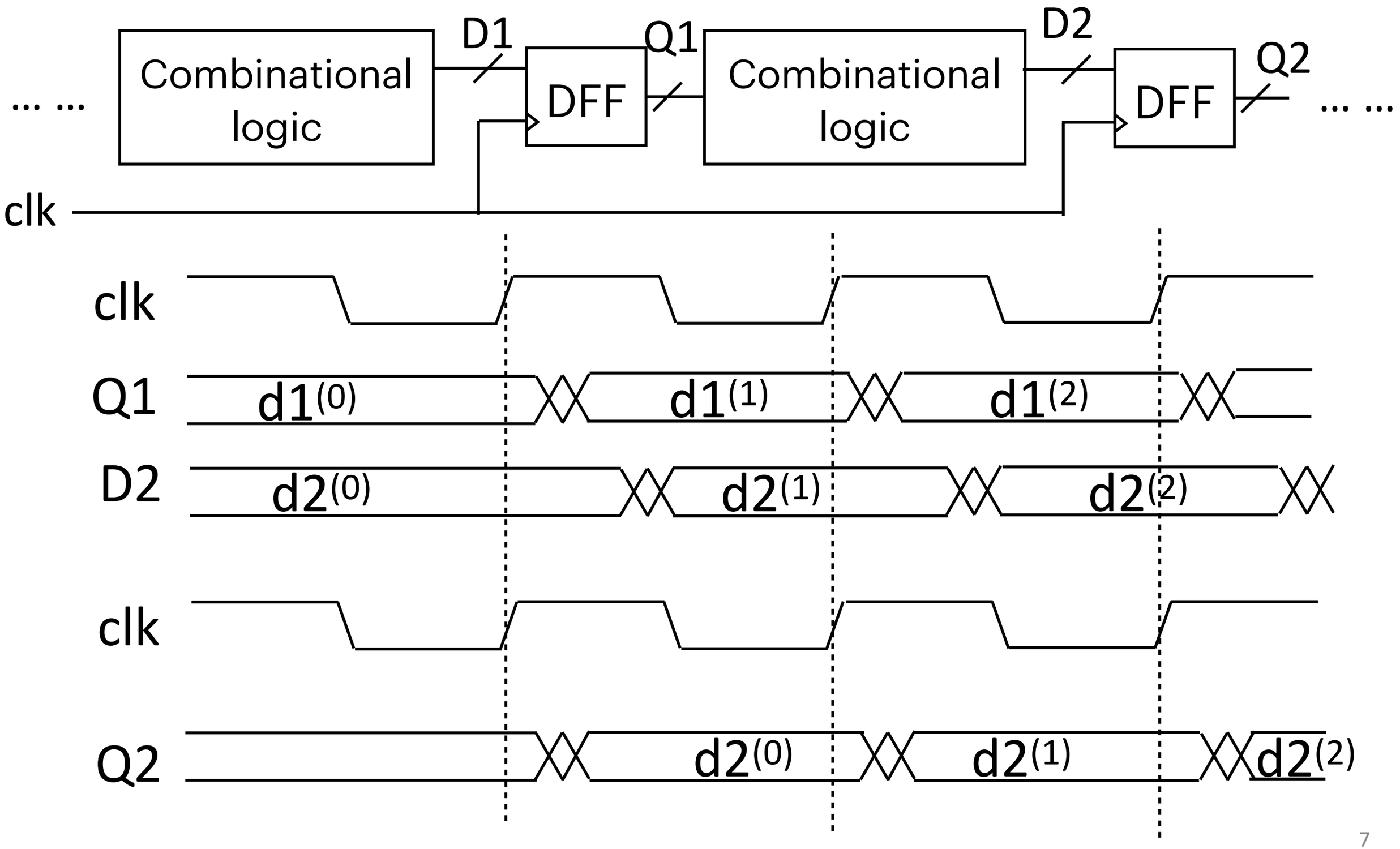
funct3		sel
000	ADD(I)	00
000	SUB	00
100	XOR(I)	11
110	OR(I)	10
111	AND(I)	01



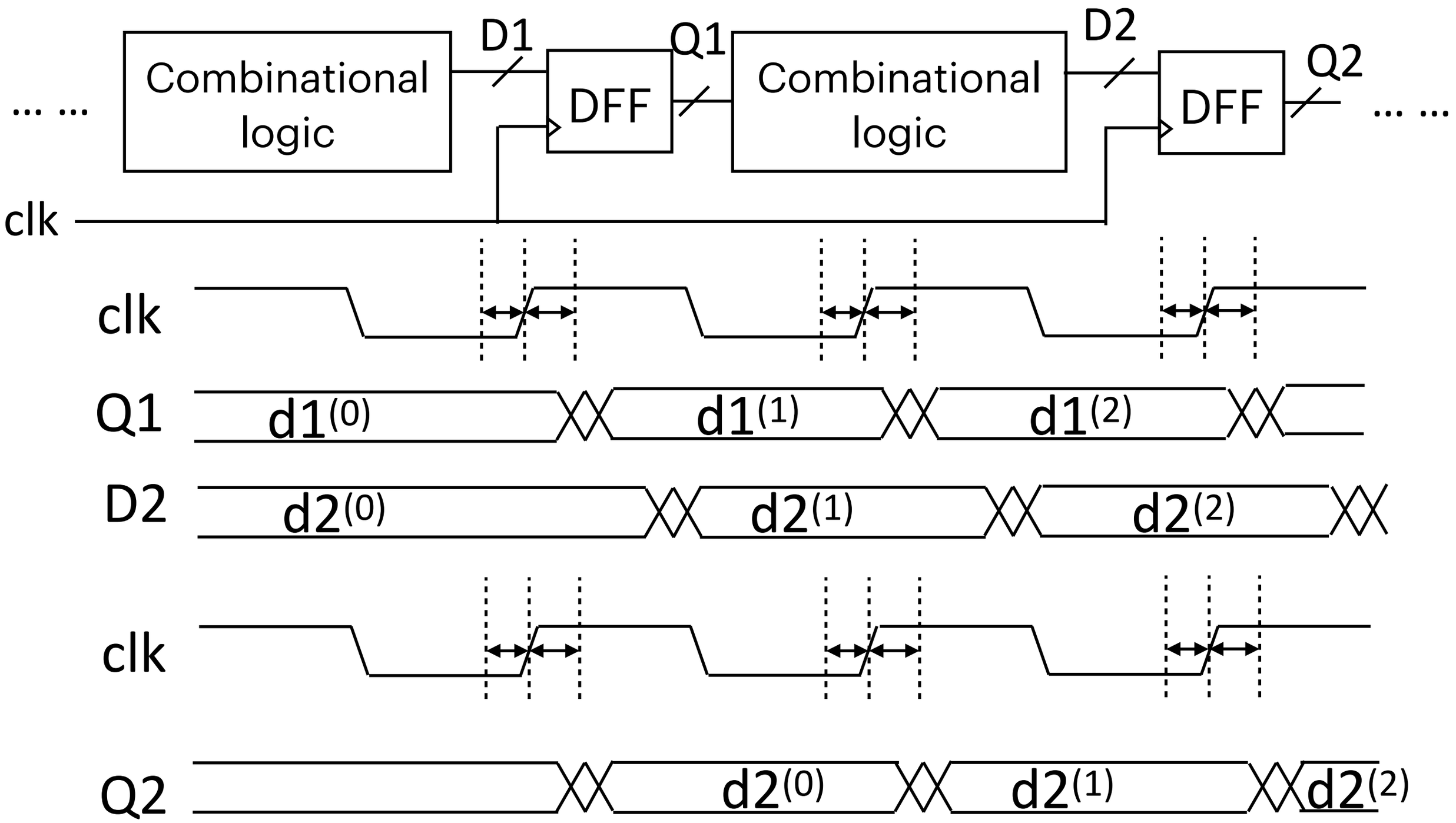
Recap: Basic Sequential Elements

- DFFs: edge-triggered, i.e., the output/stored state only changes when the clk is at transition edge (1->0 negedge or 0->1 posedge) , can store/output/write values;
- D-latch: level-triggered
- Synchronized circuits: use the same clock; DFFs/registers update at the same time, and capture the previous state/ results (at the clock edge)
- Modern digital systems consists of both combinational and sequential circuits
- Timing issues: setup/hold time
- Contamination vs. propagation: signal begin to change; signal finish changing

Full Picture



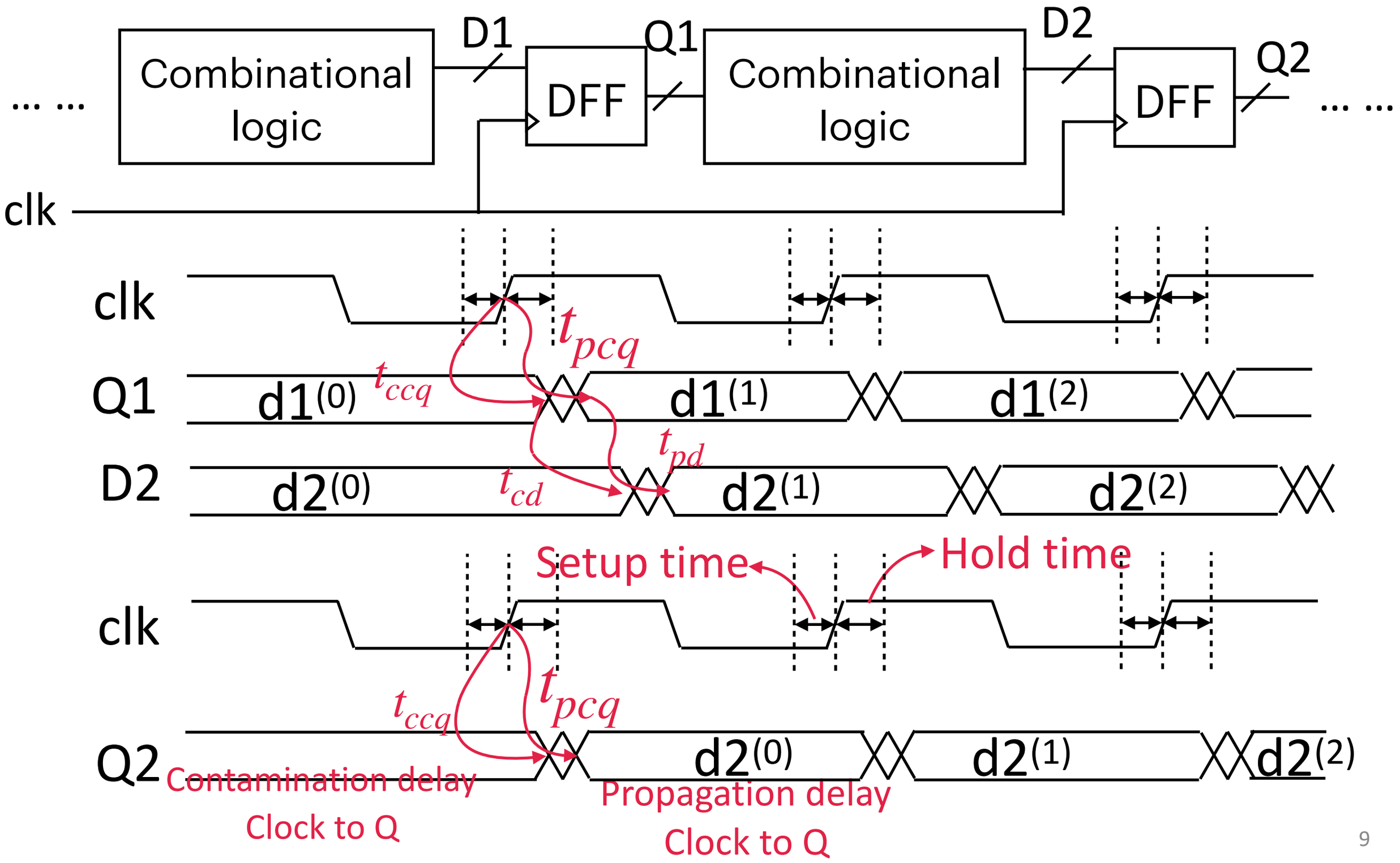
Full Picture



Full Picture

t_{cd}: comb. Logic contamination delay

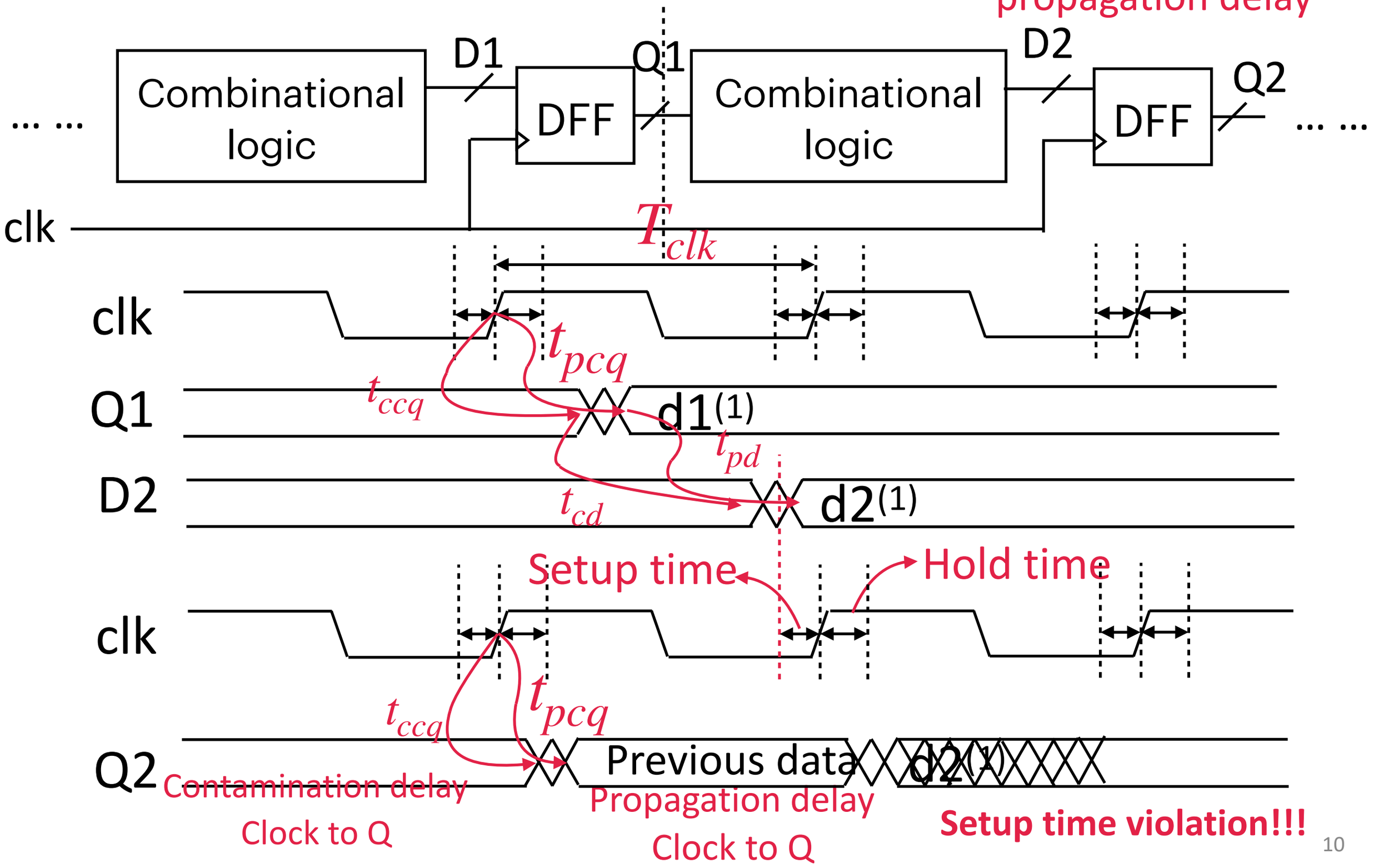
t_{pd}: comb. Logic propagation delay



Full Picture

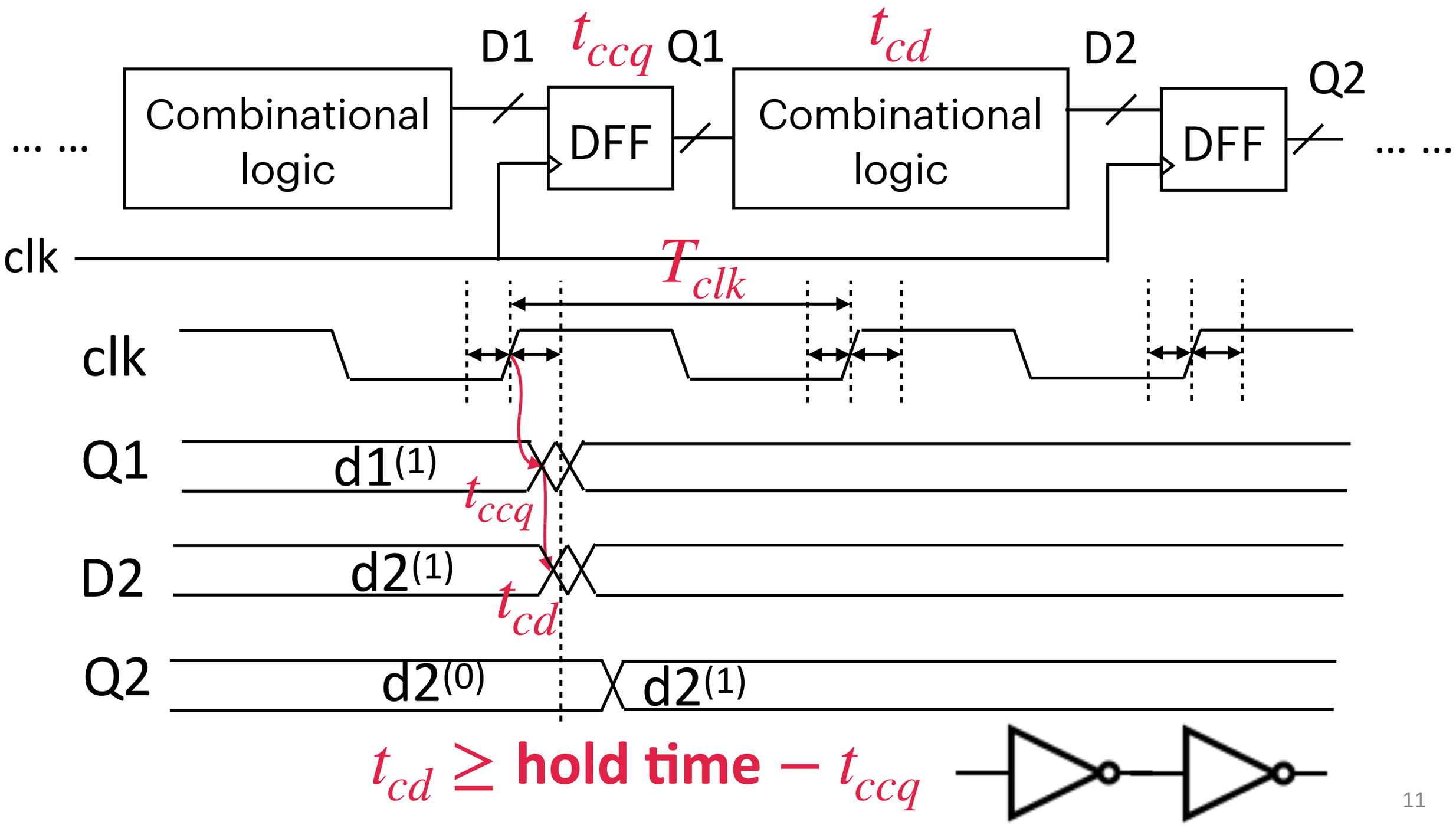
t_{cd}: comb. Logic contamination delay

t_{pd}: comb. Logic propagation delay



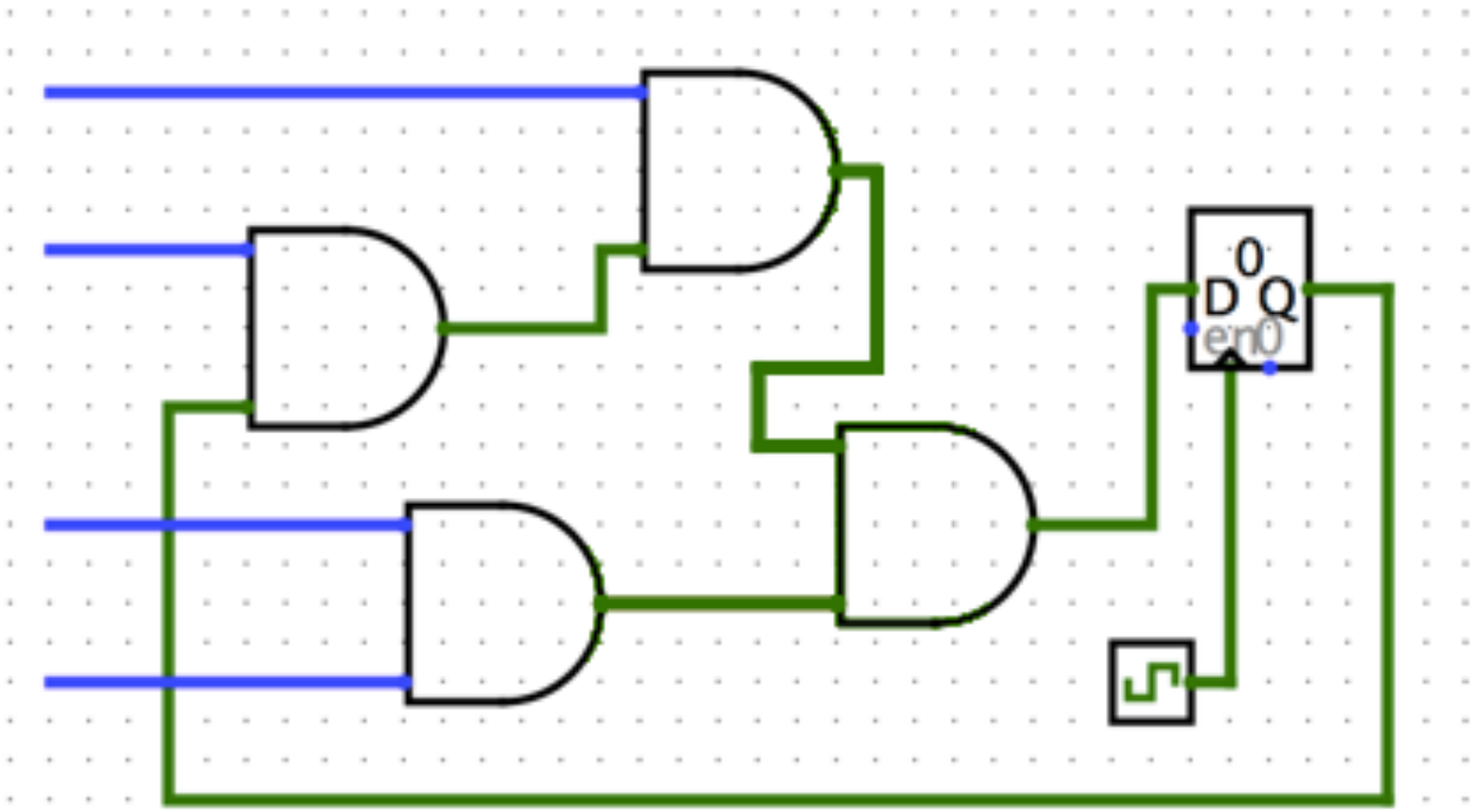
Min-Delay Constraints

- Avoid hold time violation



$t_{cd} \geq$ **hold time** $- t_{ccq}$

Question



Clock->Q (P) 1ns
Setup 1ns
Hold 1ns
AND gate delay 1ns

What is maximum clock frequency?

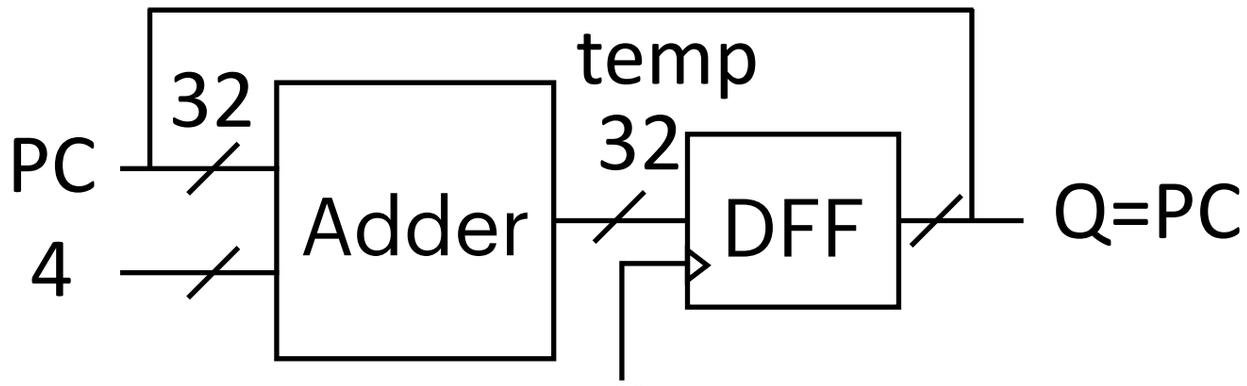
- A: 5 GHz
- B: 500 MHz
- C: 200 MHz
- D: 250 MHz
- E: 1/6 GHz

General SDS System Design

- Take too long to complete if designs are large
- Usually uses Electronic Design Automation (EDA) software to assist instead of designing manually
  - Hardware description language (Verilog HDL, VHDL, Chisel, etc.) for netlist synthesis, timing analysis, etc. with EDA tools
- Use synchronous (i.e., clocked) hardware wherever possible
- Top-down design methods, IP reuse
- To design a CPU and understand how it works, we choose to design it manually; Method as follows

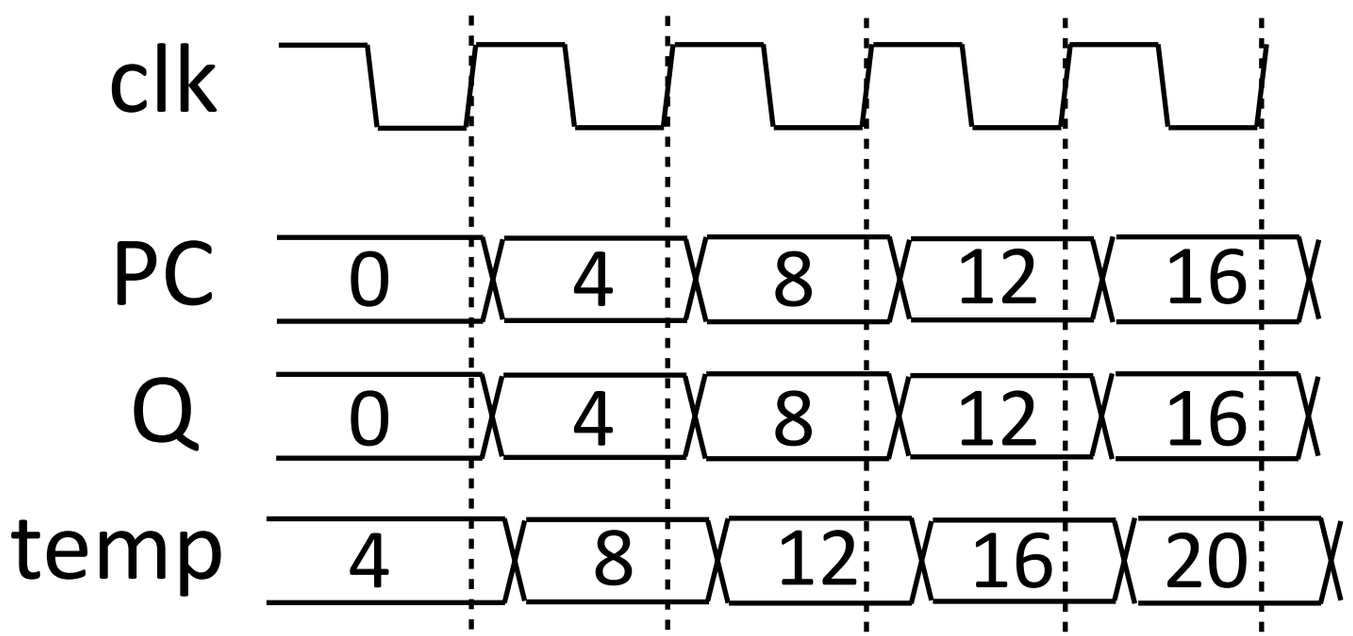
Tiny SDS Modeled as FSM: Example

- PC counter: PC = PC + 4 (w/o considering branch/jump)



Finite state machine model

Next-state logic clk



Current state

Output

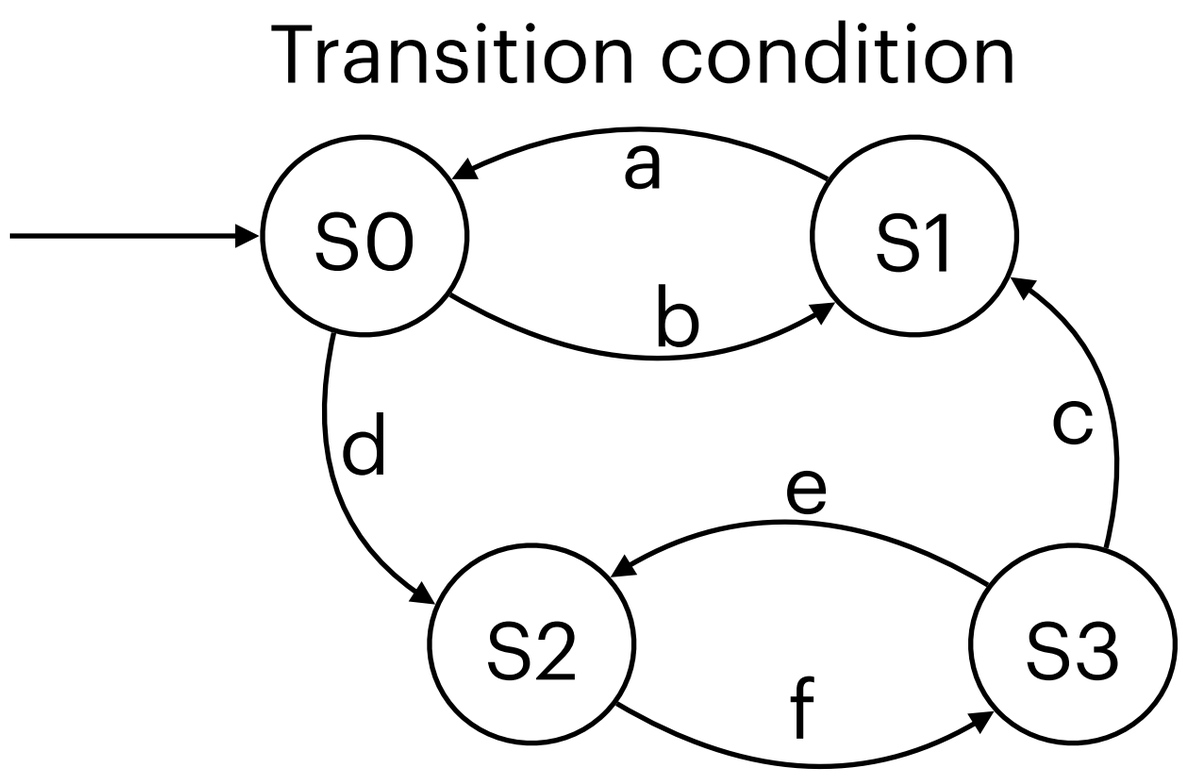
Next state

Finite-State Machine (FSM)

“Give me the place to stand, and I shall move the earth.”—Archimedes

FSM

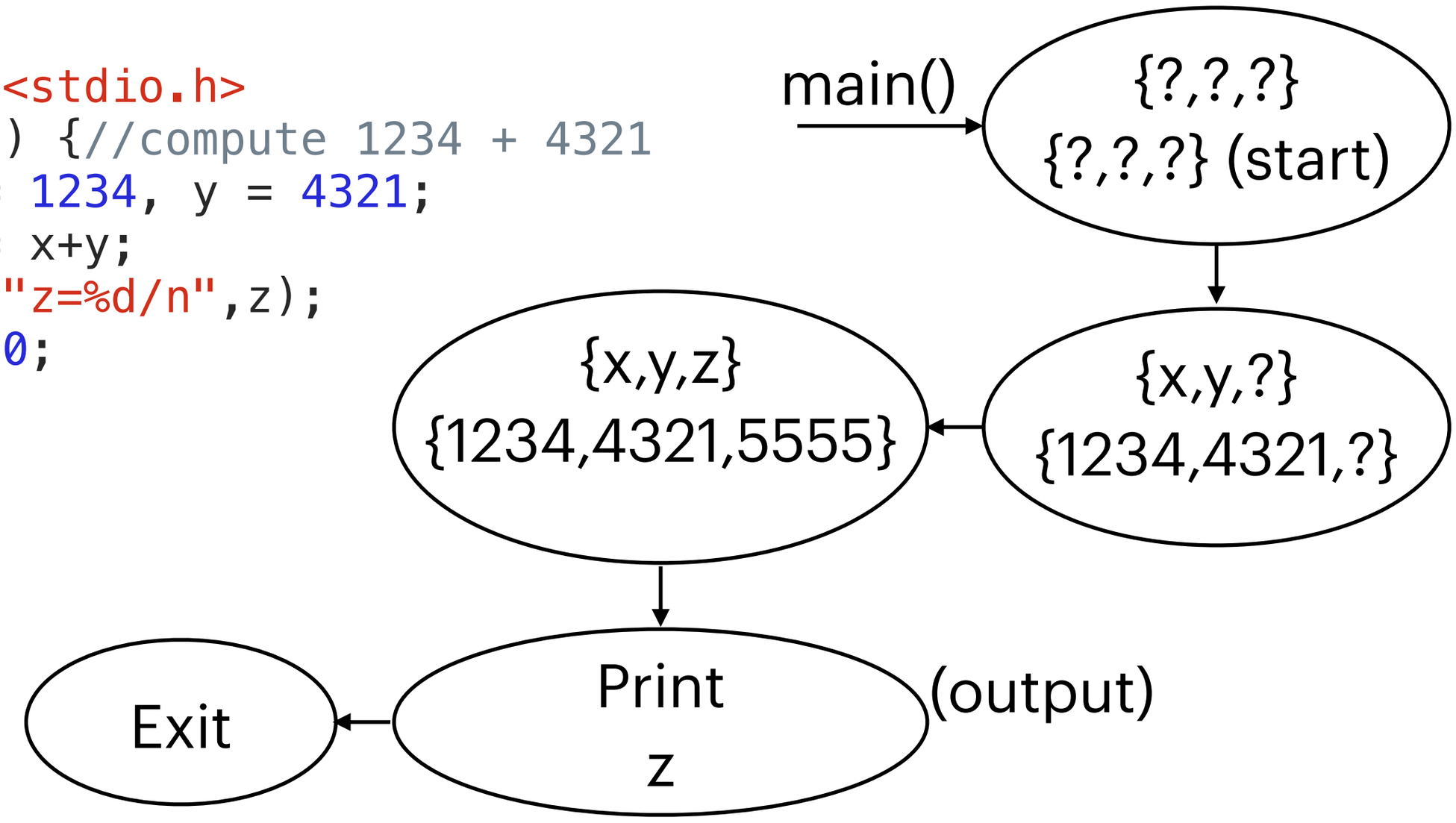
- FSMs consists of states, transitions, an entrance (initial state) and input/output (optional)



C Program as an FSM

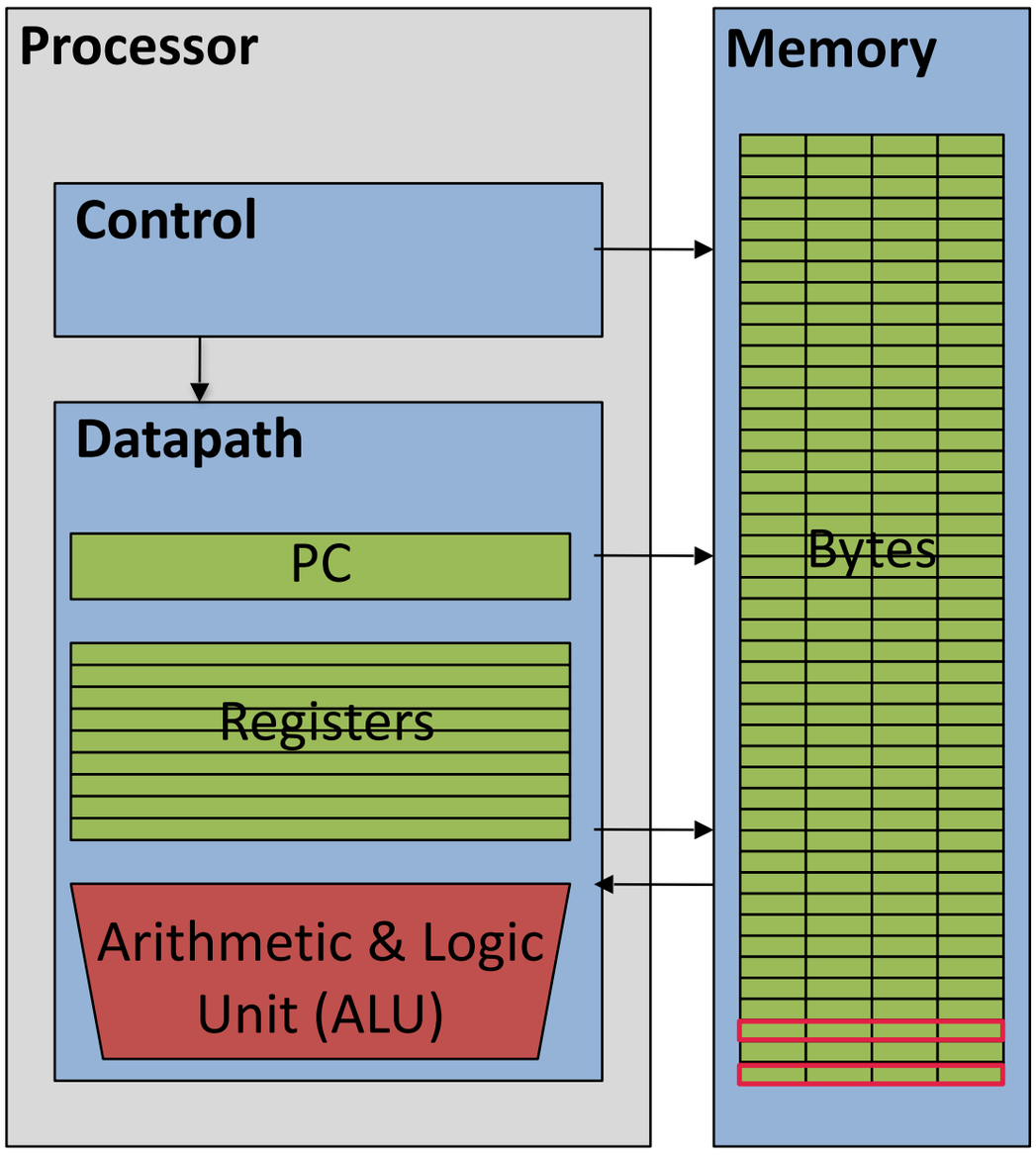
- FSMs consists of states, transitions, an entrance (initial state) and input/output (optional)

```c
#include <stdio.h>
int main() {//compute 1234 + 4321
   int x = 1234, y = 4321;
   int z = x+y;
   printf("z=%d/n",z);
   return 0;
}
```



ISA as an FSM

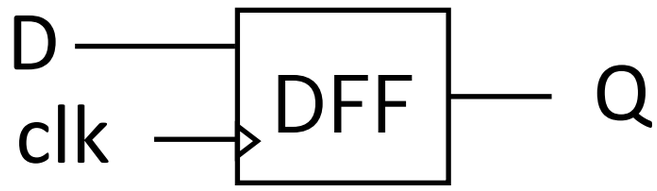
- FSMs consists of states, transitions, an entrance (initial state) and input/output (optional)

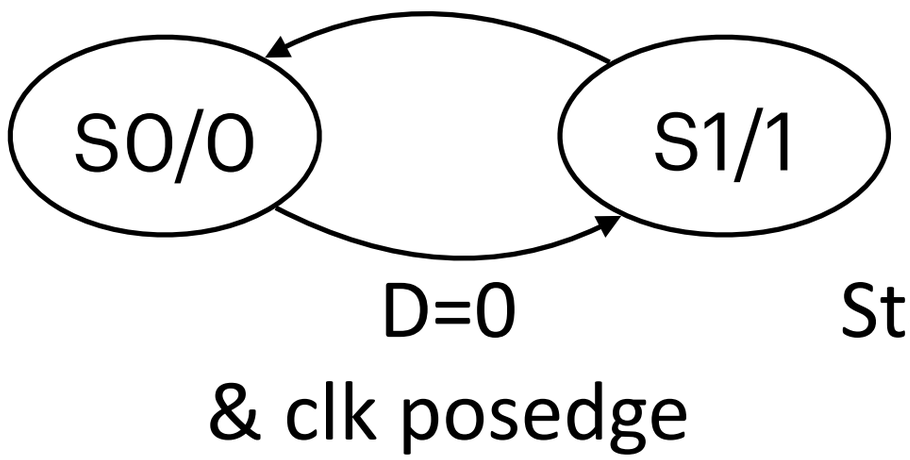


- States: all registers & memory
- Transition: register/memory value change
- Entrance: power on
- Input: instructions, can change registers/memory value
- Output: states of each registers/ memory

Digital Systems as FSMs

- FSMs consists of states, transitions, an entrance (initial state) and input/output (optional)
- Given an FSM model, we can use digital circuits to implement it

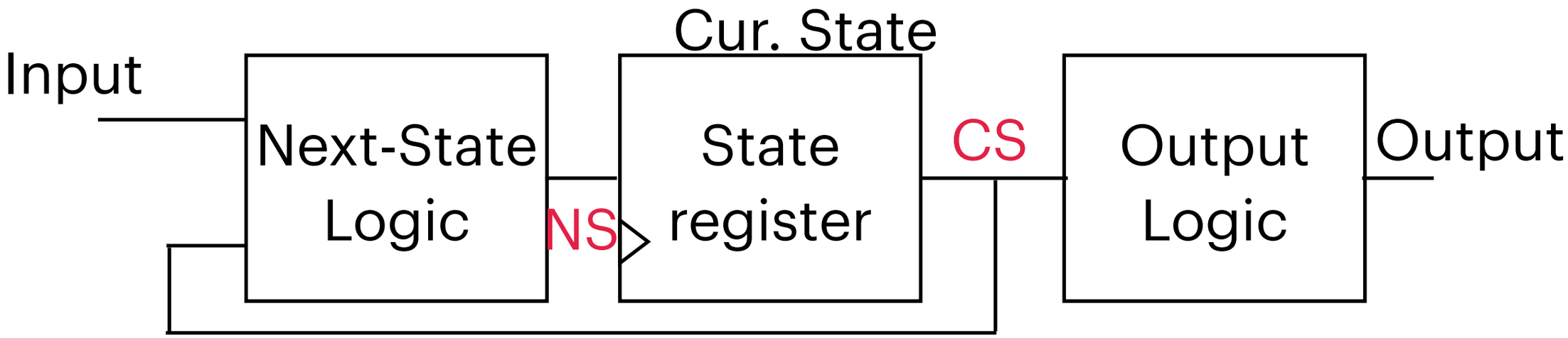




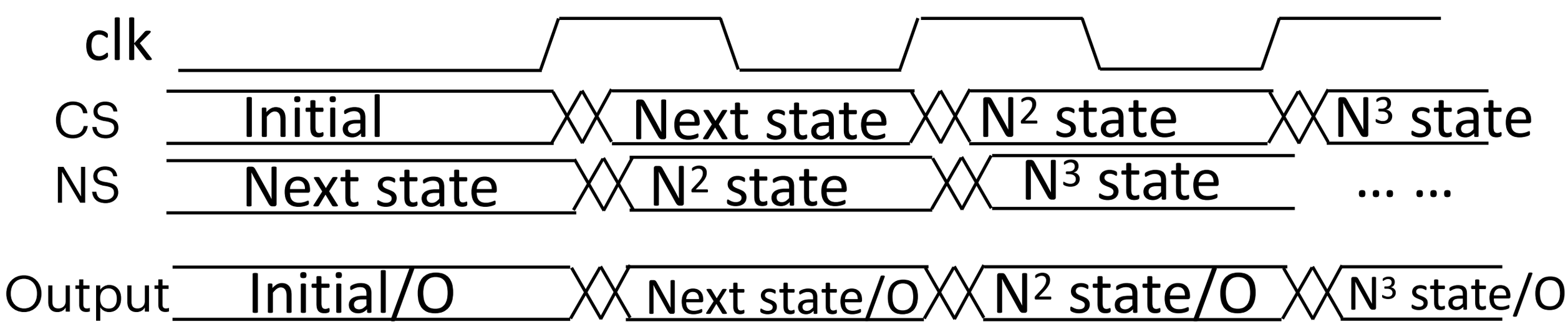
State transition diagram

FSM Digital Implementation

- A digital circuitry (TaoLu) for FSM (Moore machine)

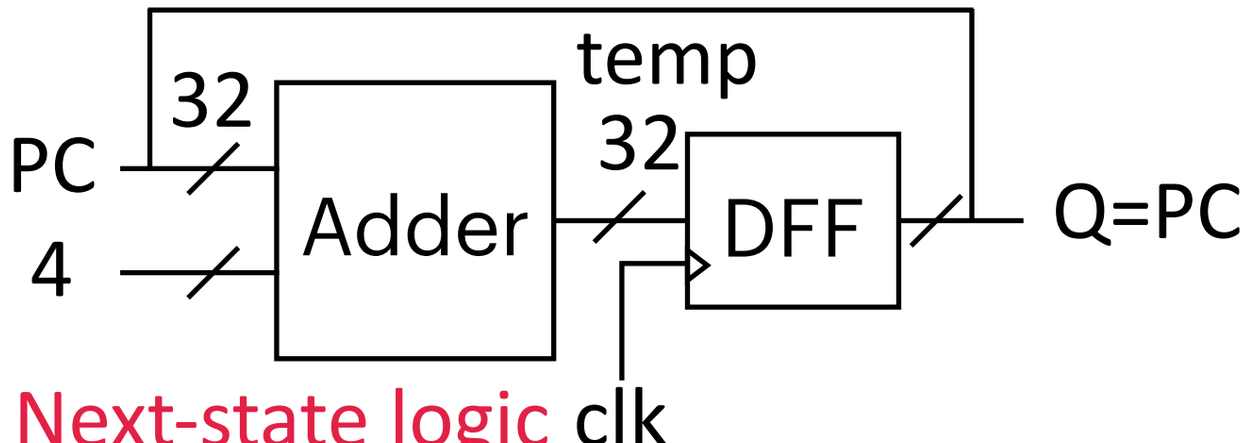


- Timing diagram



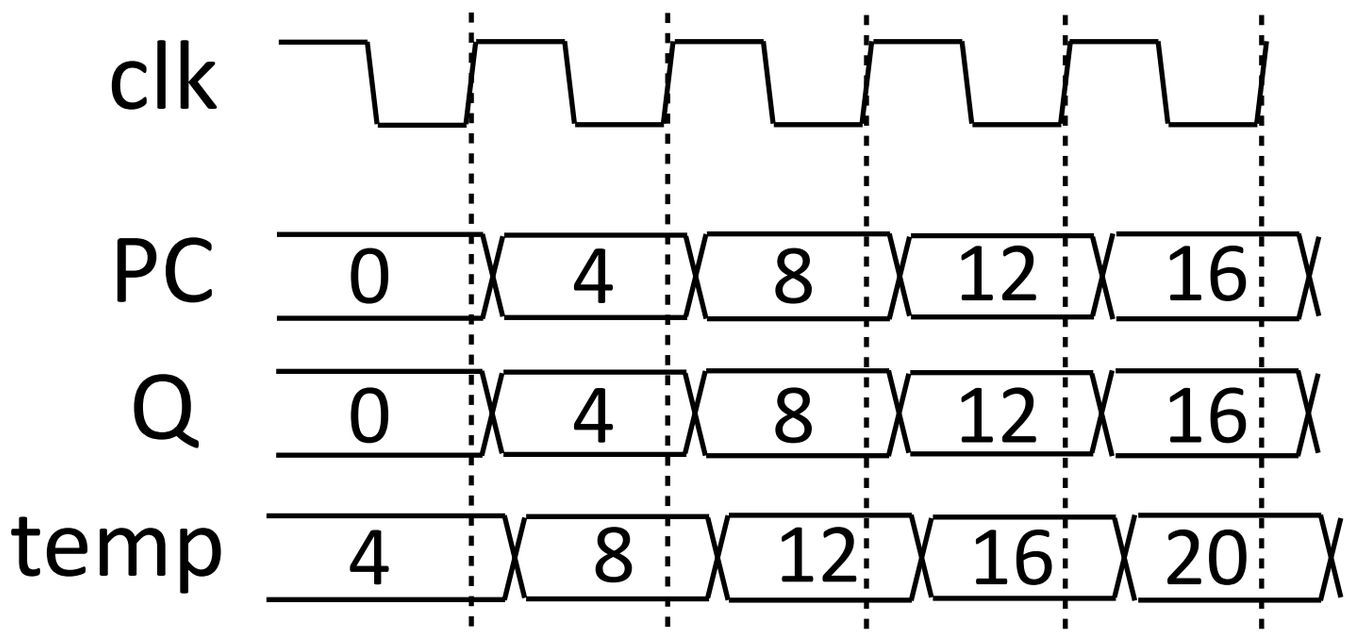
Tiny SDS Modeled as FSM: Example

- PC counter: PC = PC + 4 (w/o considering branch/jump)



Finite state machine model

Next-state logic clk



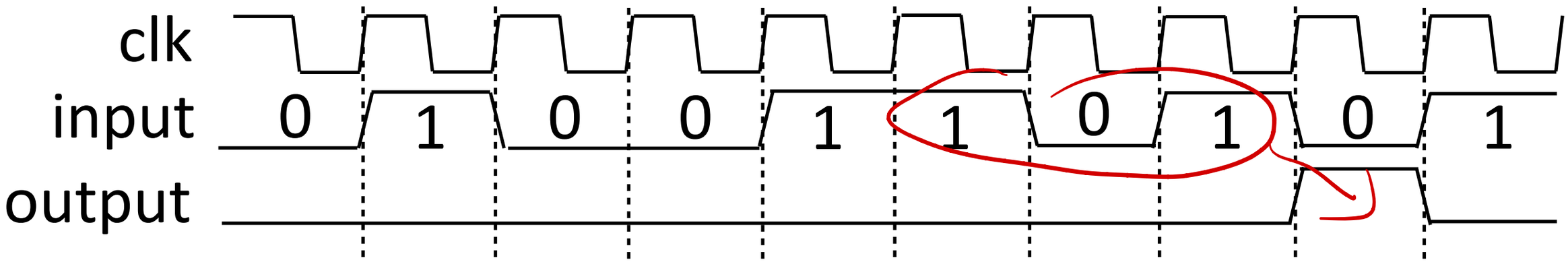
Current state

Output

Next state

A Classic Problem: Build Digital Circuit for Sequence Detection

- Build a digital circuit, detecting the occurrence of {101} in the input 0/1 sequence (non-overlapping)



input: 0 or 1 in a sequence, one bit at a clock cycle

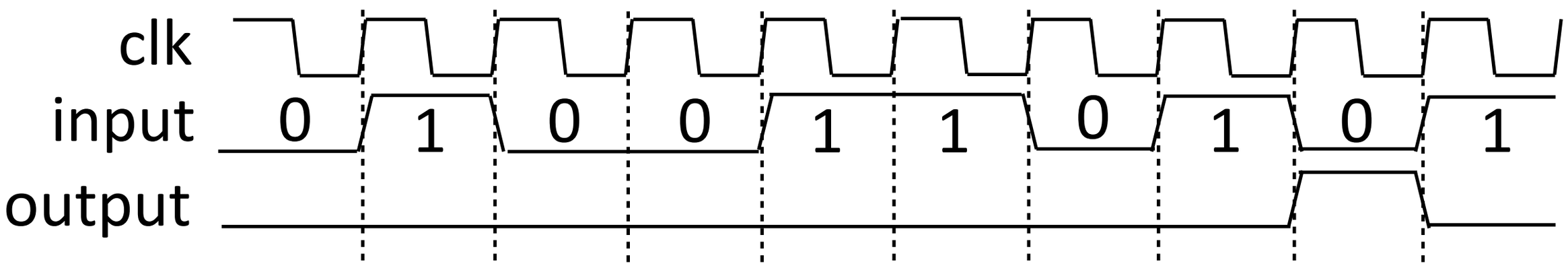
Output: 1 after {101} detected, otherwise 0;

States: ?

Transition: ?

A Classic Problem: Build Digital Circuit for Sequence Detection

- Build a digital circuit, detecting the occurrence of {101} in the input 0/1 sequence (non-overlapping)



input: 0 or 1 in a sequence, one bit at a clock cycle

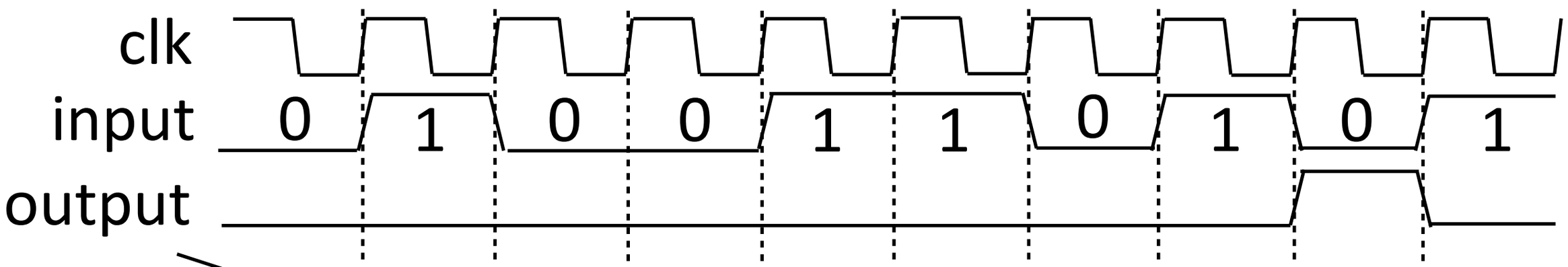
Output: 1 after {101} detected, otherwise 0;

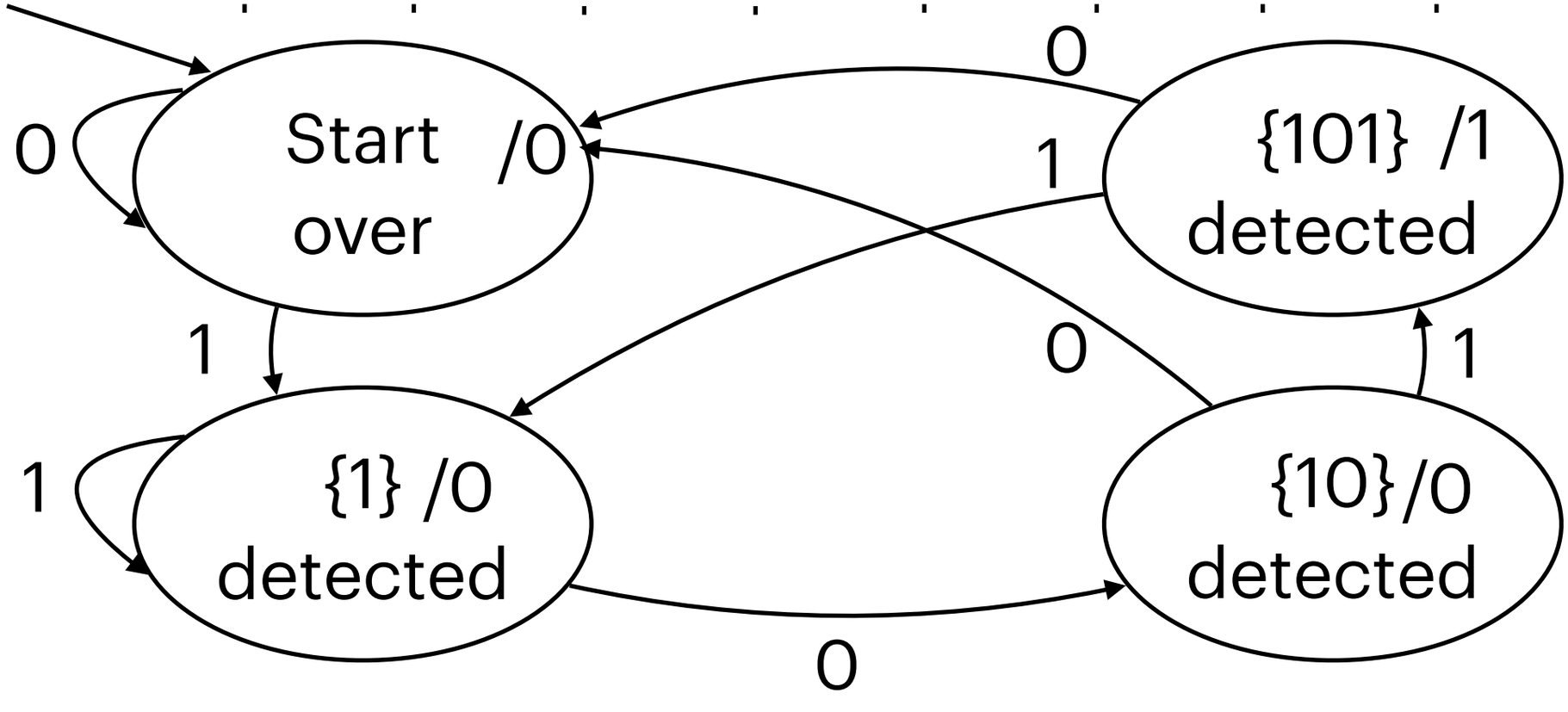
States:

Transition:

A Classic Problem: Build Digital Circuit for Sequence Detection

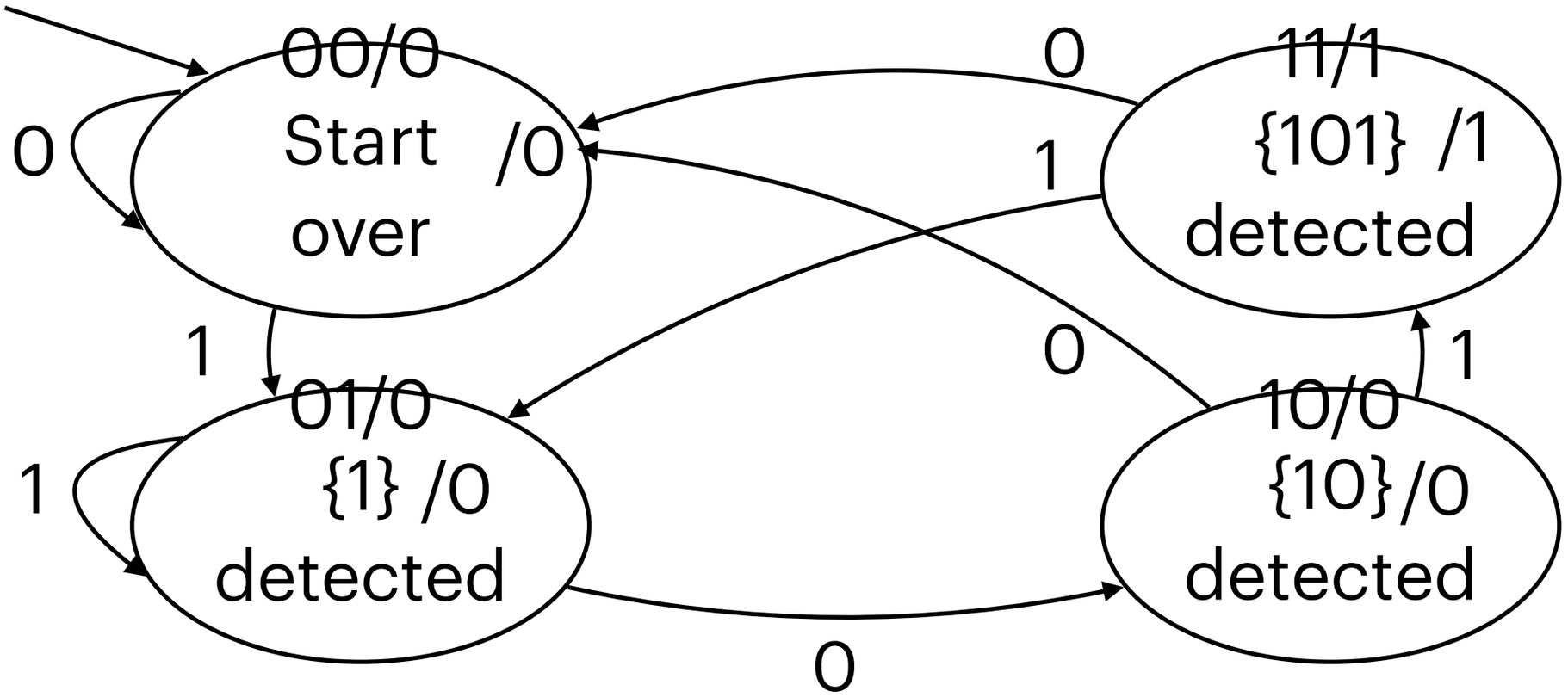
- Build a digital circuit, detecting the occurrence of {101} in the input 0/1 sequence (non-overlapping)





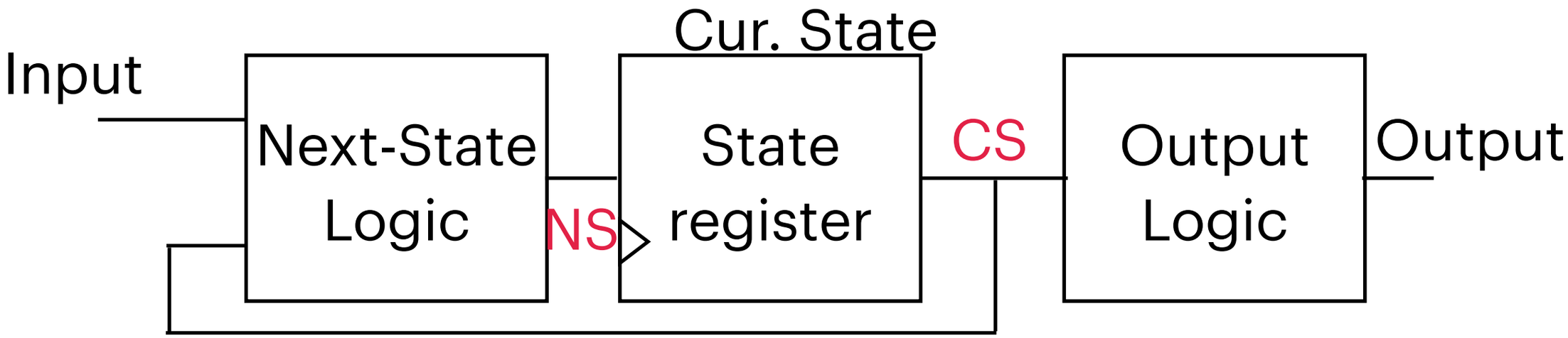
A Classic Problem: Build Digital Circuit for Sequence Detection

- Everything is a number. Use binary numbers to encode states: {00, 01, 10, 11}

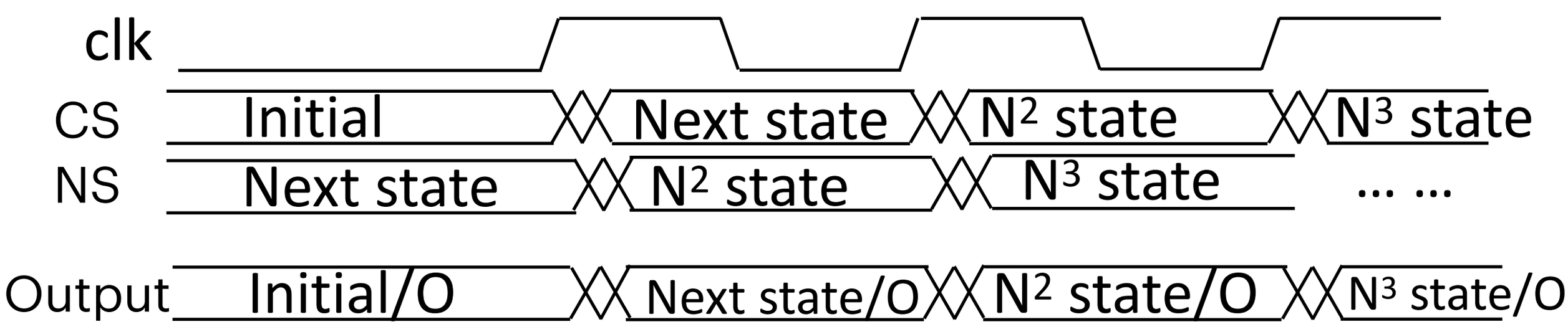


A Classic Problem: Build Digital Circuit for Sequence Detection

- A digital circuit model (TaoLu) for FSM (Moore machine)

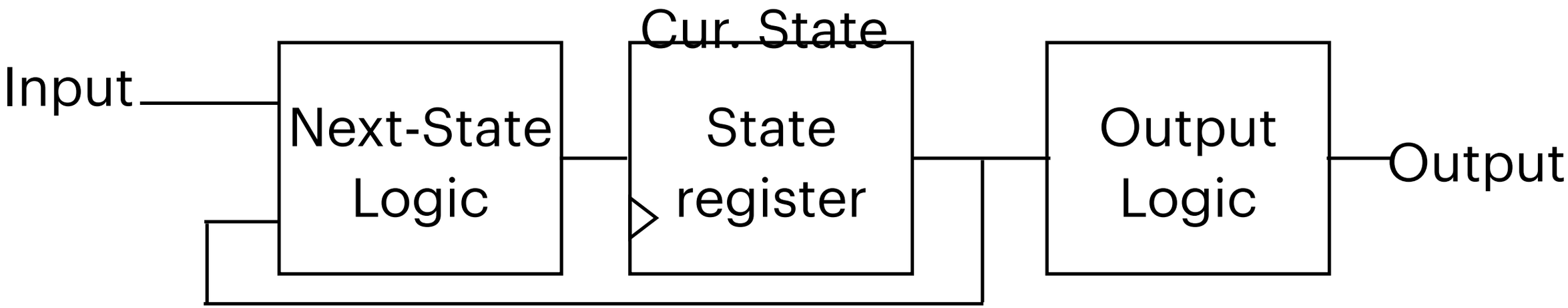


- Timing diagram



A Classic Problem: Build Digital Circuit for Sequence Detection

- A digital circuit model for FSM (Moore machine)

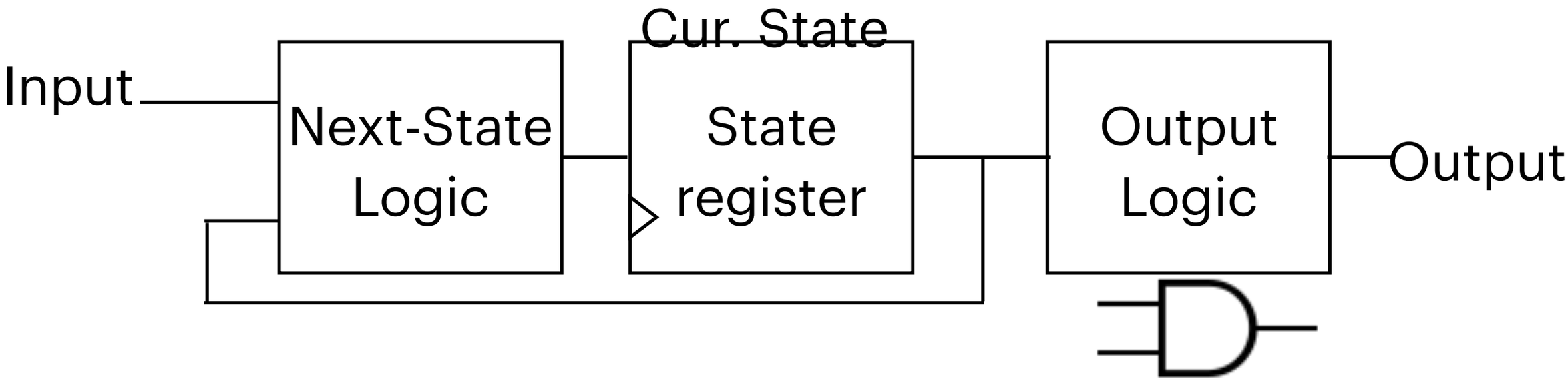


- Truth table

Cur.State	Input	Next State	Output
00	0	00	0
00	1	01	0
01	0	10	0
01	1	01	0
10	0	00	0
10	1	11	0
11	0	00	1
11	1	01	1

A Classic Problem: Build Digital Circuit for Sequence Detection

- A digital circuit model for FSM (Moore machine)

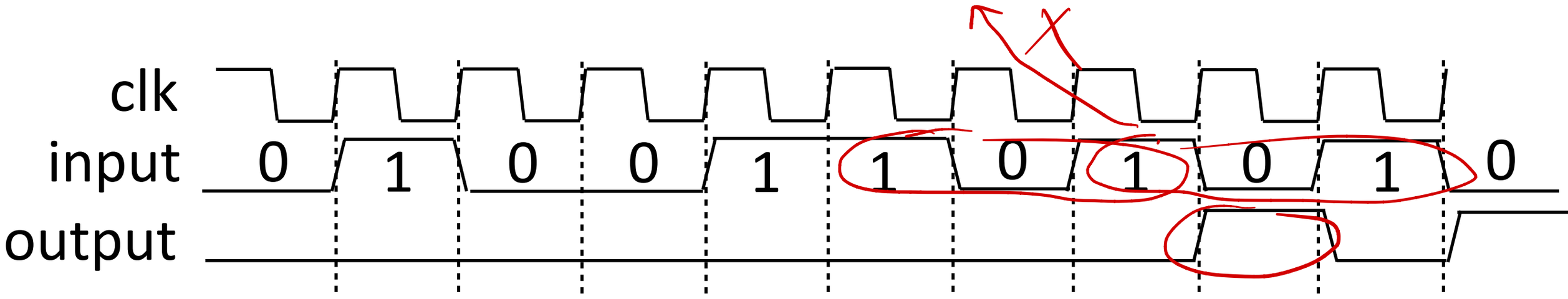


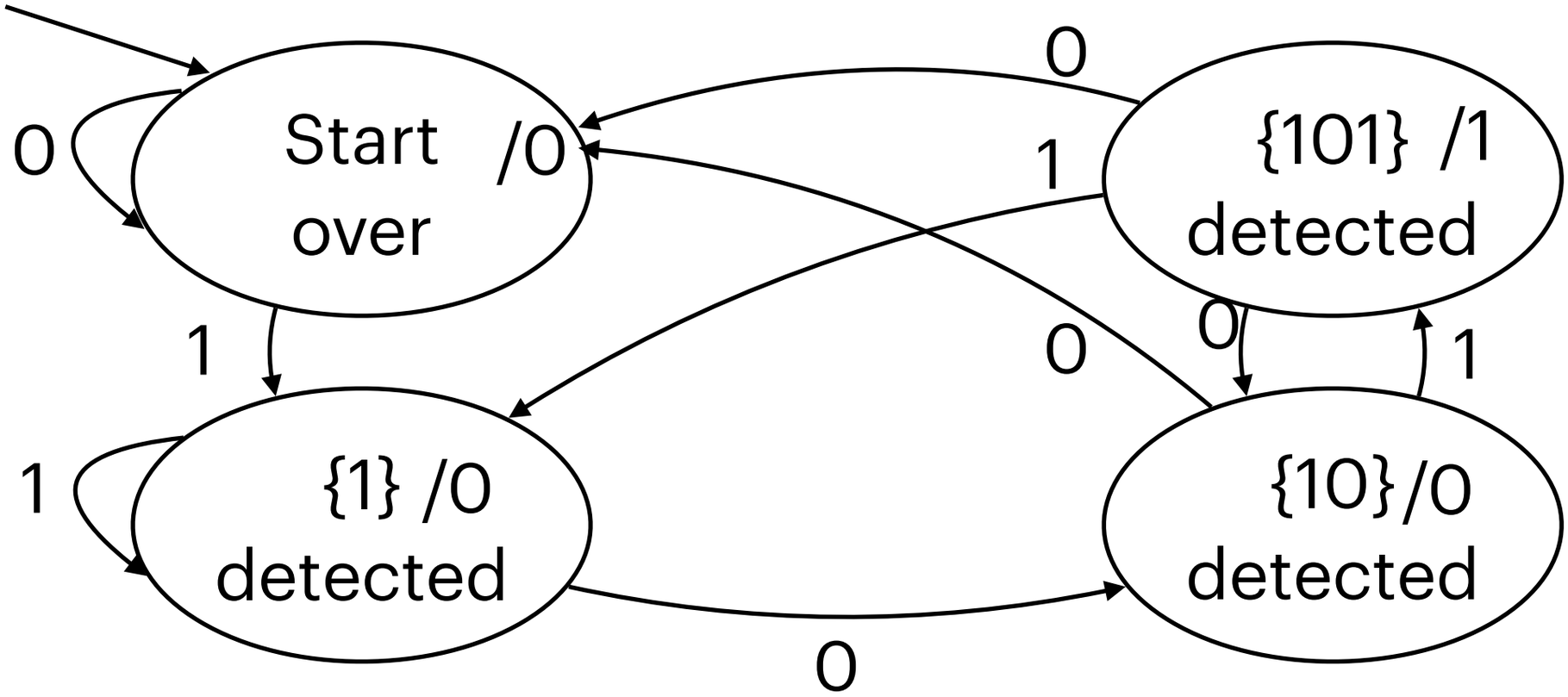
- Truth table

Cur.State	Input	Next State	Output
00	0	00	0
00	1	01	0
01	0	10	0
01	1	01	0
10	0	00	0
10	1	11	0
11	0	00	1
11	1	01	1

What about Overlapping?

- Build a digital circuit, detecting the occurrence of {101} in the input 0/1 sequence (non-overlapping)

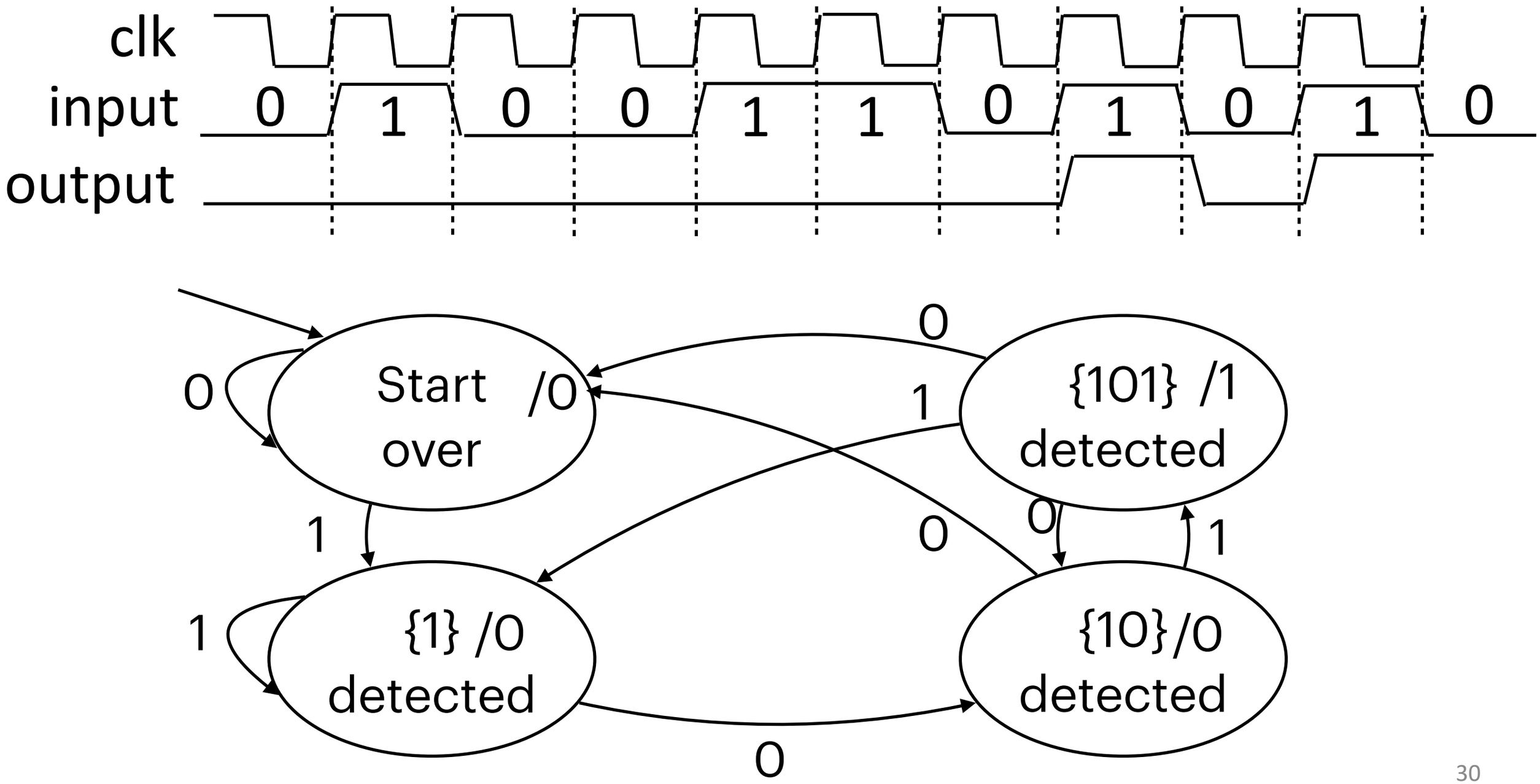




Mealy Machine

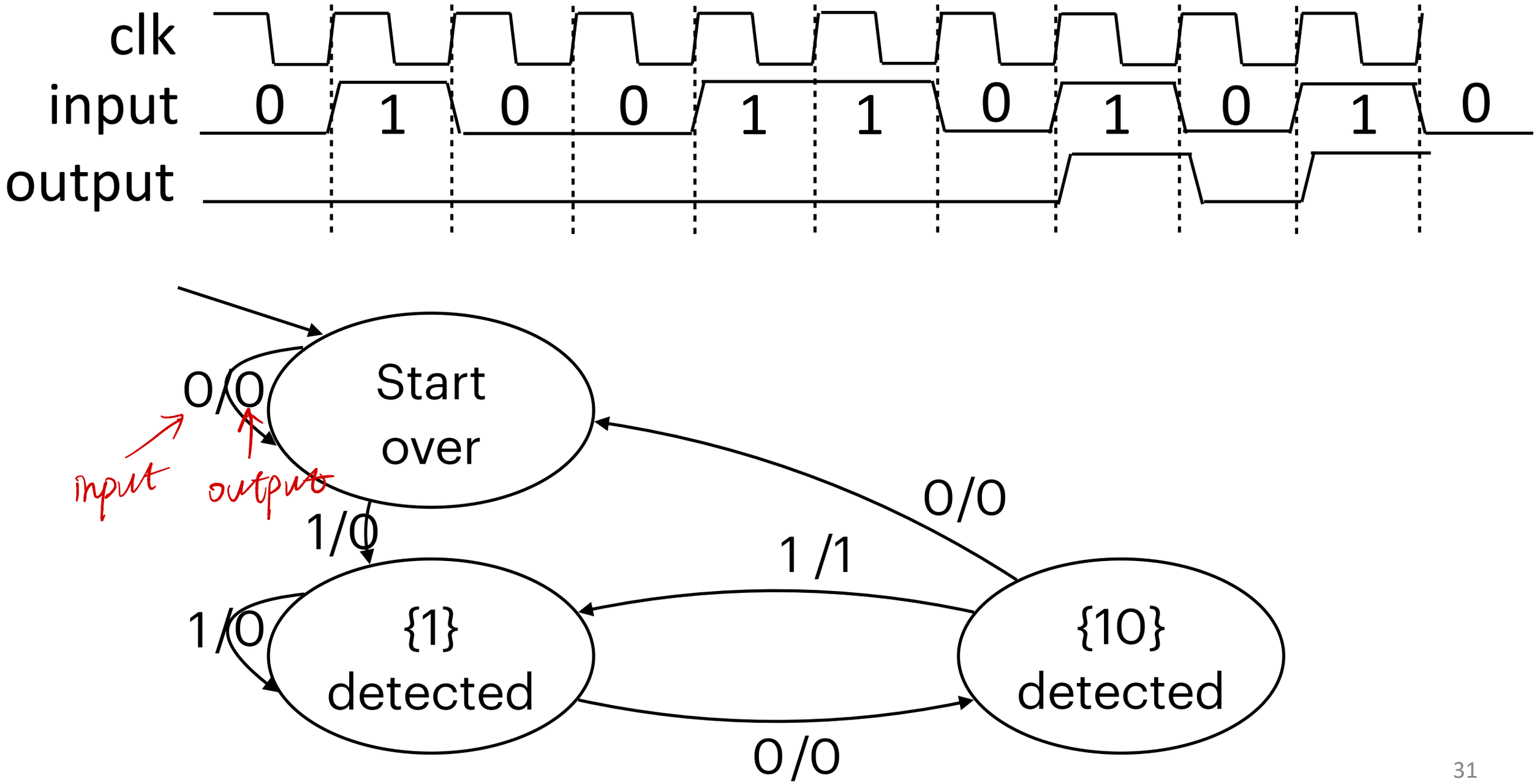
output 与 input, current state 相关

- Another model/method to build FSM circuit



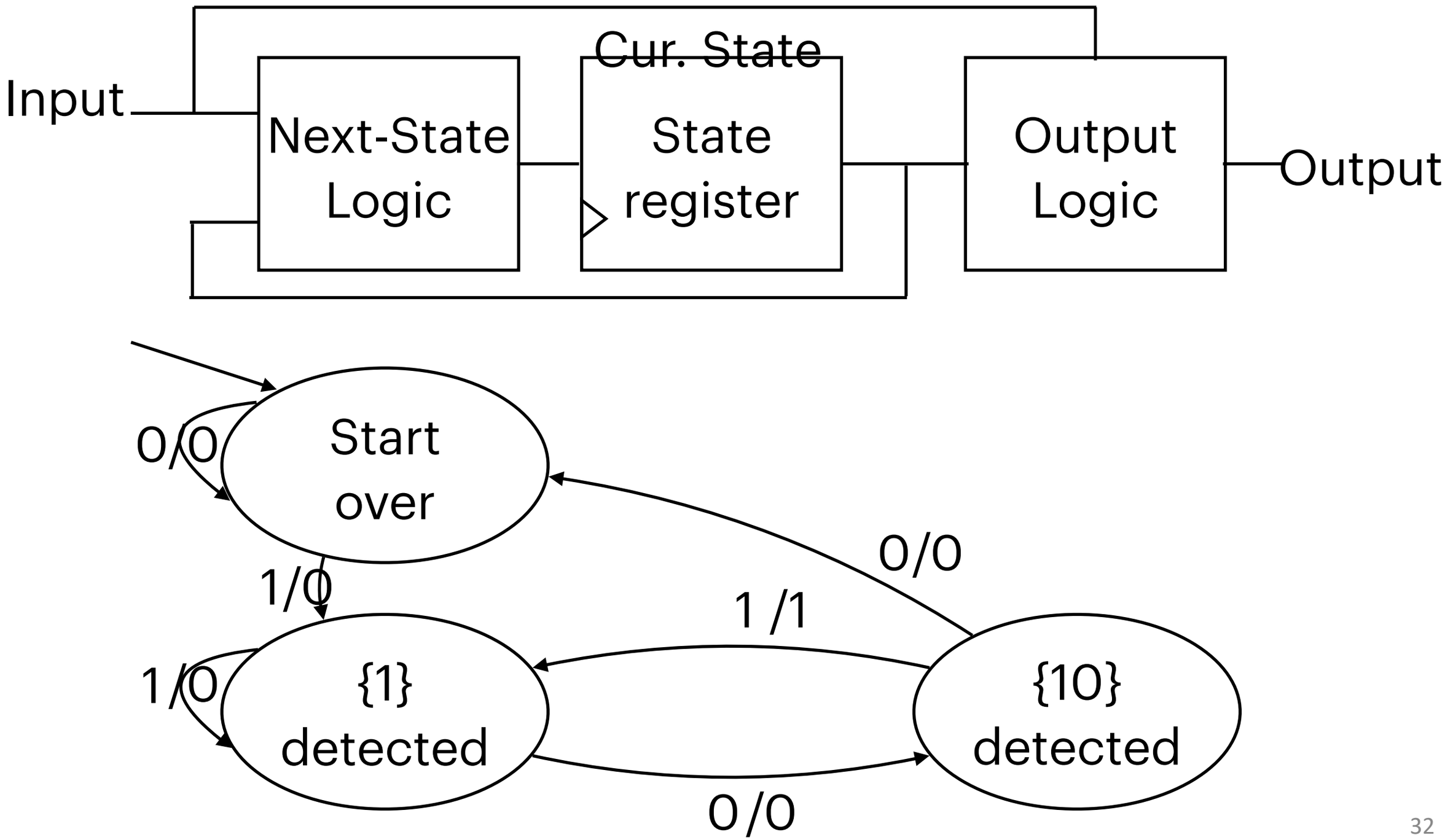
Mealy Machine

- Another model/method to build FSM circuit



Mealy Machine

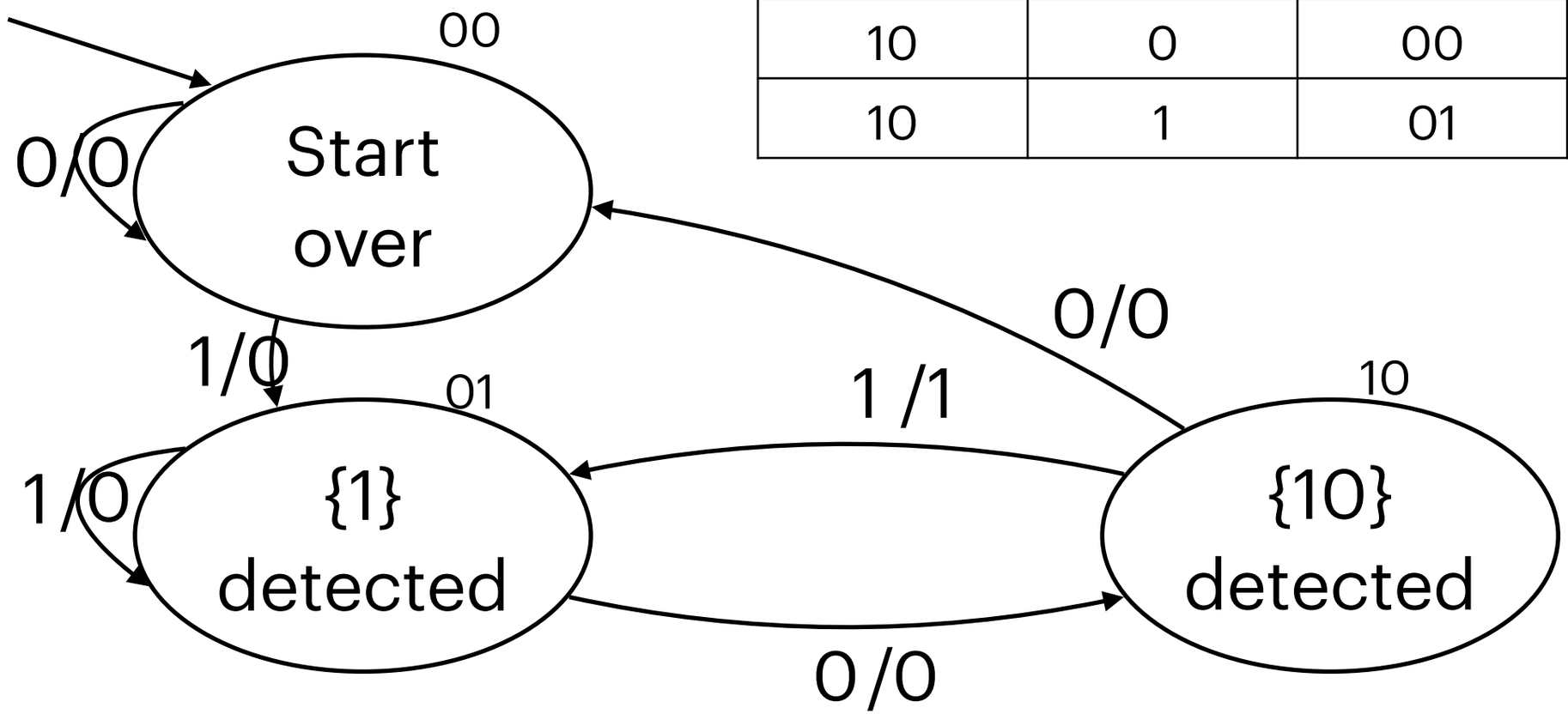
- Another model/method to build FSM circuit



Mealy Machine

- Another model/method to build FSM circuit
- +Less states, potentially less registers
- -Output not synchronized
- Exchangeable with Moore

Cur.State	Input	Next State	Output
00	0	00	0
00	1	01	0
01	0	10	0
01	1	01	0
10	0	00	0
10	1	01	1



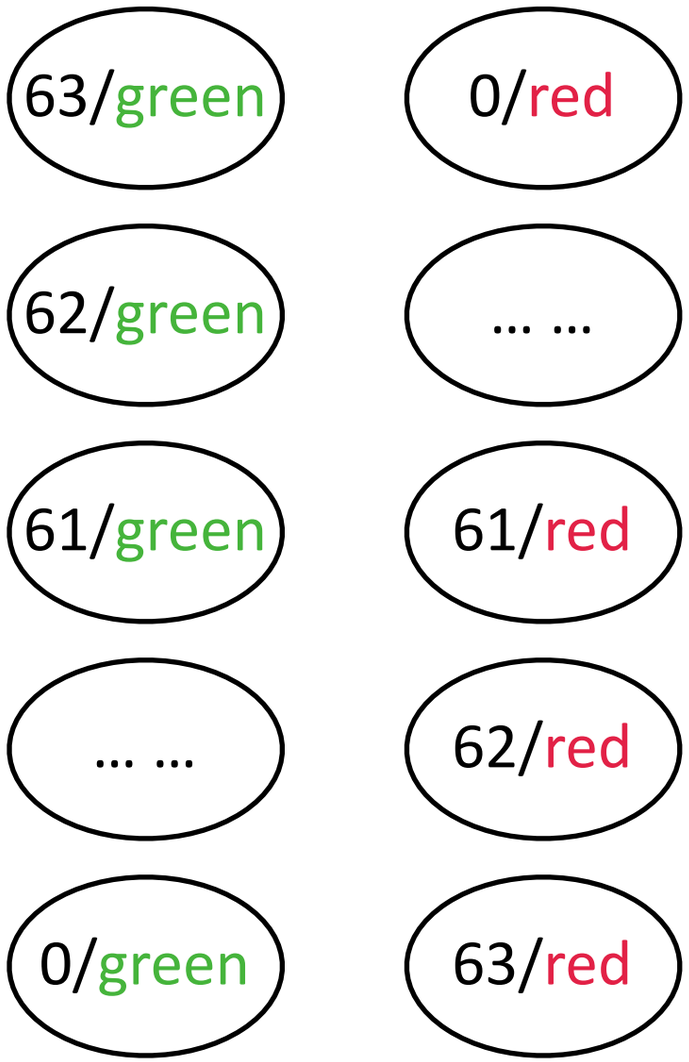
FSM Circuit Design (Manually)

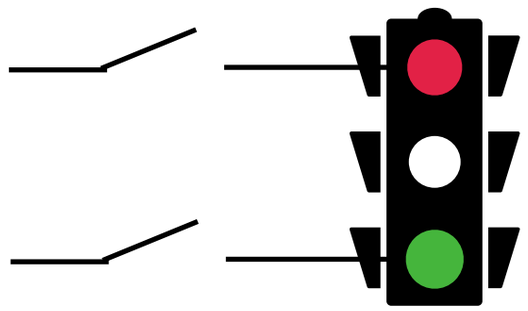
- Draw the state transition diagram
- Assign different binary numbers to the states
- Write down the truth table
  - Next-state logic (input, current state)
  - Output logic (current state for Moore machine, current state and input for Mealy machine)
- Generate next-state & output logic circuits and fit in the FSM model

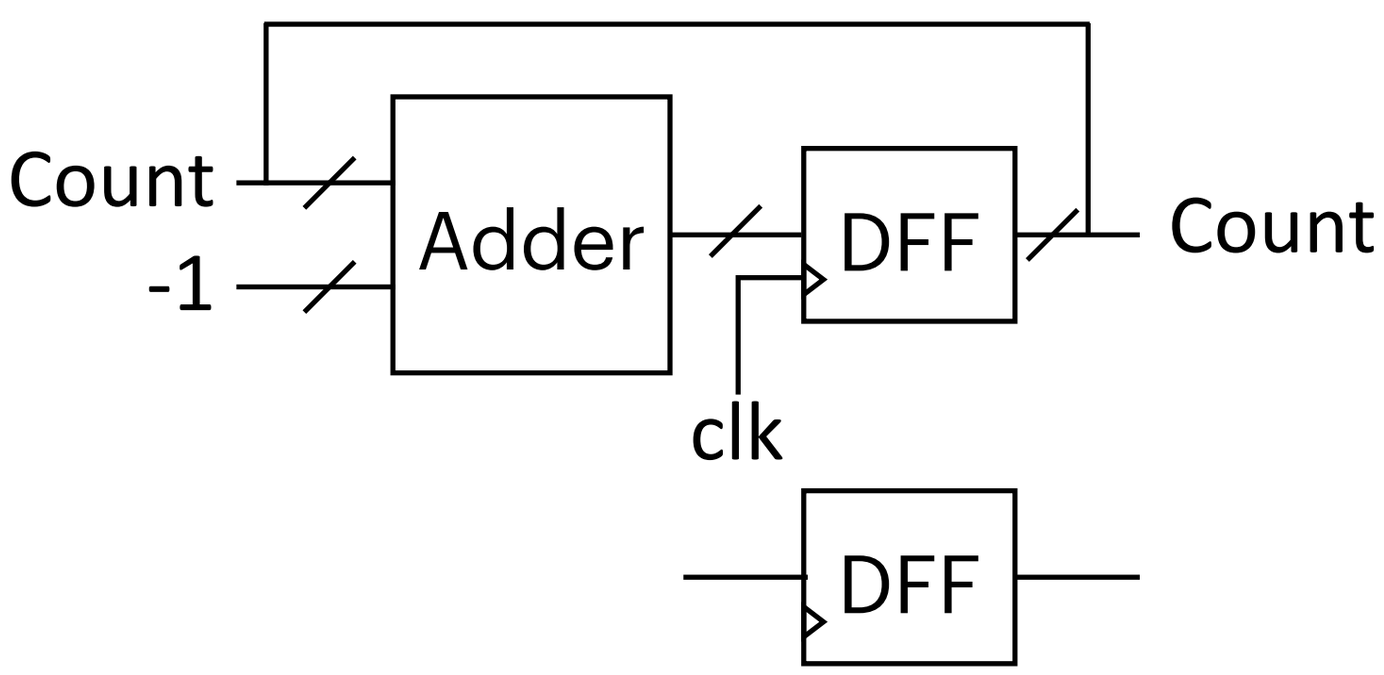
Back to CPU Design

Tiny SDS Example II

- Traffic light: green & red (count down from 63 and change)

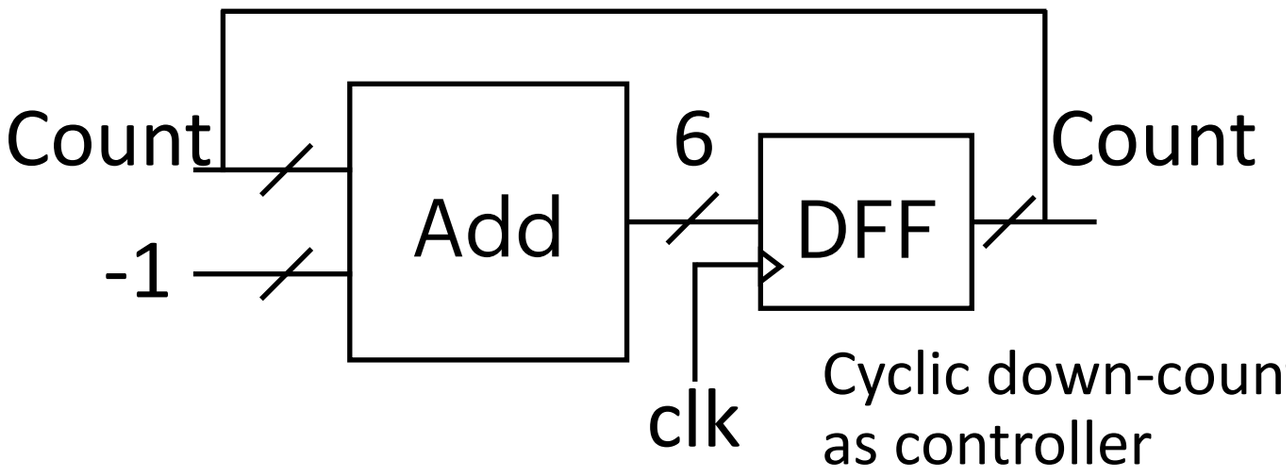




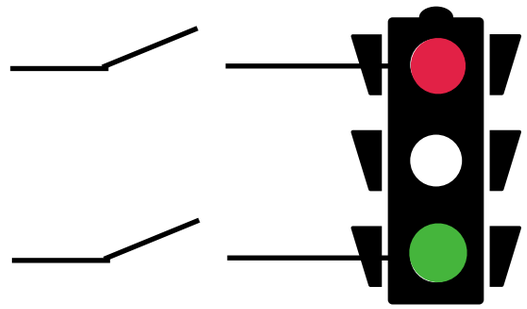


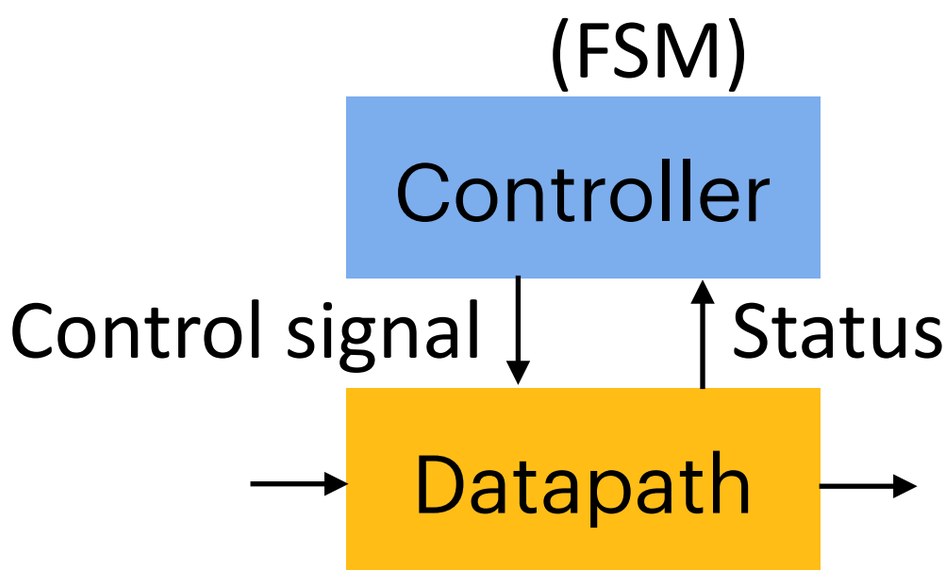
Tiny SDS Example II

- Traffic light: green & red (count down from 63 and change)
- Controller/datapath partition

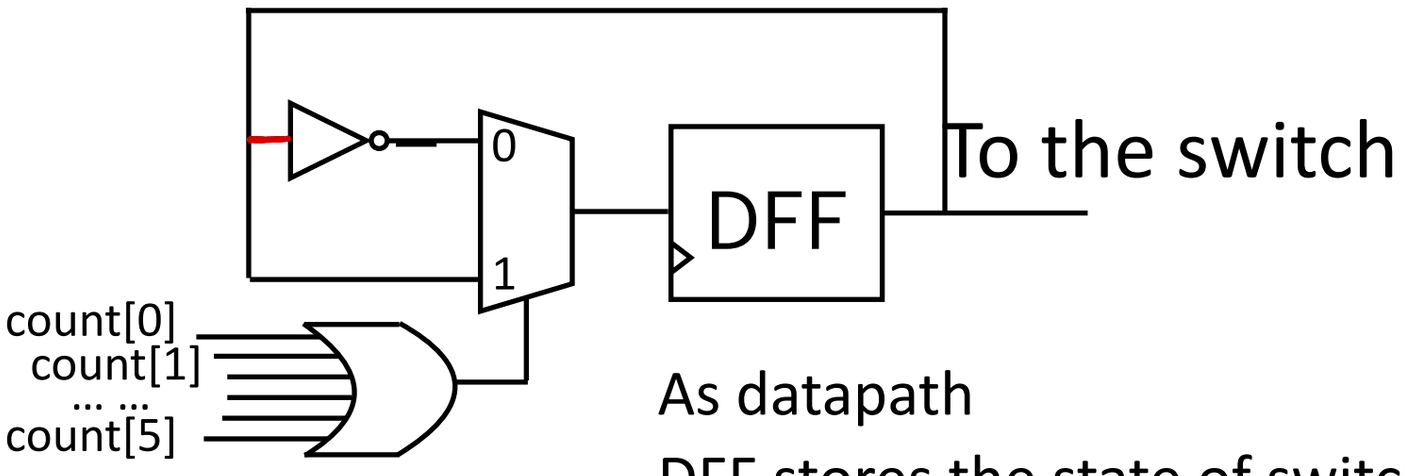


Cyclic down-counter as controller





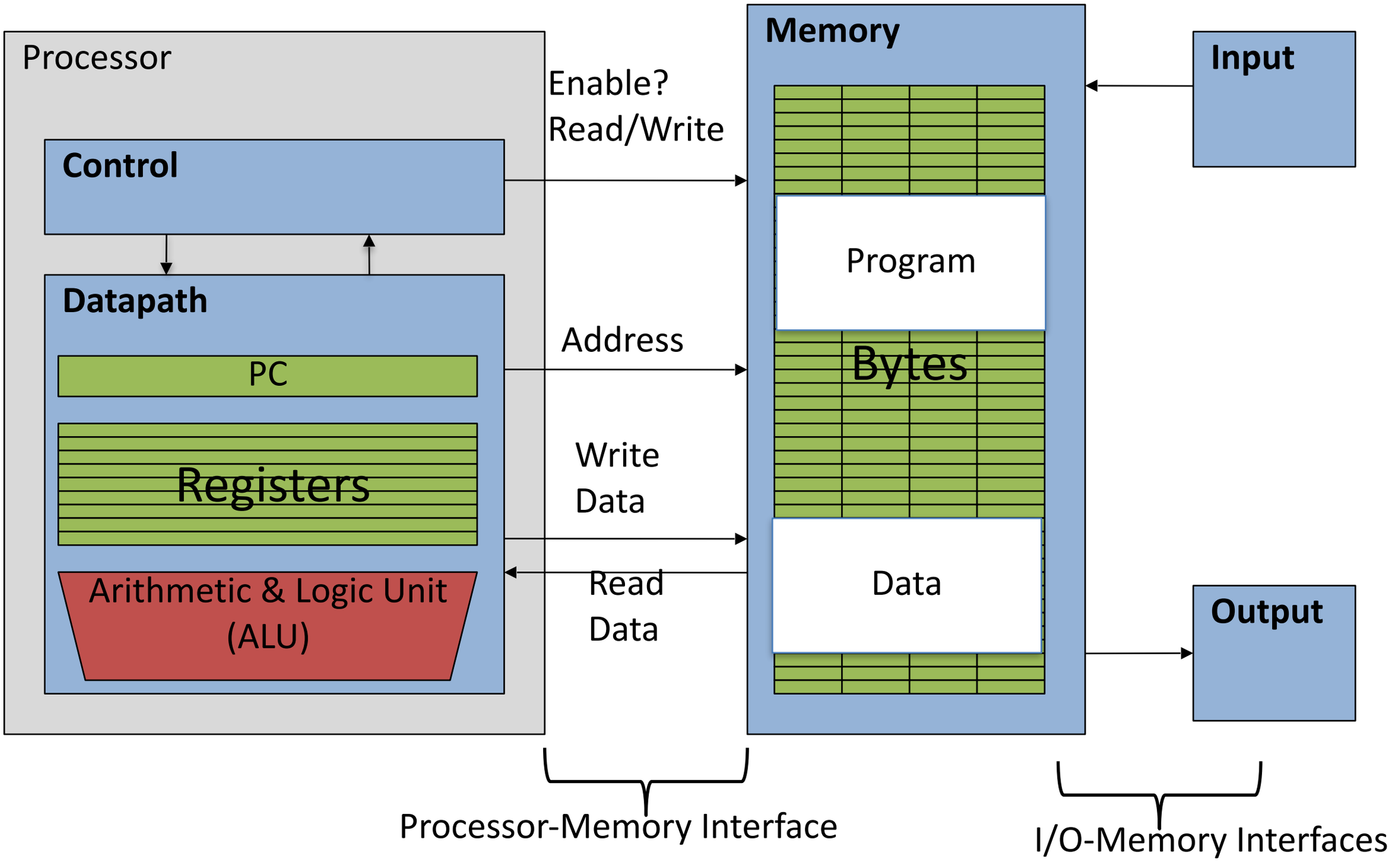
(Functional Units
Data storage
Etc.)



As datapath

DFF stores the state of switch

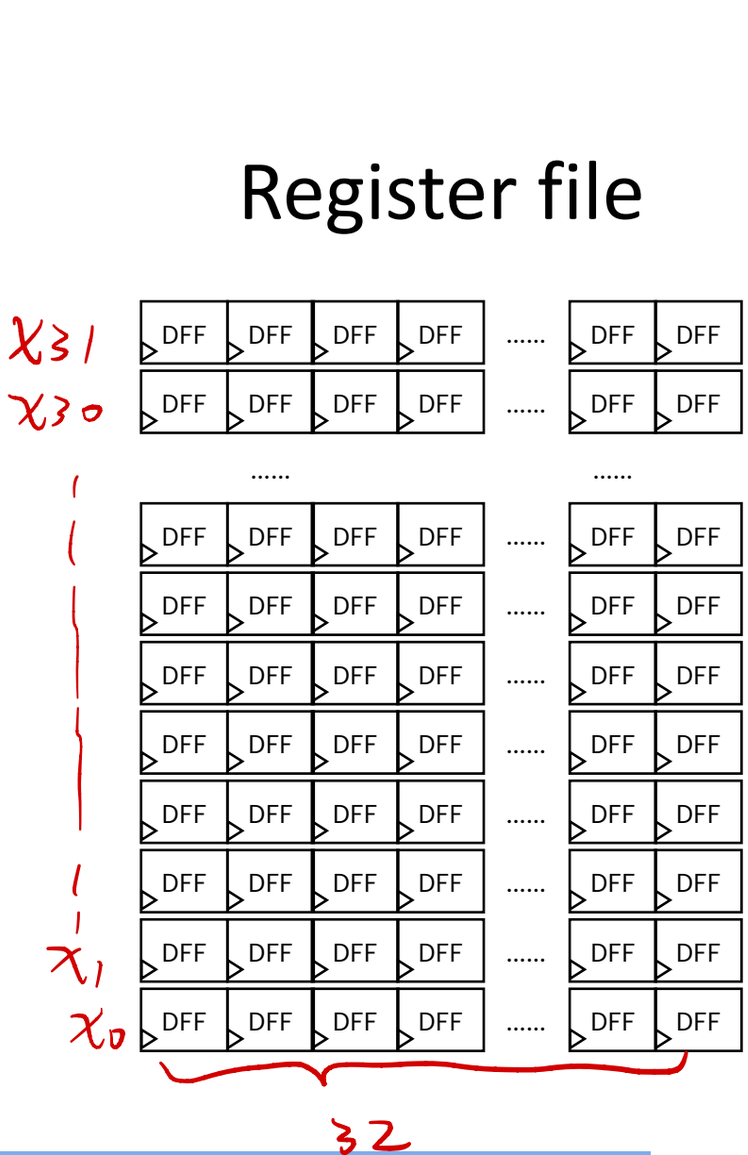
Components of a Computer

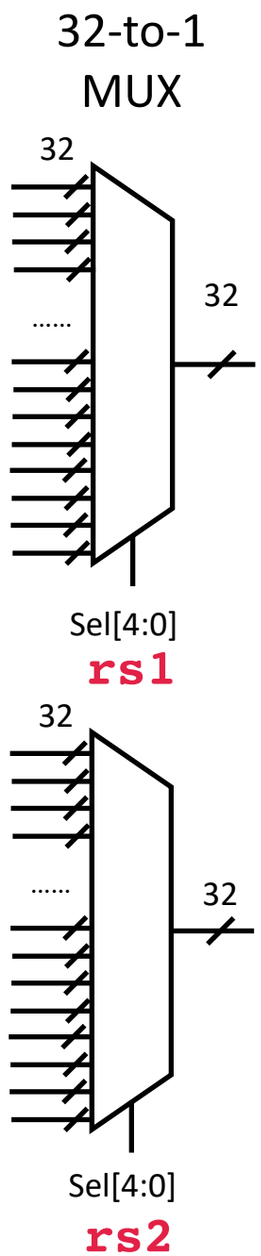


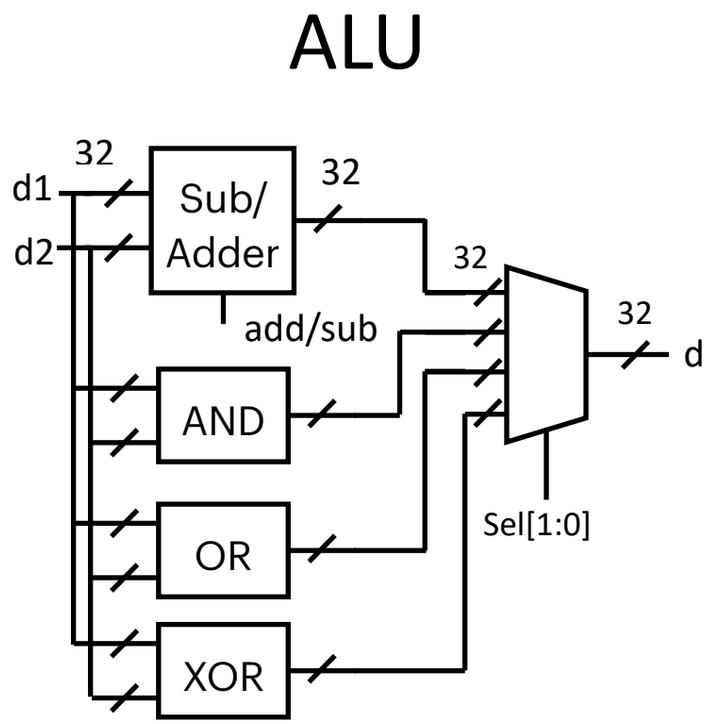
The CPU

- Processor (CPU): the active part of the computer that does all the work (data manipulation and decision-making)
- Datapath: portion of the processor that contains hardware necessary to perform operations required by the processor
- Control: portion of the processor (also in hardware) that tells the datapath what needs to be done

Components of a Computer





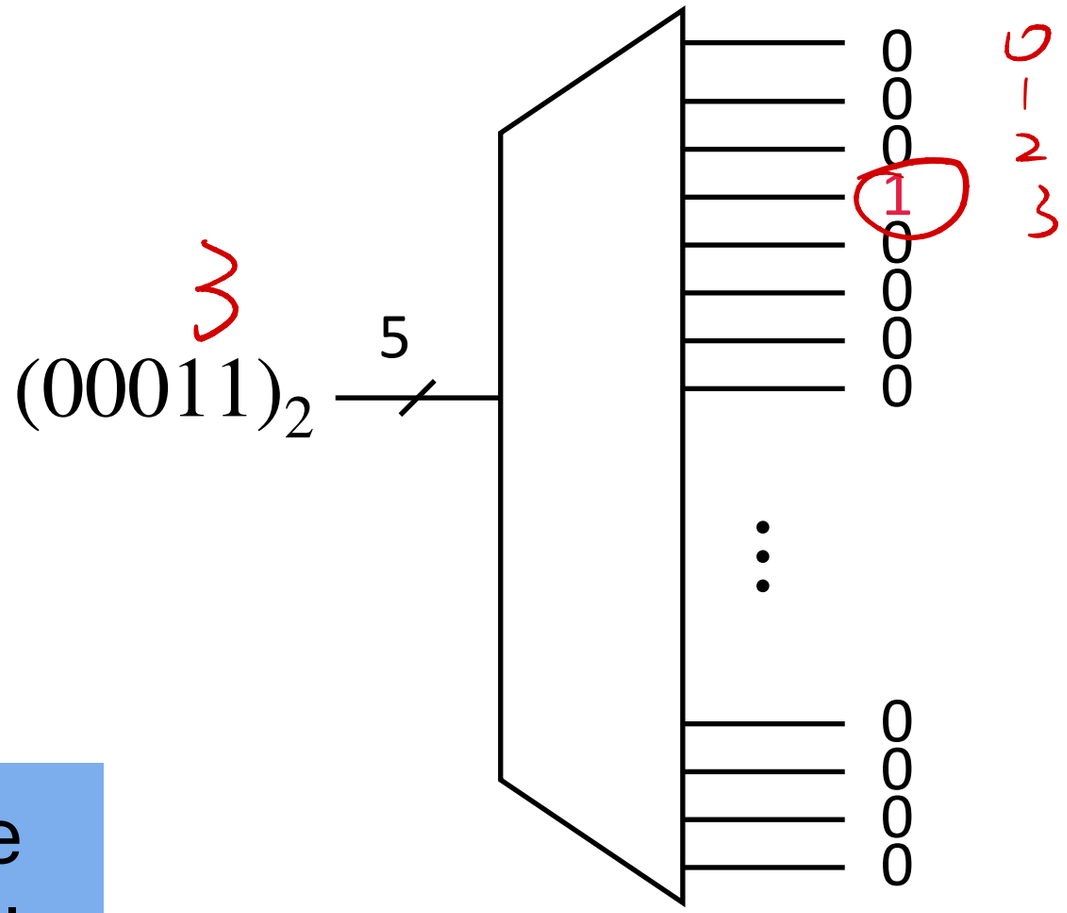


What about selecting which register to write?

A New Combinational Component — Decoder

- A component that convert a binary number to one-hot code format.

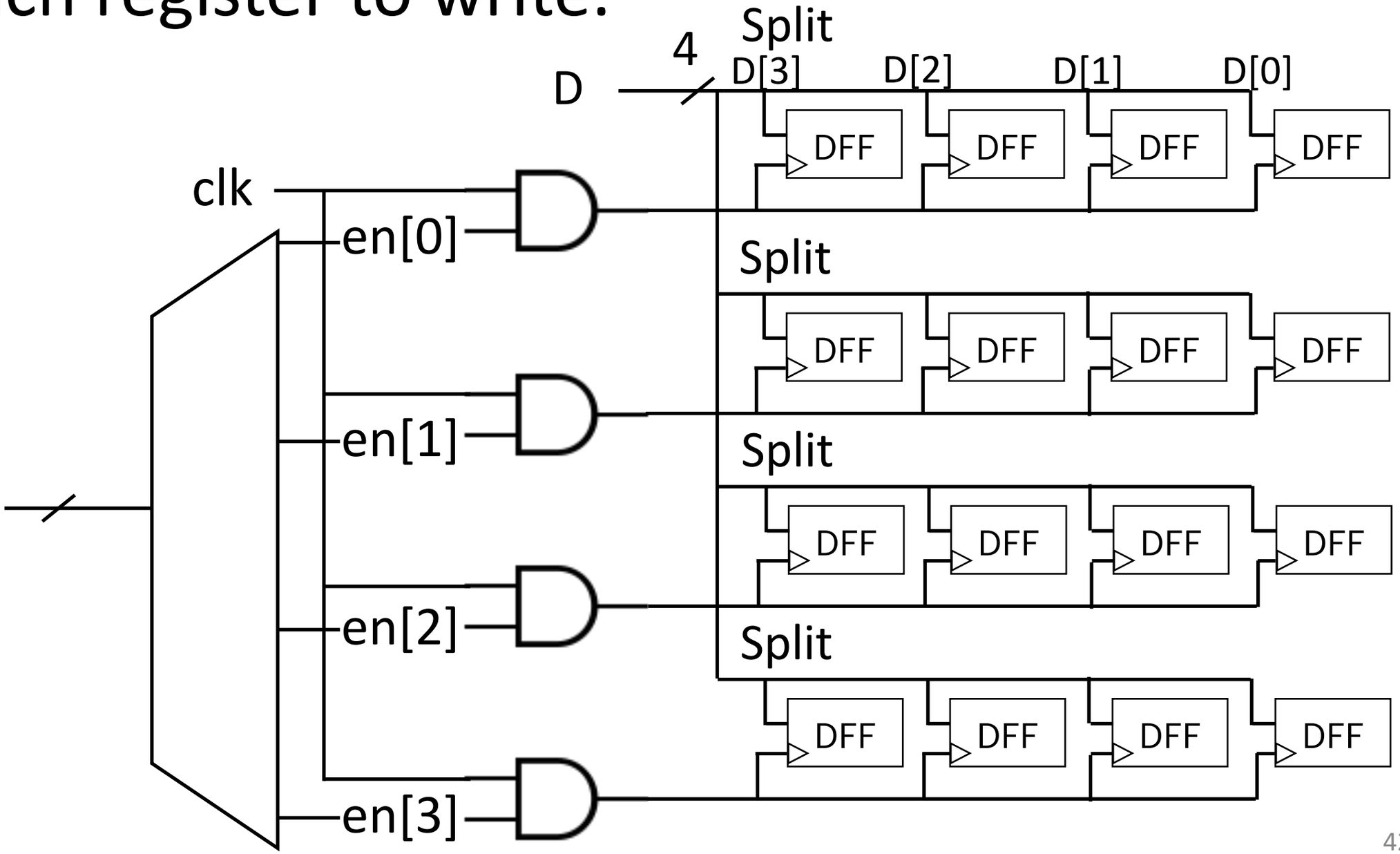
独热码



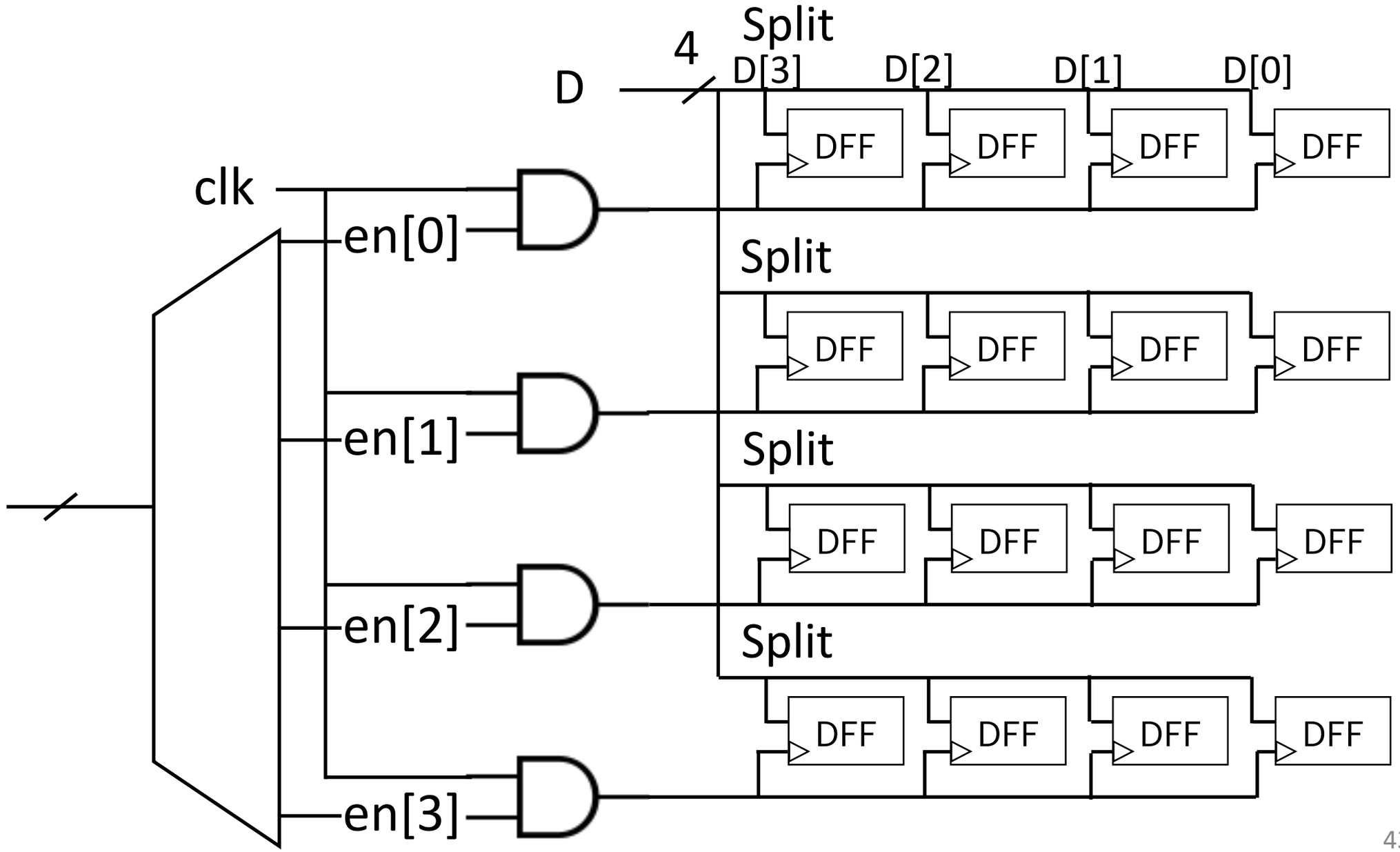
Actual circuit can be obtained by truth table

A New Combinational Component — Decoder

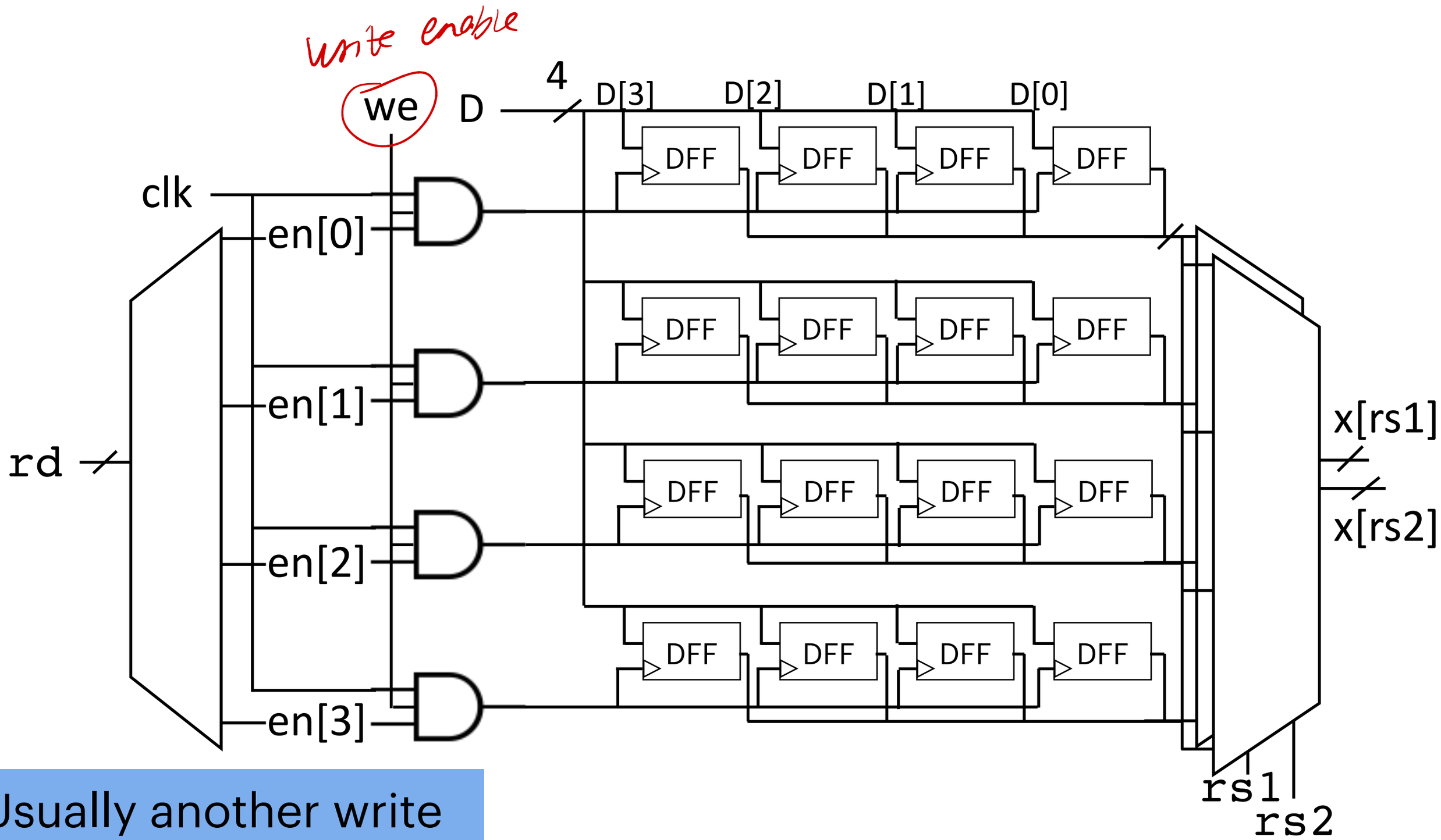
- Along with an “enable” signal, we can choose which register to write.



Full Register File

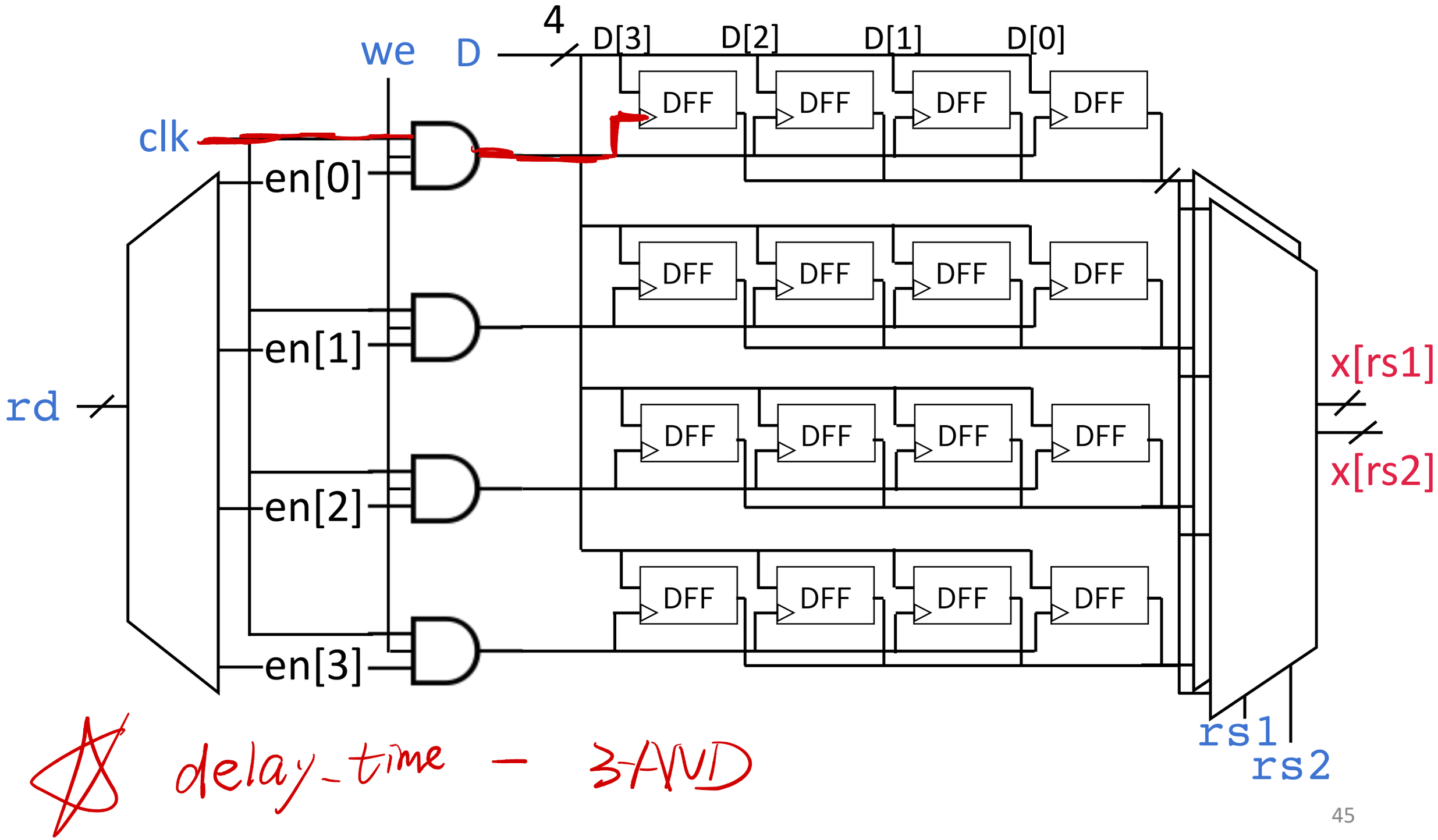


Full Register File



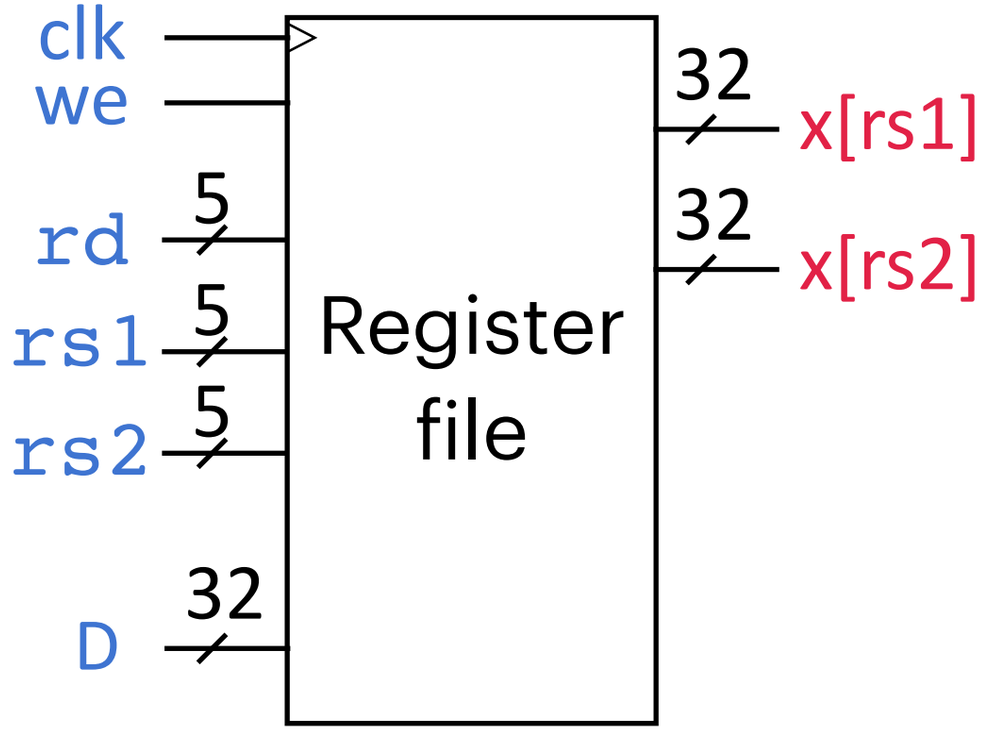
Usually another write enable signal

Full Register File—a Symbol



delay_time — 3-AND

Full Register File—a Symbol



Admin

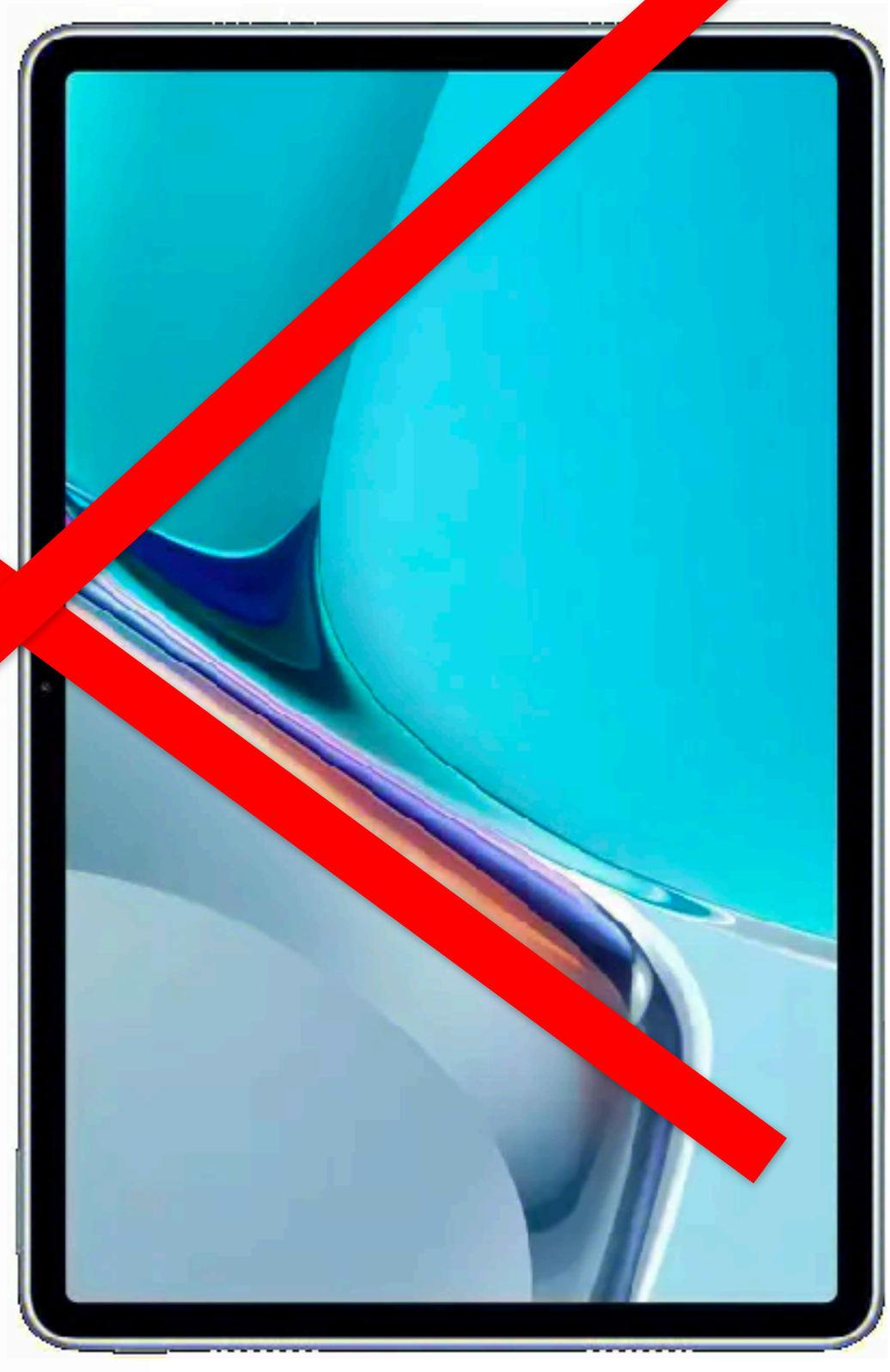
- Midterm I
  - March 21st 8:00 am - 10:00 am
    - We start sharp at 8:00 am!
    - Arrive 7:45 am to check-in (three classrooms and seat table will be determined on-site)
    - Arrive later then 8:30 am will get 0 mark.
- Contents:
  - Everything till Datapath (March 16th lecture)

Midterm I

- Switch cell phones **off**!!!
  (not silent mode – off!!!)
  - Put them in your bags.
- Bags in the front. On the table: nothing but:
  pen, 1 drink, 1 snack, your student ID card and your cheat sheet!
- The RISC V green card will be provided
- No other electronic devices are allowed!
  - No ear plugs, music, smartwatch…
- Anybody touching any electronic device will **FAIL** the course!
- Anybody found cheating (copy your neighbors answers, additional material, ...) will **FAIL** the course!


CASIO
JF-120TM
12 digits
TWO WAY POWER
F CUT 5/4
4 3 2 1 0 A
+/−
MRC
M−
TAX−
RATE SET
%
TAX RATE
TAX+
GT
C
ON
AC
7
4
8
9
6
X
0
00
1
3
=
+


08:08
Thu, Sep 1
HUAWEI



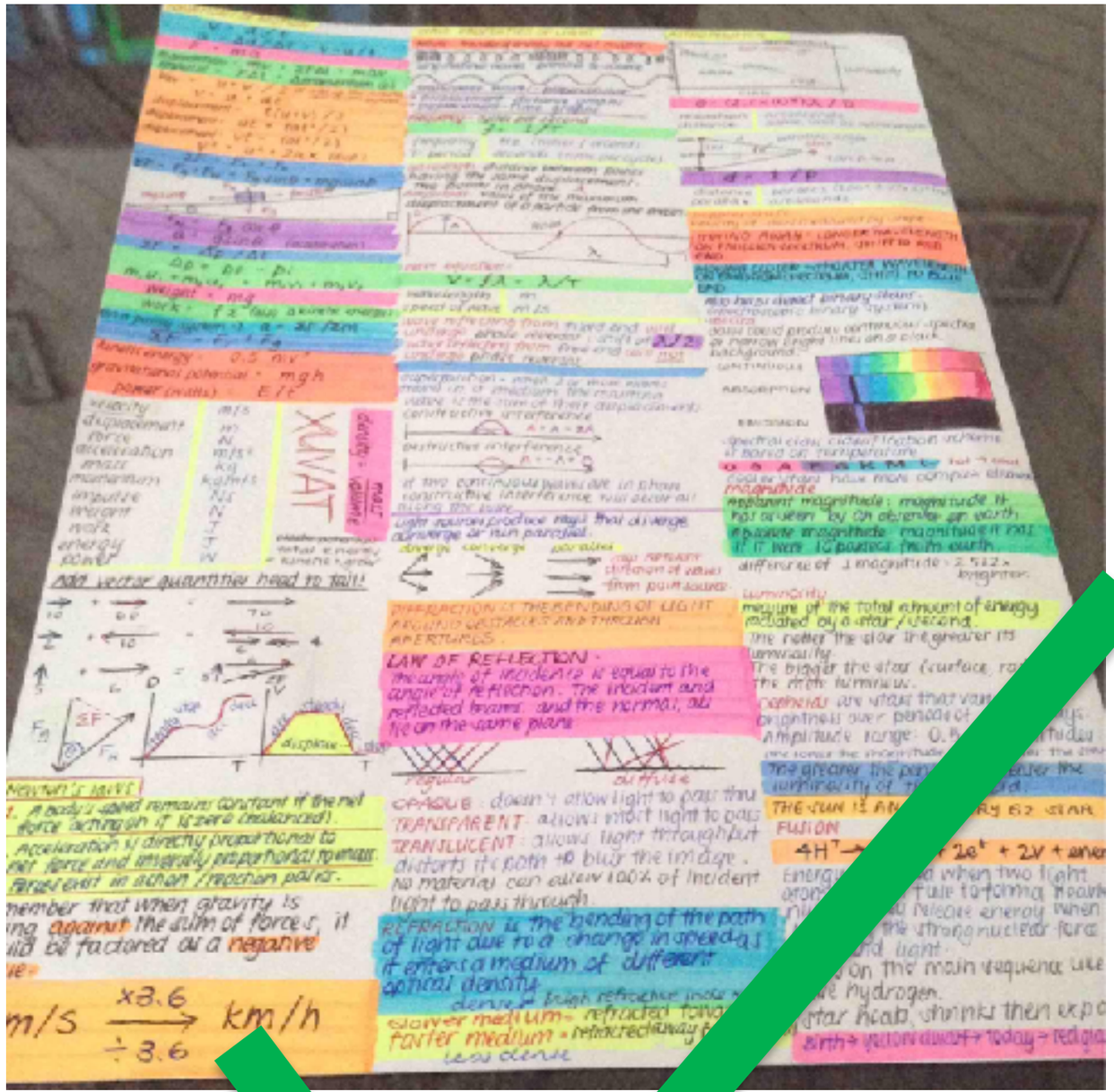

Tue
9

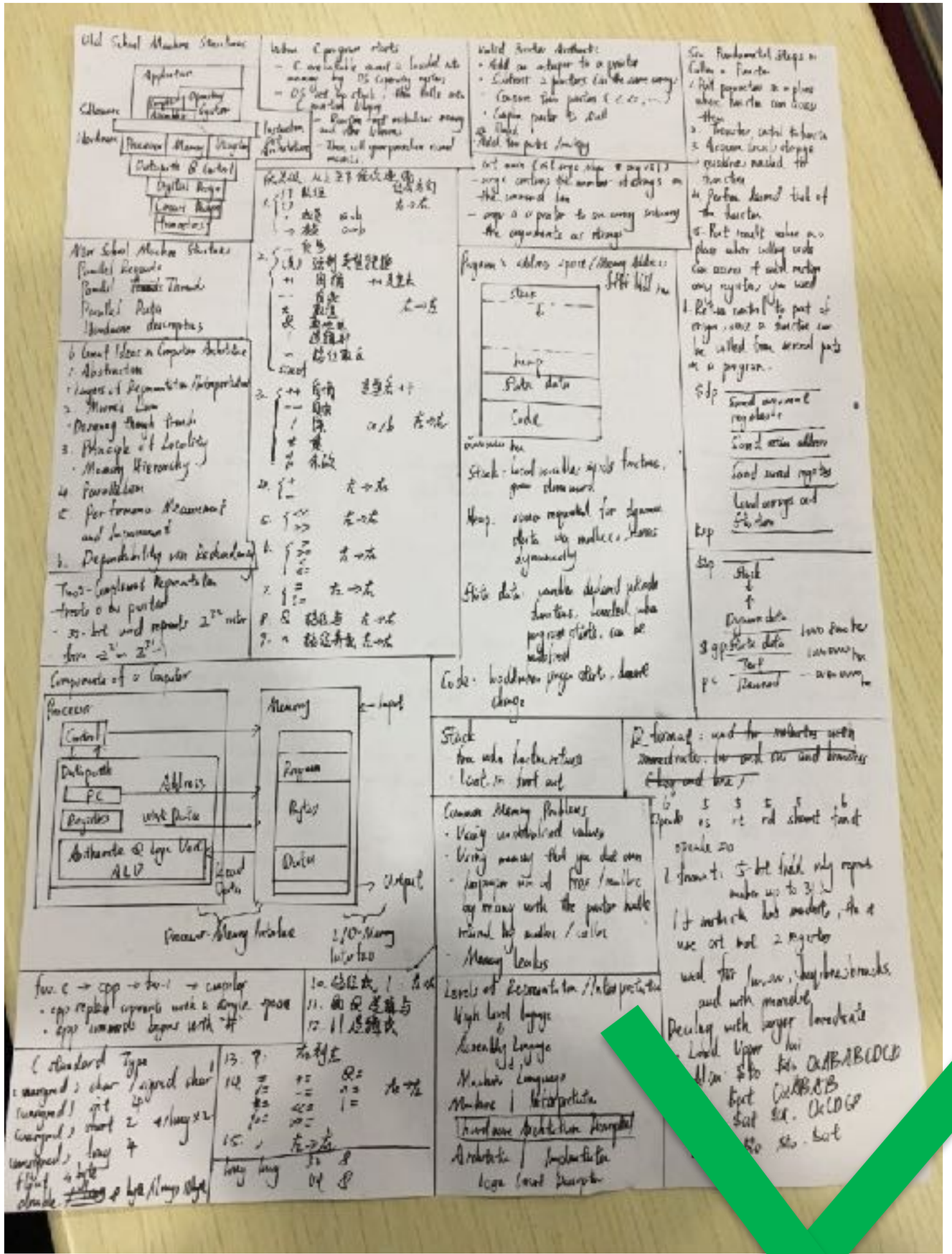

COMPUTER
ORGANIZATION
AND DESIGN
THE HA WARE/SOFTWA INTERFACE
RISC-V EDITION
DAVID A. PATTERSON
JOHN L. HENNESSY

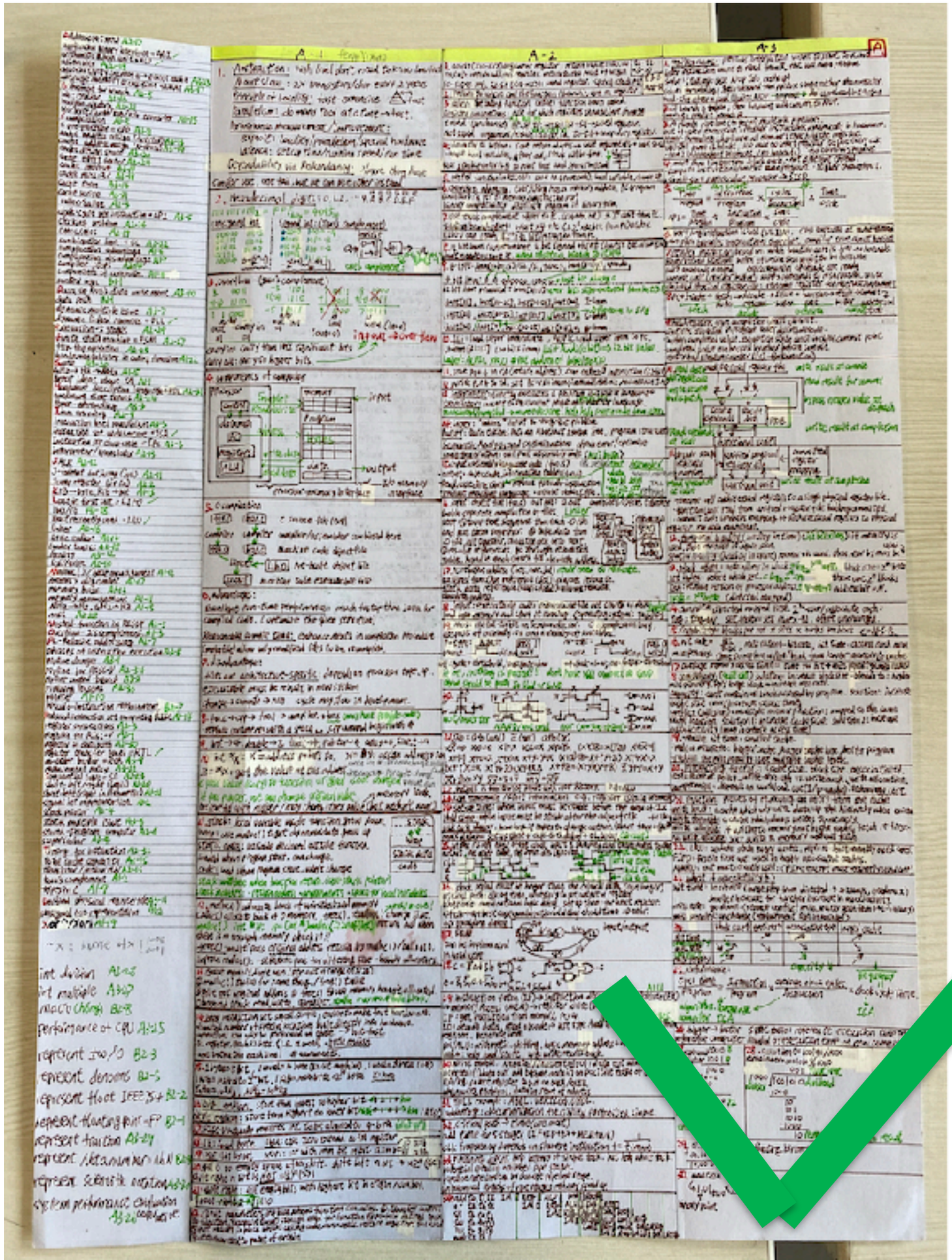
Cheat Sheet

- 1 A4 Cheat Sheet allowed (double sided)
  - Midterm II: 2 pages
  - Final: 3 pages
- Rules:
  - *Hand-written* – **not printed/photocopied!**
  - Your **name** in pinyin on the top!
  - Cheat Sheets not complying to this rule will be **confiscated**!


FUNCTIONS OF SEVERAL VARIABLES
LEVEL CURVES
FUNCTIONS OF n VARIABLES
PARTIAL DERIVATIVES
HIGHER PARTIAL DERIVATIVES
CLAIRAUT'S THEOREM
PARTIAL DIFF. EQS
THE CHAIN RULE
CASE 1
CASE 2
IMPLICIT DIFFERENTIATION
TANGENT PLANE TO A LEVEL SURFACE
NORMAL LINE TO A LEVEL SURFACE
MAXIMUM AND MINIMUM VALUES
DOMAINS
ε-δ DEFINITION OF CONTINUITY
TOTAL DIFFERENTIAL
INCREMENTS
THE GRADIENT VECTOR
DIRECTIONAL DERIVATIVES


XUVAT
LAW OF REFLECTION
REFRACTION
DIFFRACTION
THE SUN IS AN
FUSION
km/h
÷3.6


Old School Machine Structures
Application
Operating System
Software
Hardware
Processor
Memory
Input/Output
Components of a Computer
Processor
Memory
Input
Output
Control
Datapath
PC
Registers
Arithmetic & Logic Unit
ALU
Address
Program
Bytes
Data
Processor-Memory Interface
I/O-Memory Interfaces
Stack
Heap
Static data
Code
Levels of Representation / Interpretation
High Level Language
Assembly Language
Machine Language
Hardware Architecture Description
Architecture / Implementation
Logic Circuit Description



SIFT REFERENCE GUIDE (V.1.1) – CREATING TIMELINES WITH THE SIFT WORKSTATION

SANS COMPUTER FORENSICS and INCIDENT RESPONSE

THE PURPOSE OF THIS REFERENCE GUIDE IS TO WALK THROUGH THE PROCESS OF BOOTING THE SIFT WORKSTATION, CREATING A TIMELINE ("SUPER" OR "MICRO") AND REVIEWING IT.

1. VISIT: http://computer-forensics11.sans.org/community/downloads

1 → Download: SIFT Workstation VM Appliance

Download: SIFT Workstation Installation

2. BOOT SIFT VM

2 → Login: *sansforensics*
Password: *forensics*

3. ELEVATE PRIVS

3 → $ sudo su

4. CONNECT IMAGE TO SIFT

4 → Plug hard drive to physical host and attach to SIFT VM

5

5. HARD DRIVE MOUNTING *(if you are using log2timeline-sift and Single DD you can skip to 7-A)*

EWF/

SINGLE OR SPLIT IMAGE (2 options):

mount_ewf.py image.E01 /mnt/ewf

mount_ewfmount image.E01 /mnt/ewf/

Not Needed For 7-A

mount -t ntfs -o ro,loop,show_sys_files,streams_interface=windows,offset=#### /mnt/ewf/<image> /mnt/windows_mount/

MOUNT TO MOUNT POINT

DD

SINGLE IMAGE

mount -t ntfs -o ro,loop,show_sys_files,streams_interface=windows,offset=#### image.dd /mnt/windows_mount/

SPLIT IMAGE (2 step process)

affuse image.001 /mnt/aff

mount -t ntfs-3g -o loop,ro,show_sys_files,streams_interface=windows,offset=#### /mnt/aff/<image> /mnt/windows_mount/

6

HOW TO CALCULATE THE OFFSET FOR MOUNTING

1. Run mmls to query partition layout
 # mmls image.E01
2. Identify partition and byte offset
3. (Partition byte offset) x (bytes per sector) = **offset #### to use!**
Example: 63 X 512 = 32256

Note: If needed, repeat for each partition. Make new mount point:
 # mkdir /mnt/windows_mount2/

6. log2timeline default timezone is set to examiner local host. To change use -z [TIMEZONE] option. To list all available timezones:
 # log2timeline -z list

7-A & 7-B

log2timeline PARSING PLUGINS

apache2_error - Apache2 error log file
chrome - Chrome history file
encase_dirlisting - CSV file that is exported from encase
evt - Windows 2k/XP/2k3 Event Log
evtx - Windows Event Log File (EVTX)
exif - Metadata information from files using ExifTool
ff_bookmark - Firefox bookmark file
firefox2 - Firefox 2 browser history
firefox3 - Firefox 3 history file
ftk_dirlisting - CSV file that is exported from FTK Imager (dirlisting)
generic_linux - Generic Linux logs that start with MMM DD HH:MM:SS
iehistory - index.dat file containg IE history
iis - IIS W3C log file
isatxt - ISA text export log file
jp_ntfs_change - CSV output file from JP (NTFS Change log)
mactime - Body file in the mactime format
mcafee - Log file
mft - NTFS MFT file
mssql_errlog - MSSQL Log file produced by MS SQL server
ntuser - NTUSER.DAT registry file
opera - Opera's global history file
oxml - OpenXML document pcap
pcap - PCAP file
pdf - Available PDF document metadata
prefetch - Prefetch directory
recycler - Recycle bin directory
restore 9.0 - Restore point directory
safari - Safari History.plist file
sam - SAM registry file
security - SECURITY registry file
setupapi - setupAPI log file in Windows XP
skype_sql - Skype database
software - SOFTWARE registry file
sol - .sol (LSC) or a Flash cookie file
squid - Squid access log (http_emulate off)
syslog - Linux Syslog log file
system - SYSTEM registry file
tln - Body file in the TLN format
volatility - Volatility output files (psscan2, sockscan2, ...)
win_link - Windows shortcut file (or a link file)
wmiprov - wmiprov log file
xpfirewall - XP Firewall log

List plugins # log2timeline -f list
...HELP EXPAND THIS LIST. BUILD PLUGINS!!

BY DAVID NIDES (12/16/2011)
TWITTER: @DAVNADS
BLOG: DAVNADS.BLOGSPOT.COM
EMAIL: DNIDES@KPMG.COM
CREDITS TO: ED GOINGS, ROB LE... KRISTINN GUDJONSSON, KPMG & ... QUESTIONS/FEEDBACK–CONTACT US!

KEY
Red text – image/source
Blue text – mount point
Purple text – output file
Green text – log2timeline plugins
Brown text - Timezone

8

7-A: AUTOMATED SUPER TIMELINE CREATION

log2timeline-sift: -o –z [TIMEZONE] -p [PARTITION #] -i [IMAGE FILE]

DISK IMAGE (prompt for partition, mount, and run):

XP: # log2timeline-sift –z EST5EDT -i image

WIN7: # log2timeline-sift -win7 -z EST5EDT -i image

FOR PARTITION (mount and run using all applicable plugins):

XP: # log2timeline-sift –z EST5EDT -p 0 -i partition

WIN7: # log2timeline-sift -win7 -z EST5EDT -p 0 -i partition

OTHER USAGE EXAMPLES:

Display list of available plugins:
 # log2timeline -f list
Run log2timeline using only specific plugins:
 # log2timeline-sift -p prefetch –z EST5EDT -i image.dd
Help (man page):
 # log2timeline -h

8. CSV FORMAT OUTPUT (/cases/timeline-output-folder)

-date: date of file event, in the format of MM/DD/YYYY
-time: time of day, expressed in a 24h format, HH:MM:SS
-timezone: the timezone that was used to call the tool with.
-MACB: MACB meaning of the fields, comp w/ mactime format.
-source: Source short name (i.e. registry entries are REG)
-sourcetype: Description of the source ("Internet Explorer" instead of WEBHIST)
-type: Timestamp type (i.e. "Last Accessed", "Last Written")
-user: Username associated with the entry, if one is available.
-host: Hostname associated with the entry, if one is available.
-short: Contains less text than the full description field.
-desc: where majority info is stored, the actual parsed desc of the entry.
-version: Version number of the timestamp object.
-filename: Filename with the full path that contained the entry
-inode: inode number of the file being parsed.
-notes: Some input modules insert additional information in the form of a note, which comes here. Or it can be used during the review.
-format: Input module name used to parse the file.
-extra: Additional information parsed is joined together and put here.

9

9. FILTER TIMELINE

Filter timeline with date range to include only:
l2t_process -b timeline.csv MM-DD-YYYY..MM-DD-YYYY > filtered.csv
Filter timeline with keyword list (one term per line in keywords.txt):
l2t_process -b timeline.csv -k keywords.txt > filtered.csv
What sources are in your timeline?
awk -F, '{print $6;}' timeline.csv | grep -v sourcetype | sort | uniq
Find all LNK files that reference E Drive
grep "Shortcut LNK" timeline.csv | grep "E:"
FindMountPoints2 entries that reference E Drive
grep "MountPoints2 key" timeline.csv) | grep "E drive"
grepUSB timeline.csv | grep "SetupAPILog"

File System	M	A	C	B
Ext2/3	Modified	Accessed	Changed	N/A
FAT	Written	Accessed	N/A	Created
NTFS	File Modified	Accessed	MFT Modified	Created
UFS	Modified	Accessed	Changed	N/A

MANUAL "MICRO" TIMELINE CREATION

...TIONS] [-f FORMAT] [-z TIMEZONE] [-o OUTPUT MODULE] [-w ...G_FILE/LOG_DIR [--] [FORMAT FILE OPTIONS]

...EM METADATA (using log2timeline or fls)

...stem data w/log2timeline from mounted file system:
 # log... ...f mft -o mactime -r ...z EST5EDT -w mft.bo... ...olume/
OR Extract M... ...age using Sleuthkit:
 # fls -m "" -o offs... ...dd > fls.body
Convert body file for... ...format w/ mactime:
 # mactime -b fls.body -d

ARTIFACTS (run l2t on mounted file s... ...h plugins recursively)

Extract artifacts w/ log2timeline and ru... ...d file system:
 # log2timeline -f firefox3,chrome -o mactim... ...EDT -w web.body /mnt/volume/
Convert body file format to CSV format w/ mactime:
 # mactime -b log2timeline.body -d > log2timeline.csv

HELP? OPTIONS? USAGE?
log2timeline -help
Log2timeline-sift -help
L2t_process -help

OTHER log2timeline OUTPUT FORMATS
Note: CSV is Default Output
-BeeDocs - Mac OS X visualization tool
-CEF - Common Event Format - ArcSight
-CFTL - XML file CyberForensicsTimeLab visualization tool
-CSV - comma separated value file
-Mactime - Both older and newer version of the format supported for use by TSK's mactime
-SIMILE - XML file - SIMILE timeline visualization widget
-SQLite - SQLite database
-TLN - Tab Delimited File
-TLN - Format used by some of H Carvey tools, expressed as a ASCII output
-TLNX - Format used by some of H Carvey tools, expressed as a XML document

10

10. CONNECT TO SIFT

✓ ...> SETTINGS -> OPTIONS -> Shared Folders -> Always Enabled (Check)

✓ 2. SIFT Desktop -> VMware-Shared-Drive

✓ Access from a Win Machine \\SIFTWORKSTATION

11

11. REVIEW TIMELINE

Review timelines using:
- Open, Sort, Filter with Excel
- Import into SP_LINK
- SIMILE
- Tapestry