Hilroy

80 Pages
27.6 cm x 21.2 cm

Ruled 7 mm • Ligné 7 mm

EXERCISE BOOK
CAHIER D'EXERCICES

NAME/NOM ______________________

SUBJECT/SUJET CPEN 211

ASSEMBLED IN CANADA WITH IMPORTED MATERIALS
ASSEMBLÉ AU CANADA AVEC DES MATIÈRES IMPORTÉES

12107

Digital Abstraction Combinational Logic Verilog

9/6/18

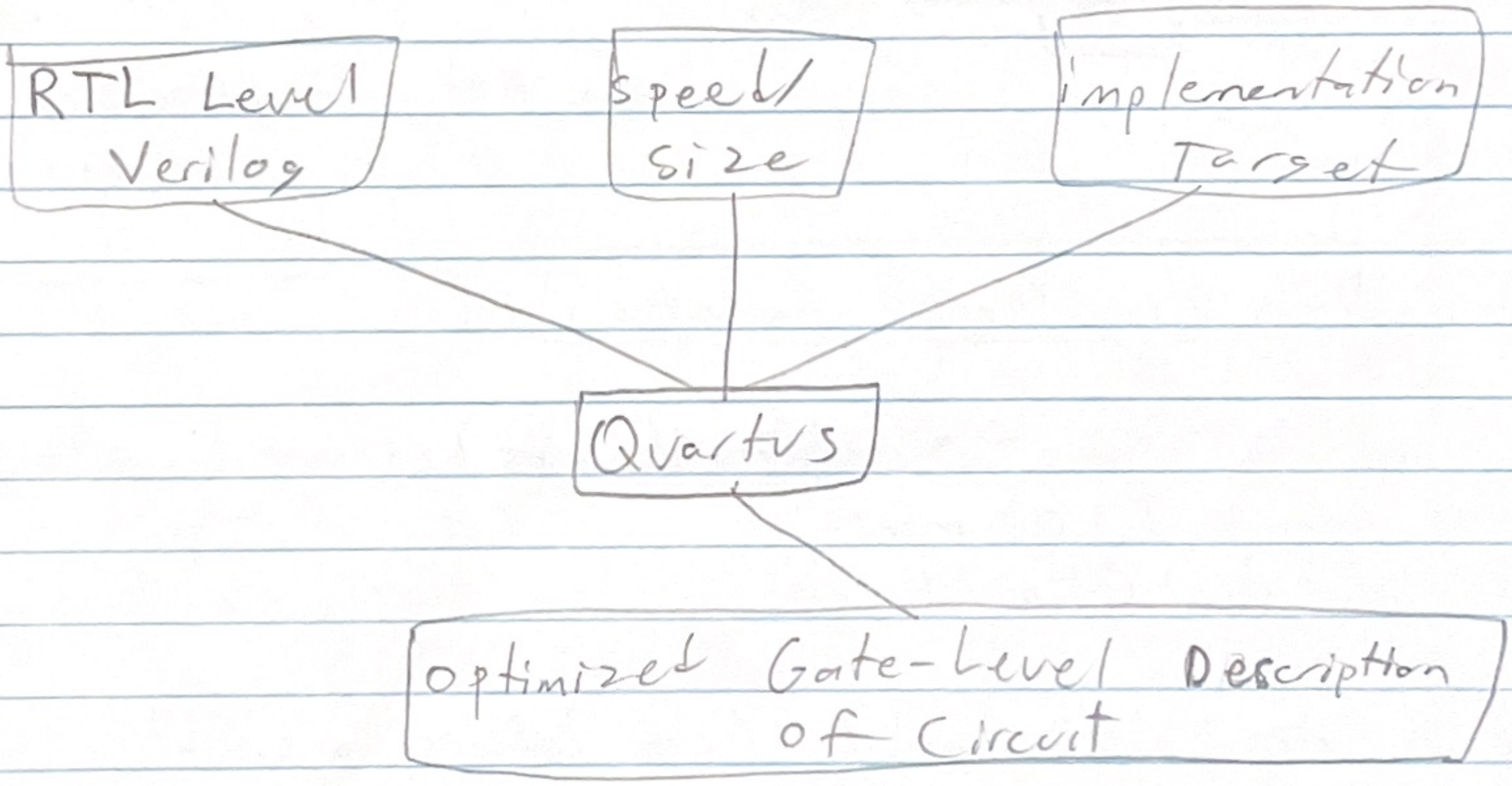
Reading (Dally)

Chapters: 1, 3, 7.1, 10.1, app. A for Slide set 1

Verilog

- Not a programming language
- Used to describe hardware
- Hardware Description Language HDL
- Used (2 ways)

① synthesis (Quartus)
② simulation (ModelSim)



- Basic Unit => Module; describes a block of hardware

Modules

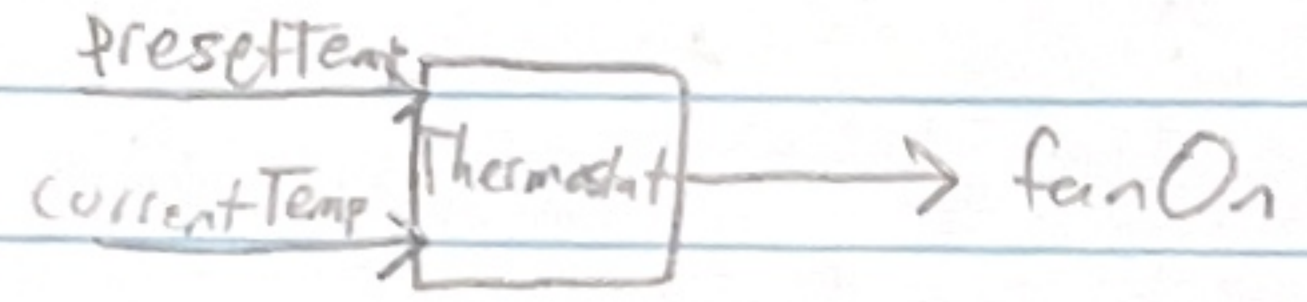
have...

- Module declarations
- input and output declarations
- internal signal declarations
- Logic definition
  - Assign statements - module instantiations
  - Case statements

[2:0]

ex. 1 **Veriloy for thermostat**

```
module Thermostat(presetTemp, currentTemp, fanOn);
  input [2:0] presetTemp, CurrentTemp; //temp inputs 3 bits e
  output fanOn; // true when current > preset //1 bit

  wire fanOn;
  assign fanOn = (currentTemp > presetTemp);
endmodule
```



*constantly changing 1 or 0

True or false

depending on inputs

ex.2 **Days in Month function**

```
module DaysInMonth(month, days);
  input [3:0] month;
  output [4:0] days;

  reg [4:0] days; //reg defines a signal set in an always bl

  always @(month) begin
    case(month)
      2: days = 5'd28;
      4,6,9,11: days = 5'd30;
      default: days = 5'd31; //5' because 5 bits wide
    endcase
  end
endmodule
```

reg does NOT define a register

- always block says evaluate this whenever month changes

```
    begin
///////// ───── always @(month)
end
```

- case is like Switch

input [3:0] month; => 4 bit value
input [2:0] month; => 3bit value

- Default covers values not listed

$\log_2 12 = 3.4 \rightarrow 4$
$\log_2 31 = 4.9 \rightarrow 5$

8
C

Unsigned non-negative integers

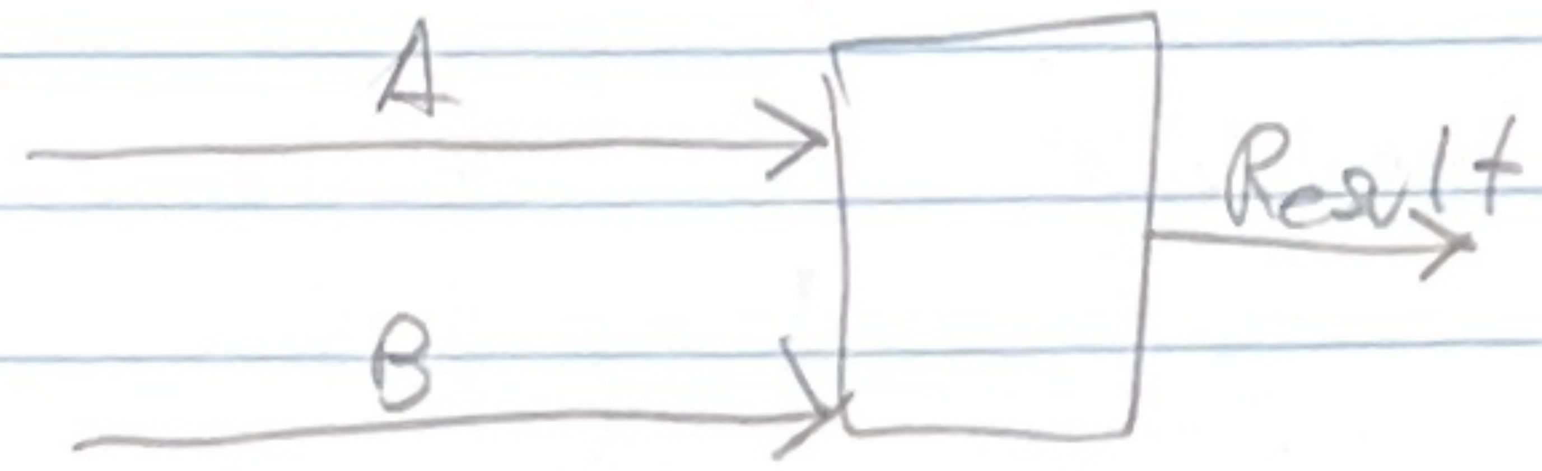
in Binary....

$00110 = 6$
$2^4\, 2^3\, 2^2\, 2^1\, 2^0 \quad = 2+4$

01001
$2^3\, 2^2\, 2^1\, 2^0$
$8+1$
9

Combinational Logic Circuit

- No memory
- only given by present inputs



output = f(input)

Sequential Logic Circuit

- Has memory
- Synchronas; uses a clock

Combinational + Combinational (without cycle)
= Combinational

Designing Combinational Logic

① Describe in Verilog
② minimize cost in implementation (Quartus)

How to Describe Combinational Logic

③ Boolean Algebra
④ Logic gate
⑤ Verilog

Boolean Algebra

=> over two numbers $\{0,1\}$

3 operations => AND ($\wedge$) OR ($\vee$) NOT ($\neg$)

AND Operation (conjunction)

$a \wedge b = 1$ if and only if $a = 1$
$b = 1$

OR operation (disjunction)

$a \vee b = 1$ if $a = 6$ or $b = 1$

NOT operation

$\neg a = 1$ only iff $a = 0$

Axioms

$0 \wedge x = 0$
$1 \vee x = 1$

$1 \wedge x = x$
$0 \vee x = x$

$\neg 0 = 1$
$\neg = 0$

Dual Functions

- Dual of logic function f, is function f^D derived by flipping each $\vee \Rightarrow \wedge$
$\wedge \Rightarrow \vee$
$1 \Rightarrow 0$
$0 \Rightarrow 1$

Example Rewrite the following in normal form
$f(x,y,z) =$ 1 if exactly 0 or 2 inputs are 1

xyz	f
000	1
001	0
010	0
011	1
100	
101	
110	1
111	0

$(\bar{x} \wedge \bar{y} \wedge \bar{z}) \vee (\bar{x} \wedge y \wedge z) \vee (x$

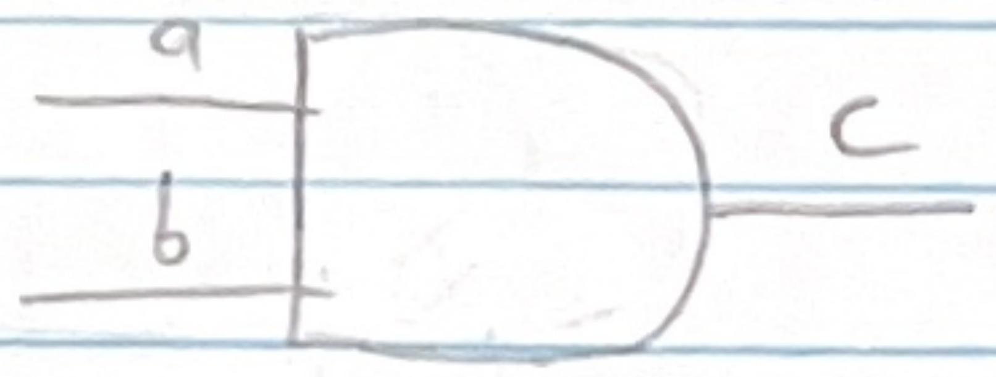
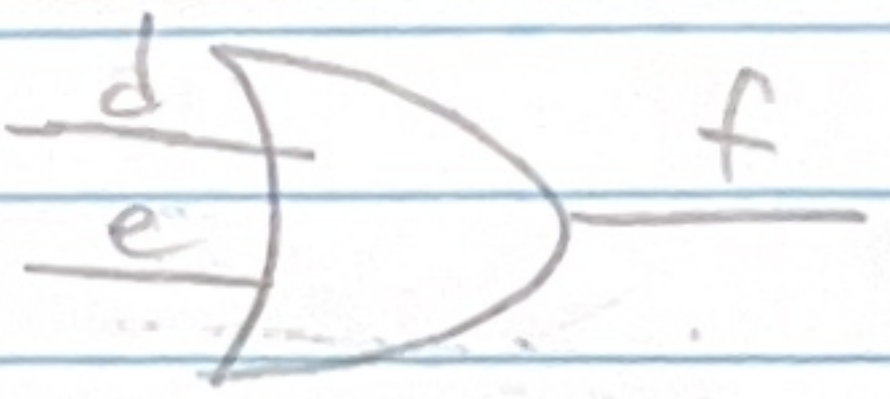
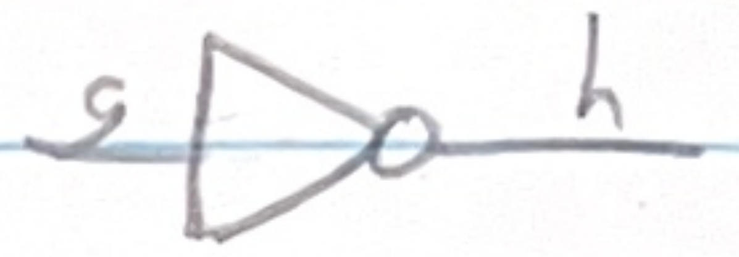
Normal Form => Sum of Products

Product term is a # of inputs or complements ANDed together

END of SLIDE SET 1

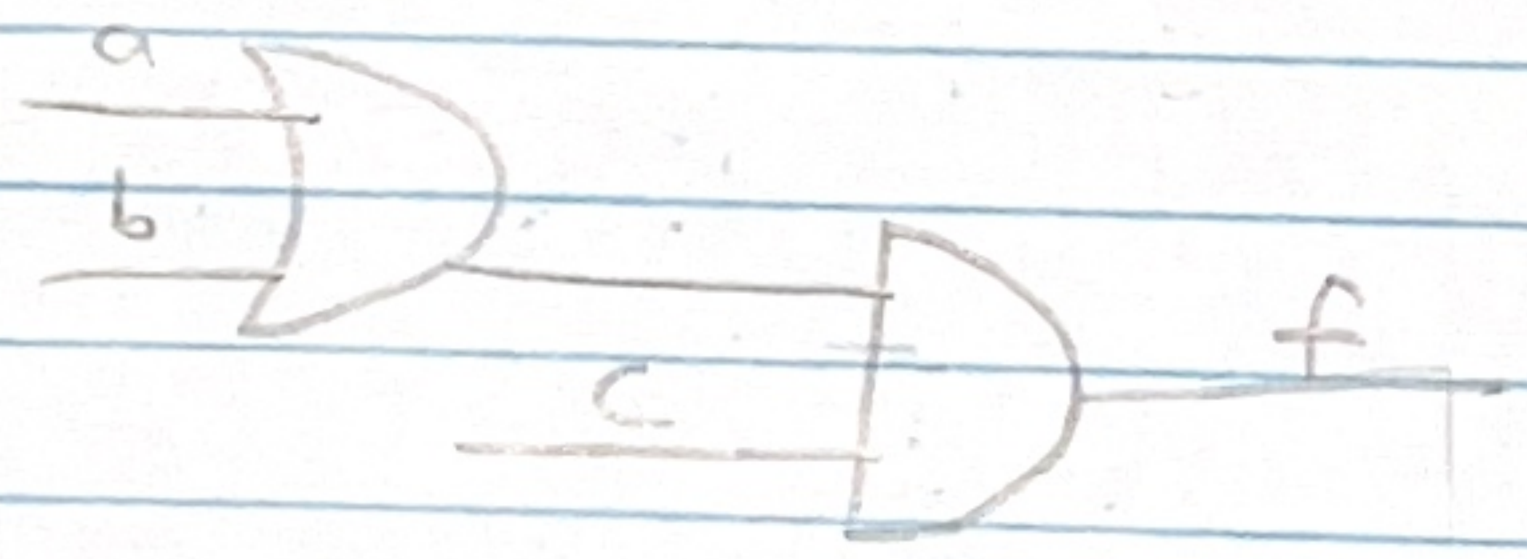
Slide Set #2

Basic Logic Gates

	AND gate	OR gate	NOT gate
Symbol			
Function	$c=a\land b$	$f=d\lor e$	$h=\neg g$

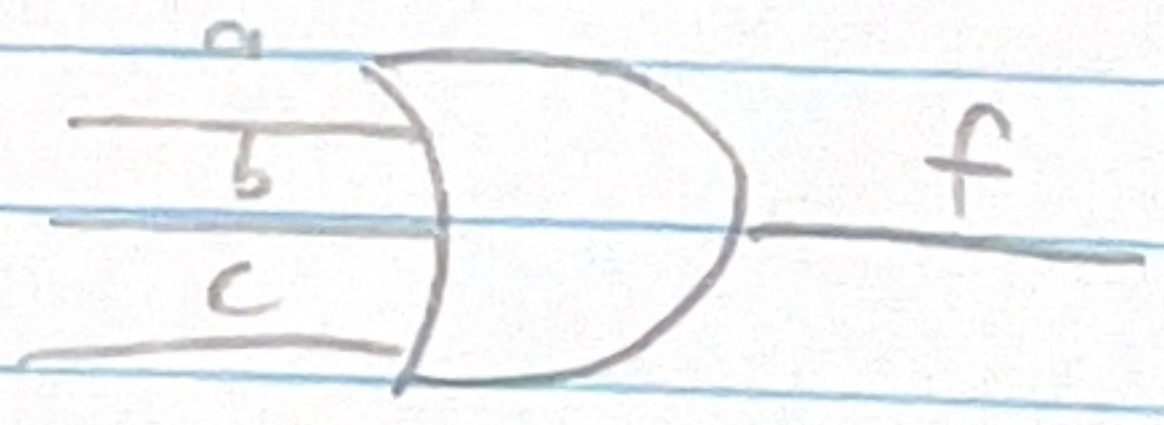
Connecting Gates

$f=(a\lor b)\lor c$



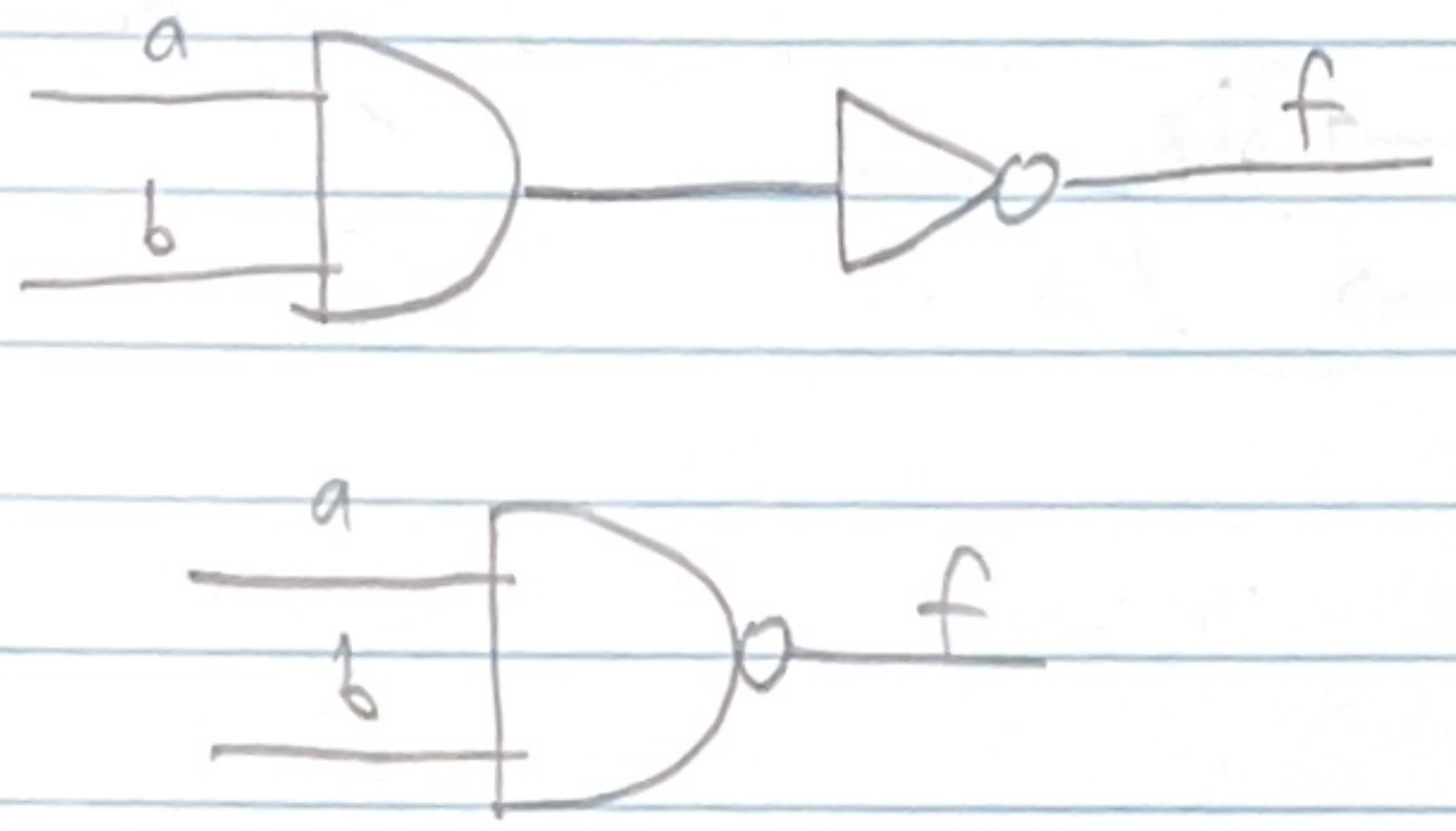
By Associative Rule

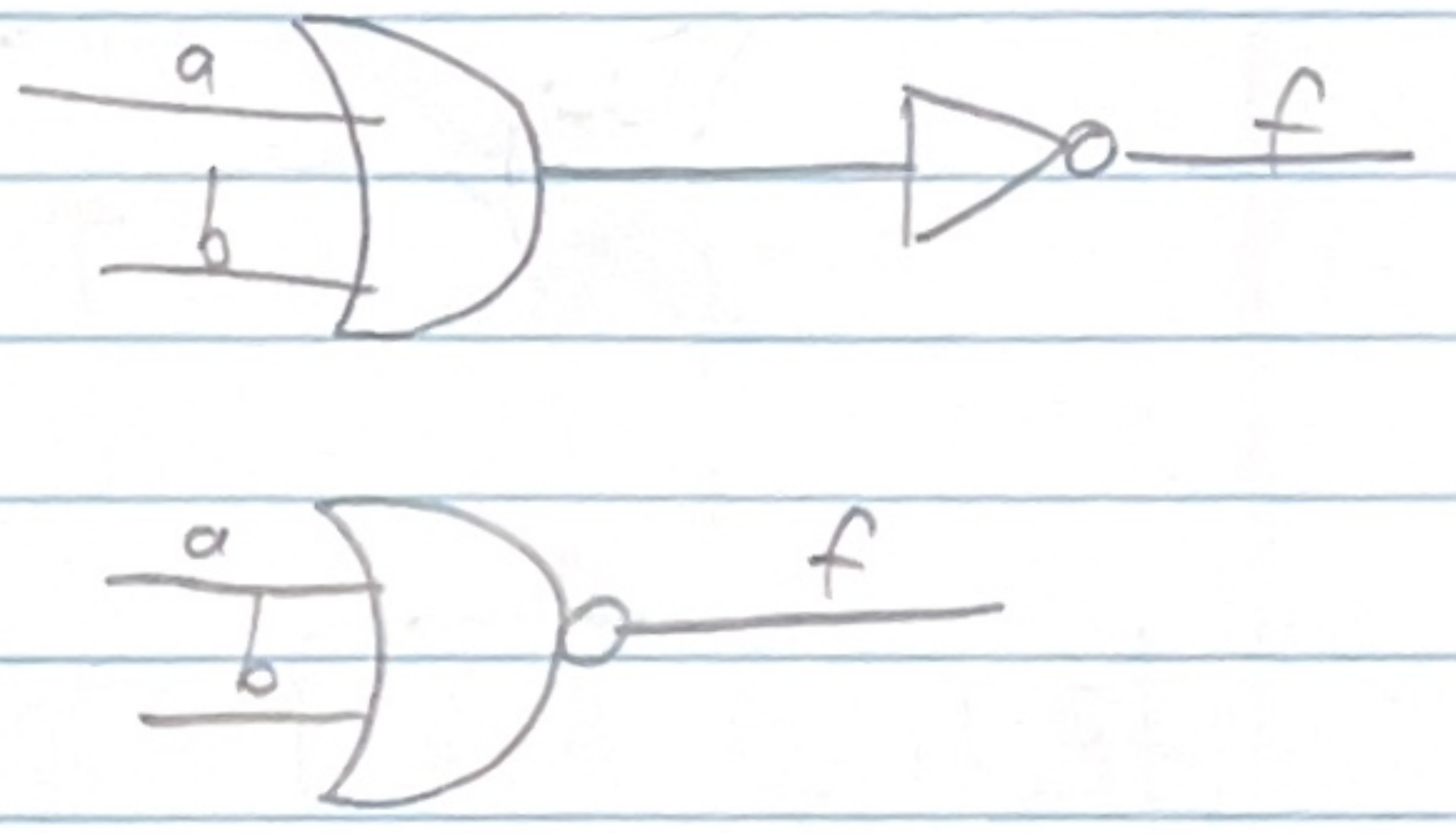
$$\begin{aligned} f &= (a\lor b)\lor c \\ &= a\lor(b\lor c) \\ &= a\lor b\lor c \end{aligned}$$



NAND Gate

NOR Gate

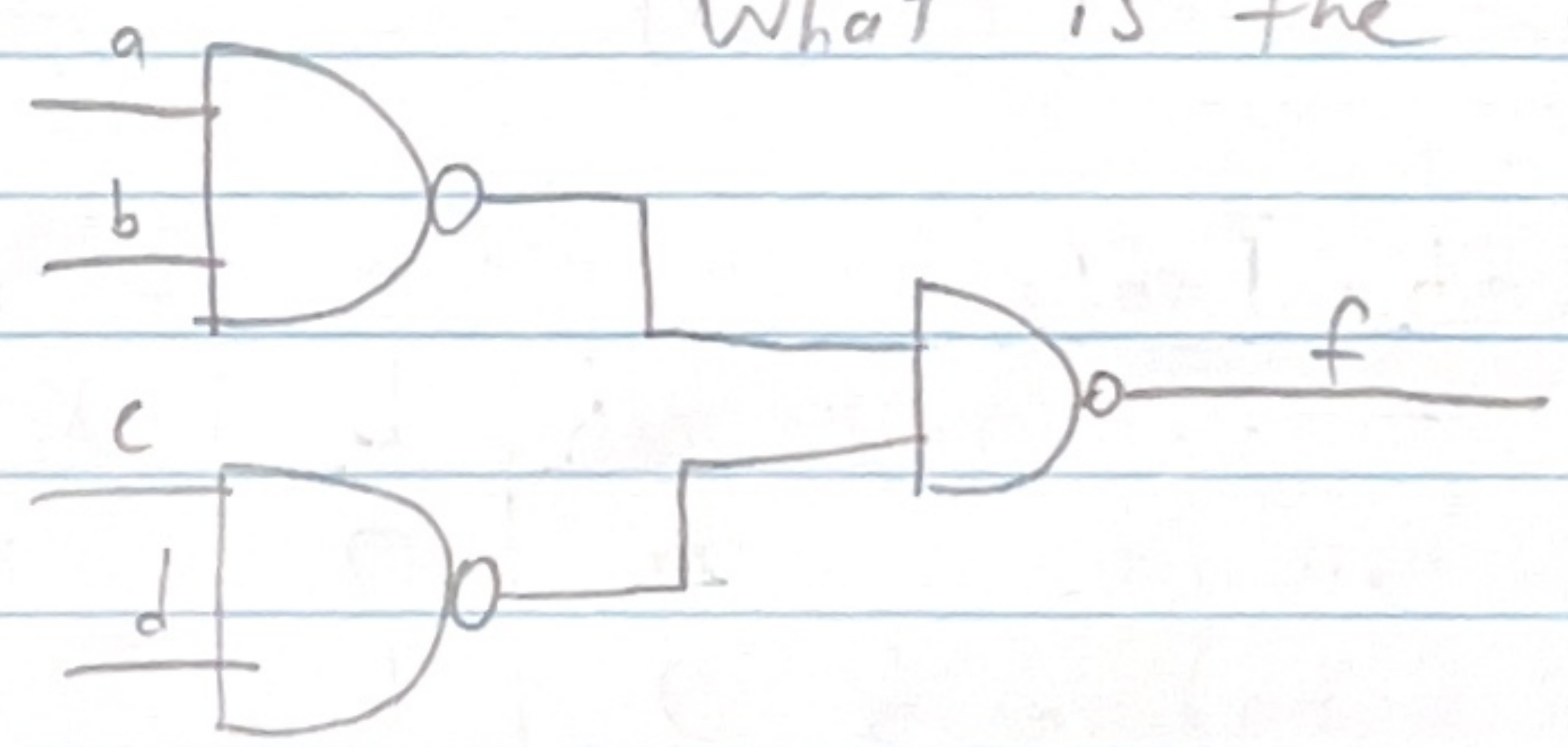




Why?

A: Fewer transistors *Explained later

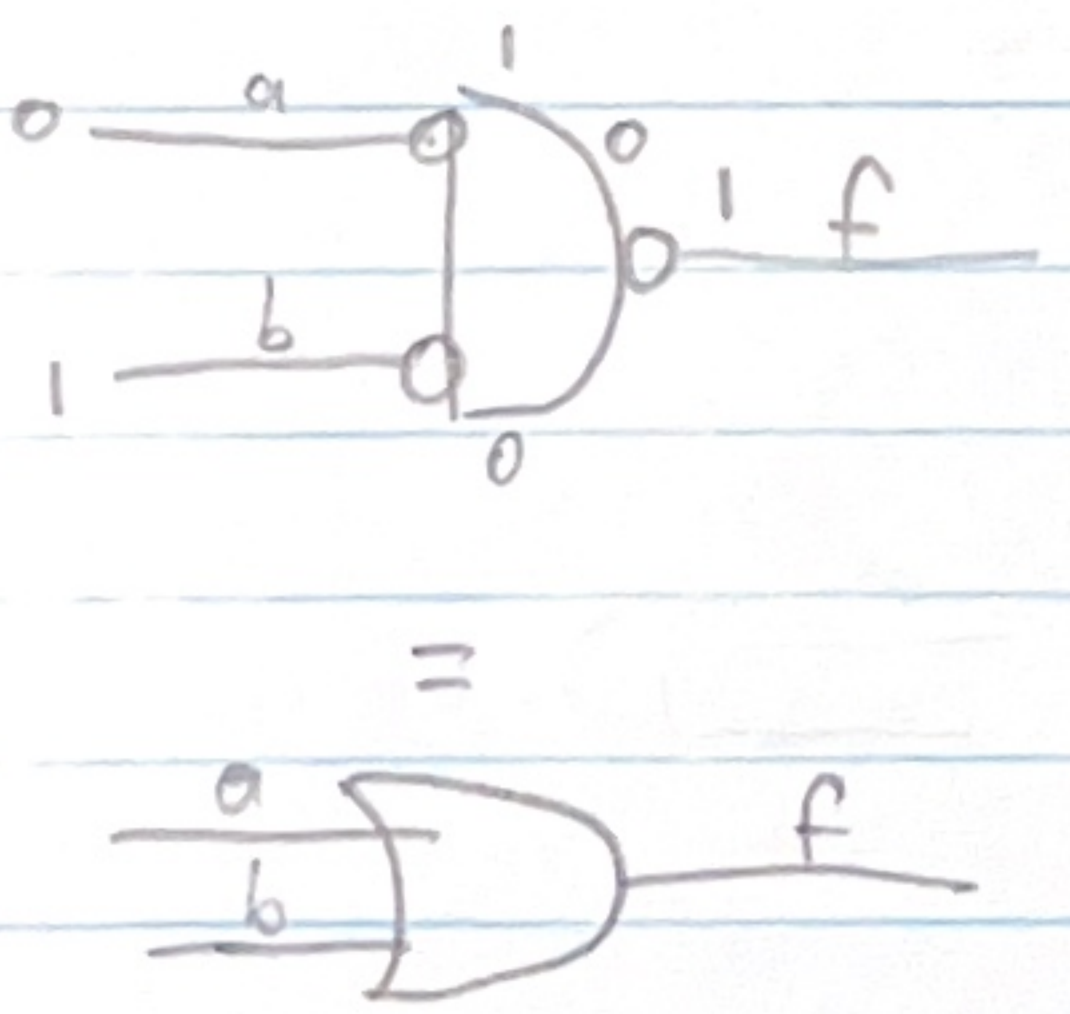
What is the output ie: f ?



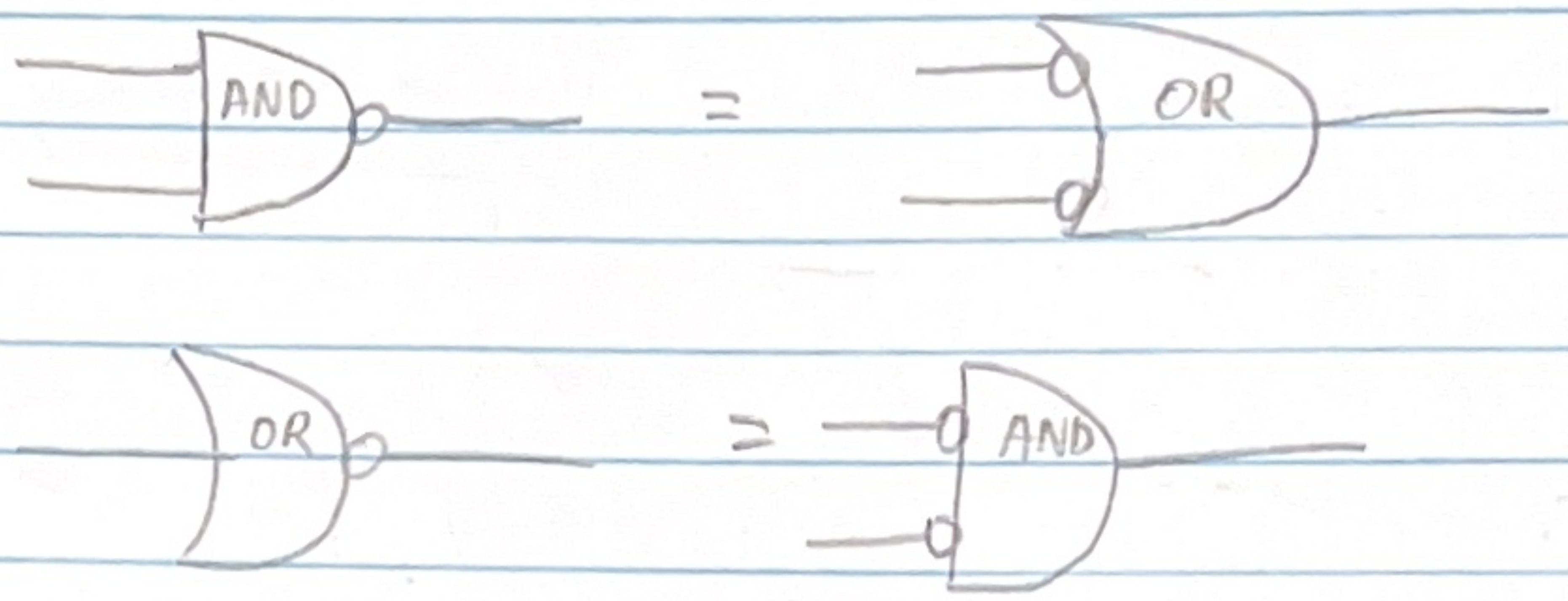
A:

$f = (a \land b) \lor$

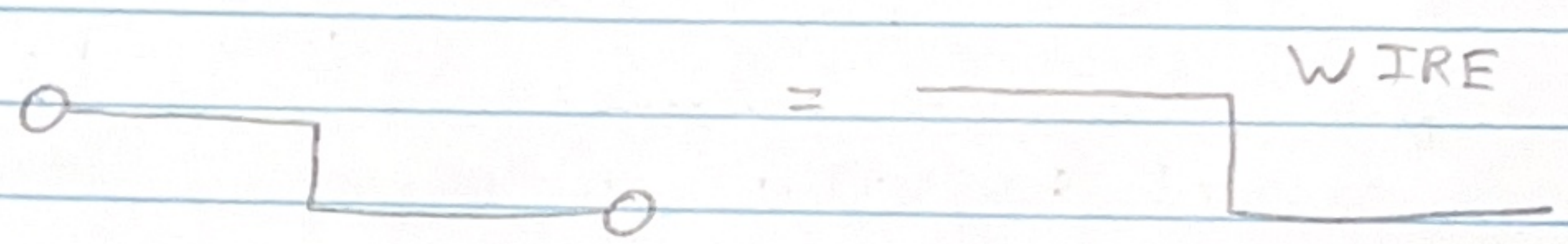
$\neg a$
$\bar{a}$
a' } equivilant



Demorgan



Bubble Rule



Exclusive OR (XOR)

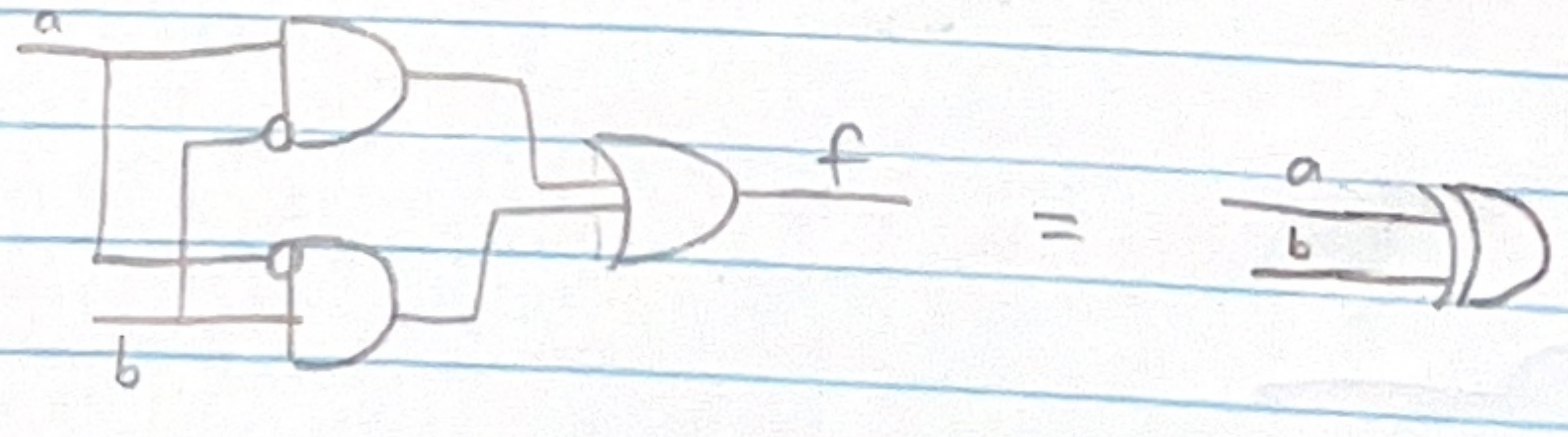
Truth Table:

*output high if one (not both) inputs is high

a	b	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Function $f(a,b) = (a \land \neg b) \lor (\neg a \land b)$

Logic Gate Diagram



Buses in Verilog

```
wire [2:0] X;
```

This defines X as a bus with 3 parallel wires

*if a signal is in an always or initial bloc
use reg instead of wire otherwise erro

```
[0:7] // 8 elements addressable with indices 0 to
[3:11] // 9 elements adressable with
            indices 3 to 11
[33:23] // 11 elements adressable
            indices 33 to 23
```

Verilog Literal numbers

8'sb0001_1101

<size> number of bits
<signed> if s ommitted then unsigned
<radix> b - binary
        d - decimal
        o - octal
        h - hexidecimal

*underscores ignored

0101
0110
0100

0X01

Flipped Lecture

00XX

dc \ ba	00	01	11	10
00	1 0	1 1	1 3	1 2
01	0 4	1 5	0 >	0 6
11	0 12	1 13	0 15	0 14
10	1 8	0 9	0 11	0 10

a (columns 01, 11) b (columns 11, 10) c (rows 01, 11) d (rows 11, 10)

dcba	out
0000	1
0001	1
0011	1
0010	1
0100	0
0101	1
0111	0
0110	0
1100	0
1101	1
1111	0
1110	0
1000	1
1001	0
1011	0
1010	0

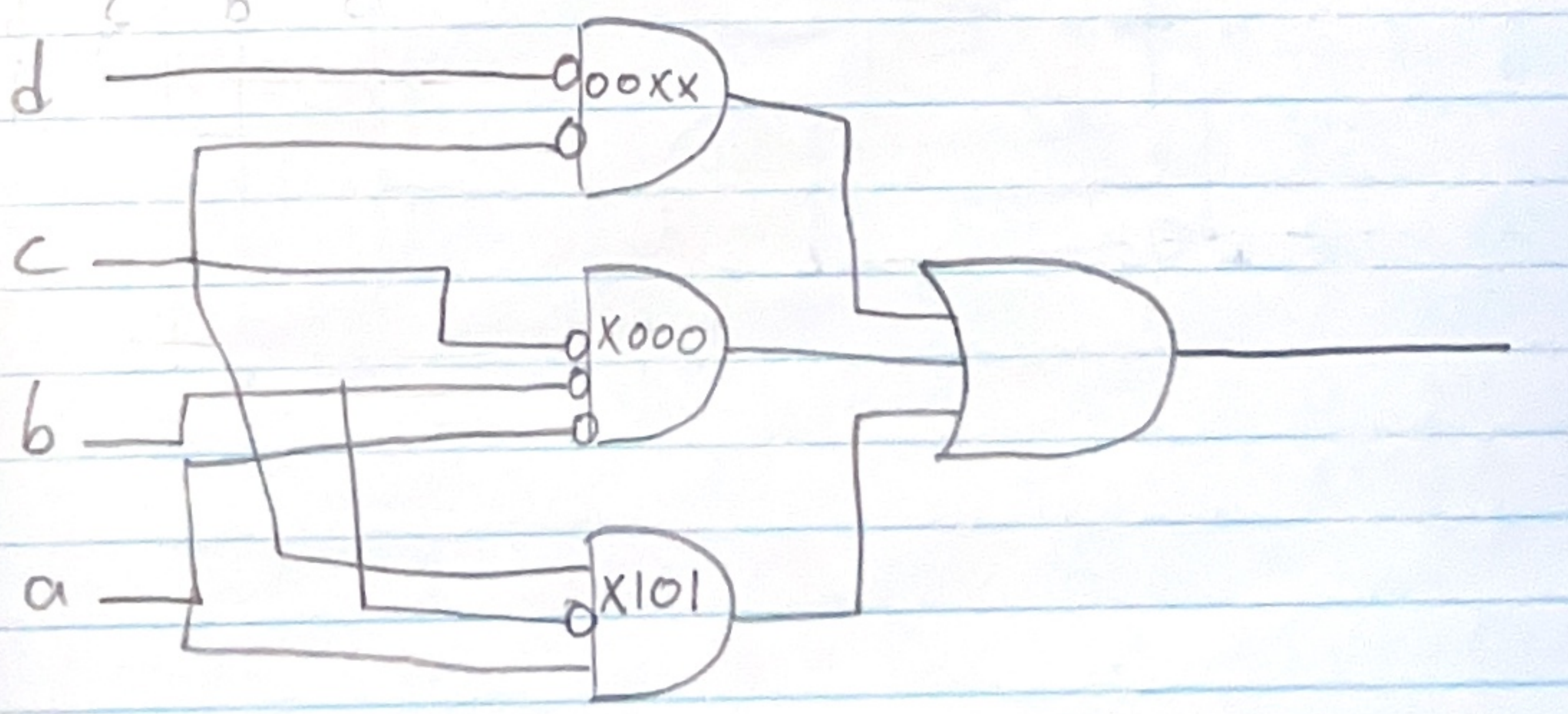
Min cost cover

0000

00XX, X000, X101

0011

d c b a



Question

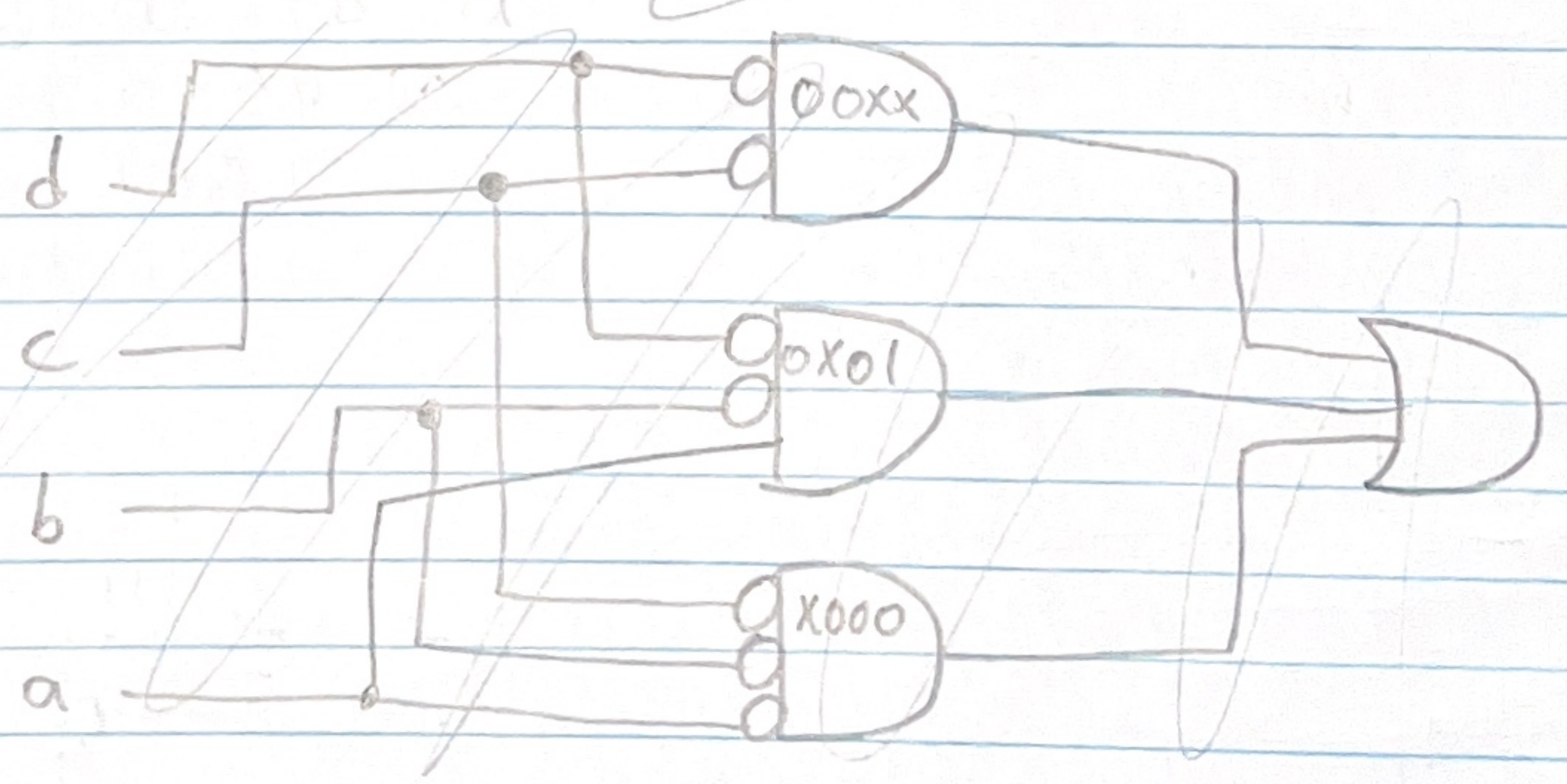
dc \ ba	00	01	11	10
00	1 0	1 1	1 3	1 2
01	0 4	1 5	0 7	0 6
11	X 12	X 13	X 15	X 14
10	1 8	0 9	X 11	X 10

Min cost Cover

00XX, 0X01, X000

min cost cover

00XX, 0X01, 1XX0



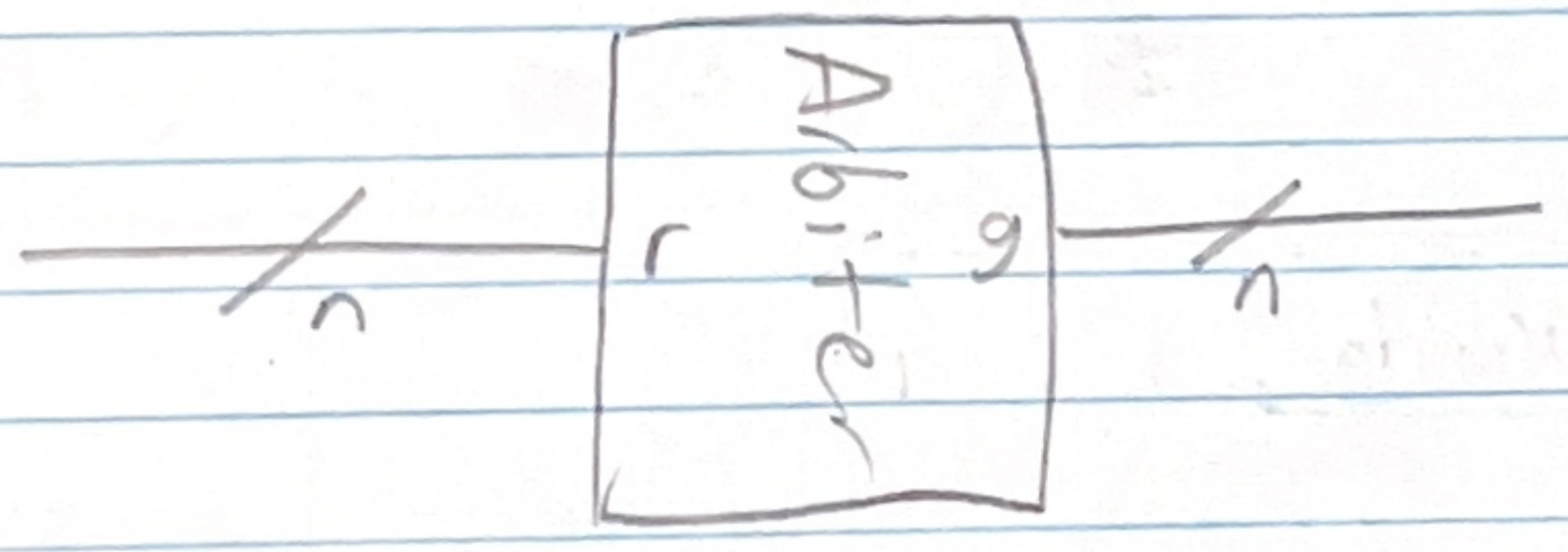
9/19/18
LEC

one hot representation

Binary	One-hot	Base 10
000	001	0
001	010	1
010	100	2

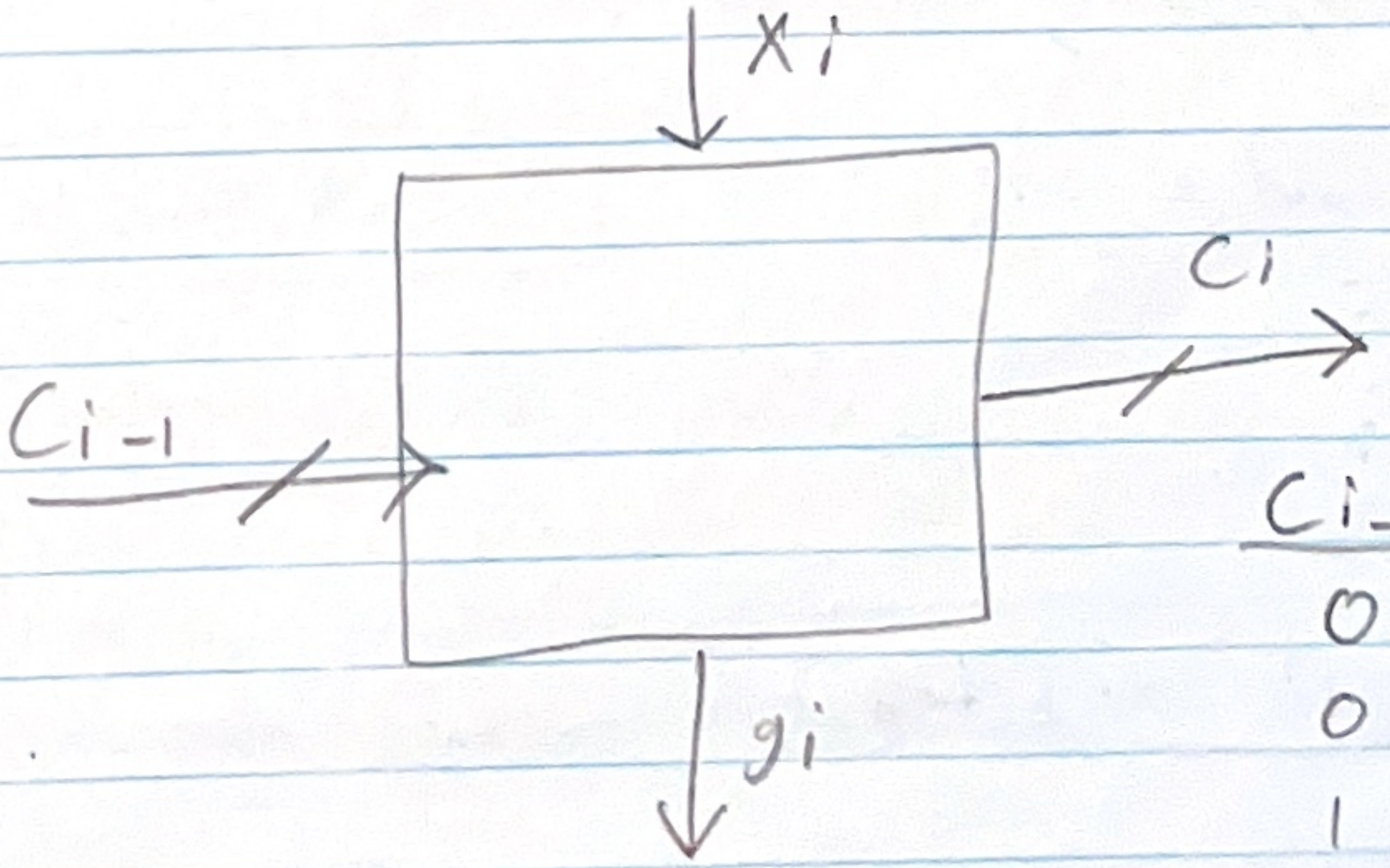
Arbiter

-handles requests from multiple devices



r=request
g=grant

-Finds the first '1' bit in input r



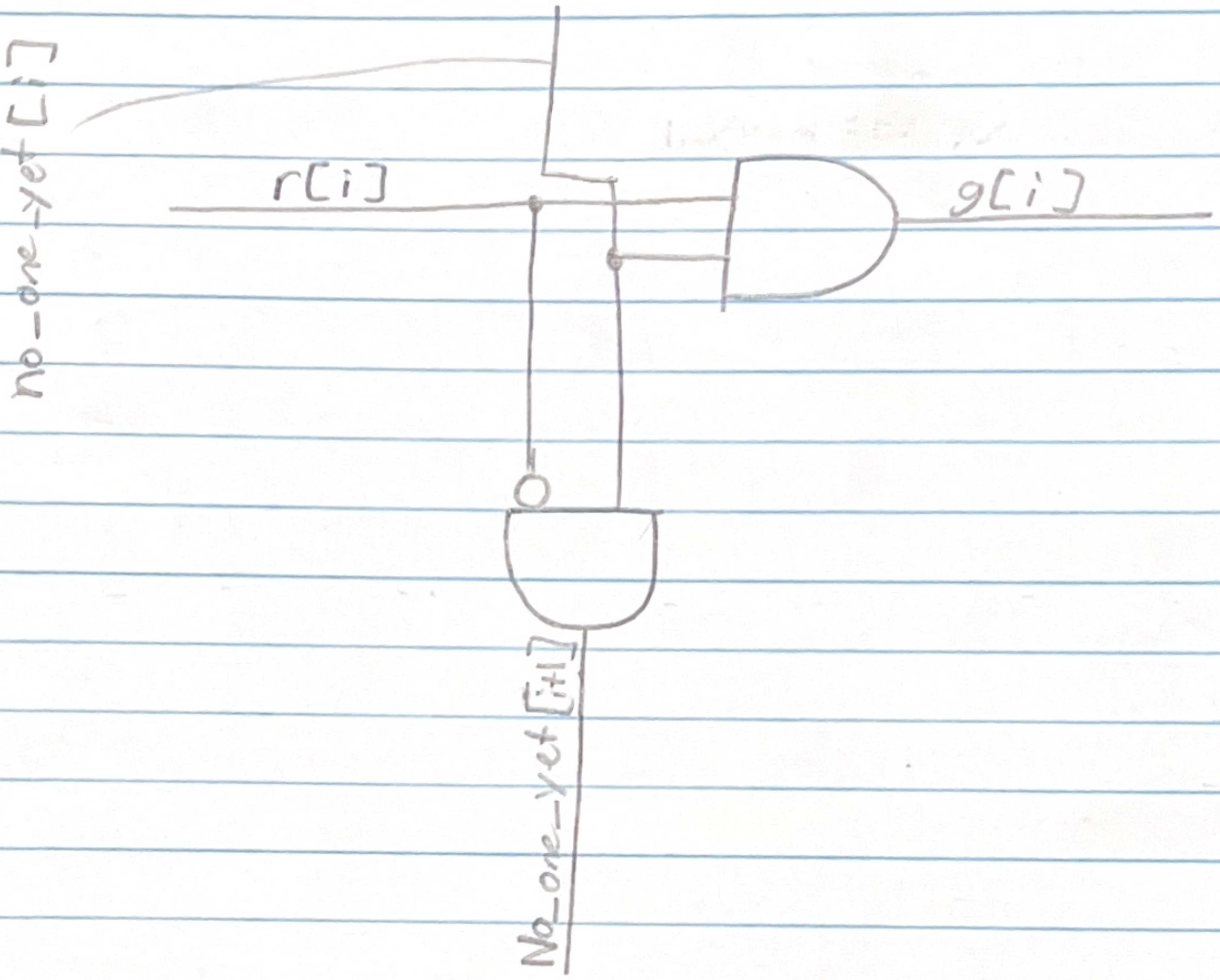
(xi)		(yi)	
C_{i-1}	r_i	C_i	g_i
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

No ones yet	1
seen a one	0

$g_i = C_{i-1} \wedge r_i$

$C_i = C_{i-1} \wedge \overline{r_i}$

Logic Diagram for one bit arbiter



Arbiter in Verilog

```
module  Arbstager (r, cin, g, cout);
   input r, cin;
   output g, cout;

   assign cout = ~r
         ⋮
```

Paramederized arbiter

```
module Arb(r,g);
parameter n;
input [n-1:0] r;
output[n-1:0] g;
```

Testbench Script

- Different syntax than verilog for synthesis in Quartus => gates

- No inputs or outputs

Verilog for synthesis — YES for Testbenches

- Initial
- $display
- repeat and other loops
- #delay ie: #100

} NO!

Debugging

① Check if inputs are correct (Modelsim)
② Check very carefully hardware block inputs go through *if they are good

Top Level Module

- instantiates the other modules
- Tell quartus it is
- Similar to "main()"
- connects to pins on DE1SoC

Always Block

```
always @(sensitivity_List) Begin

        // statements, order matters//

end
```

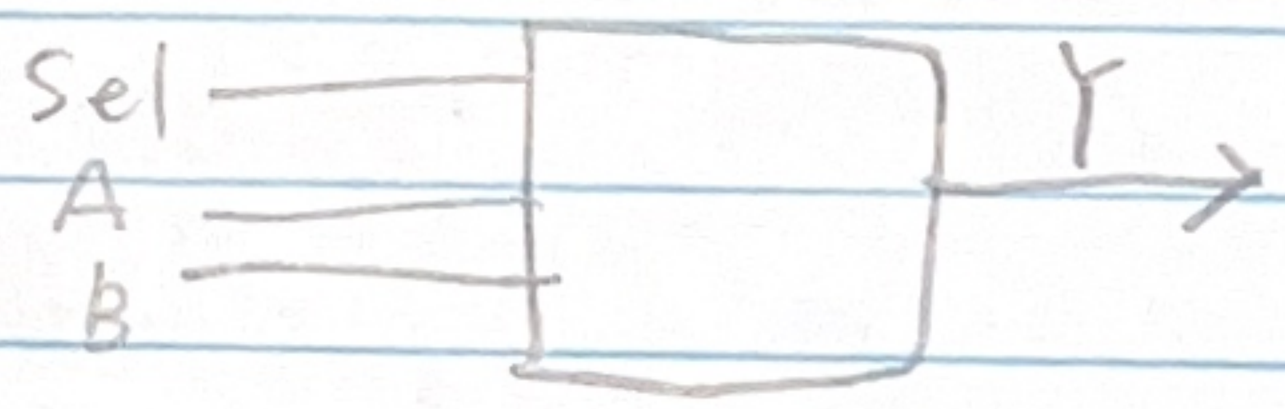
*Don't write everything in a big always block

4 types of always Block for synthesis

1. purely combinational
2. Later
3. Later
4. CPEN 211

Purely Combinational

```
always @(*) begin

  if (sel == 1'b0)
     Y = A;
  else
     Y = B;
end
```



Rules

1. Every input that can affect the outputs must be put in the sensitivity list or (*)
2. Every output must be assigned a value for every possible combination of the inputs

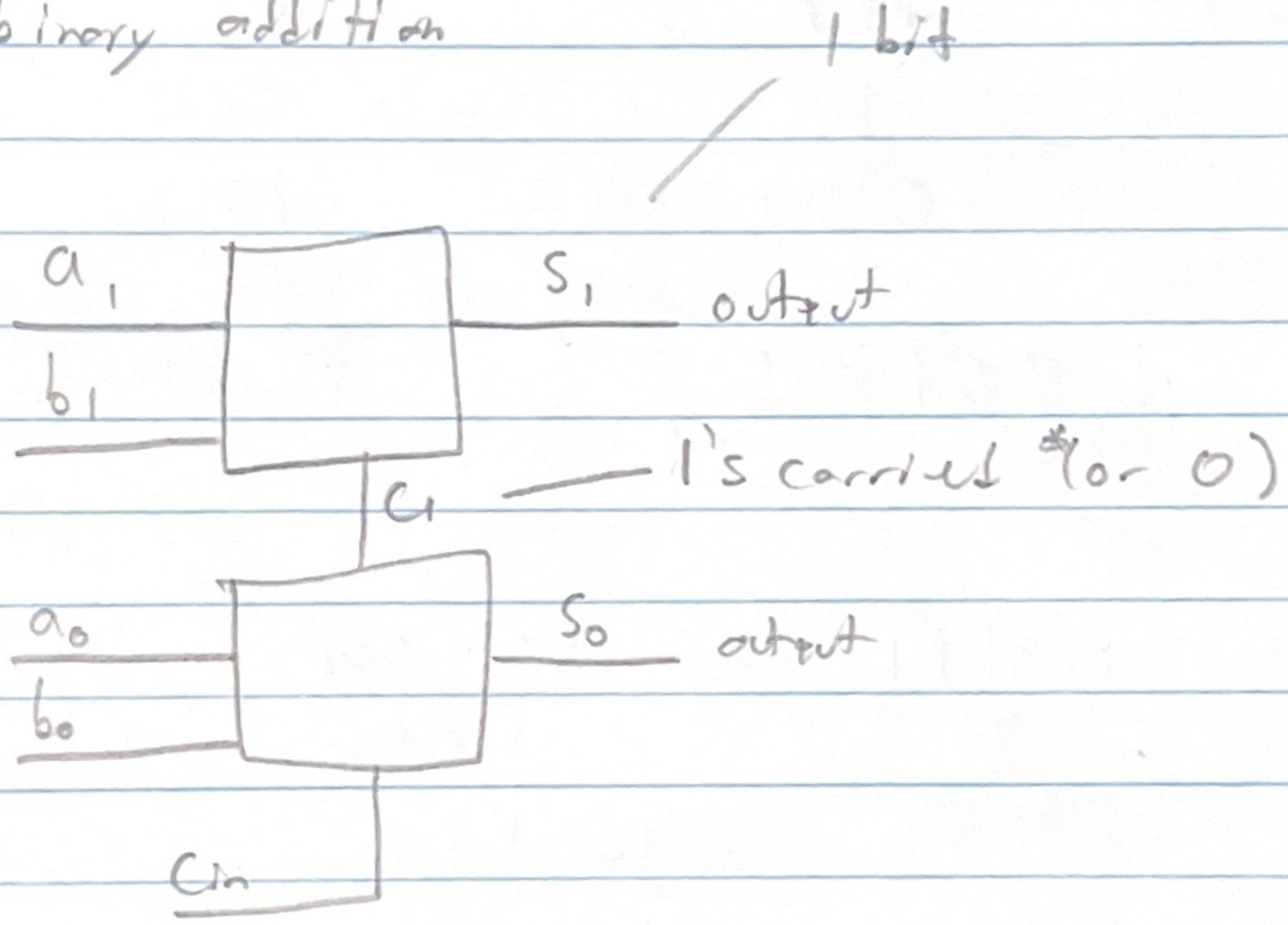
Combinational Logic, Multiplexers, decoders,..

Arbiter output → 1 hot code

encoder takes it and
converts it to binary

Multi bit adder

- Does binary addition



In verilog not using + operator

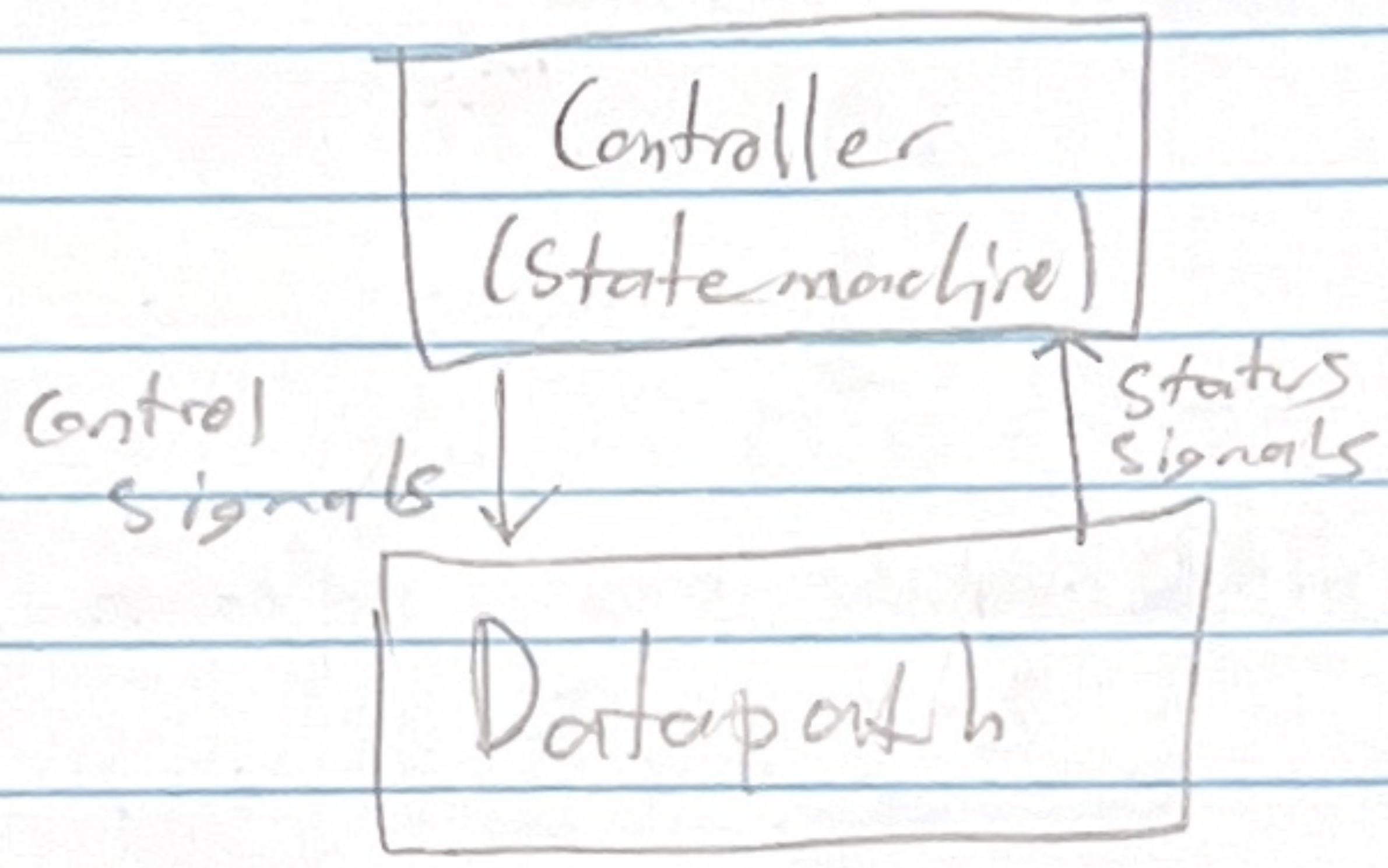
*careful using
don't cares in the controller

Slide Set 7 Datapath State Machines

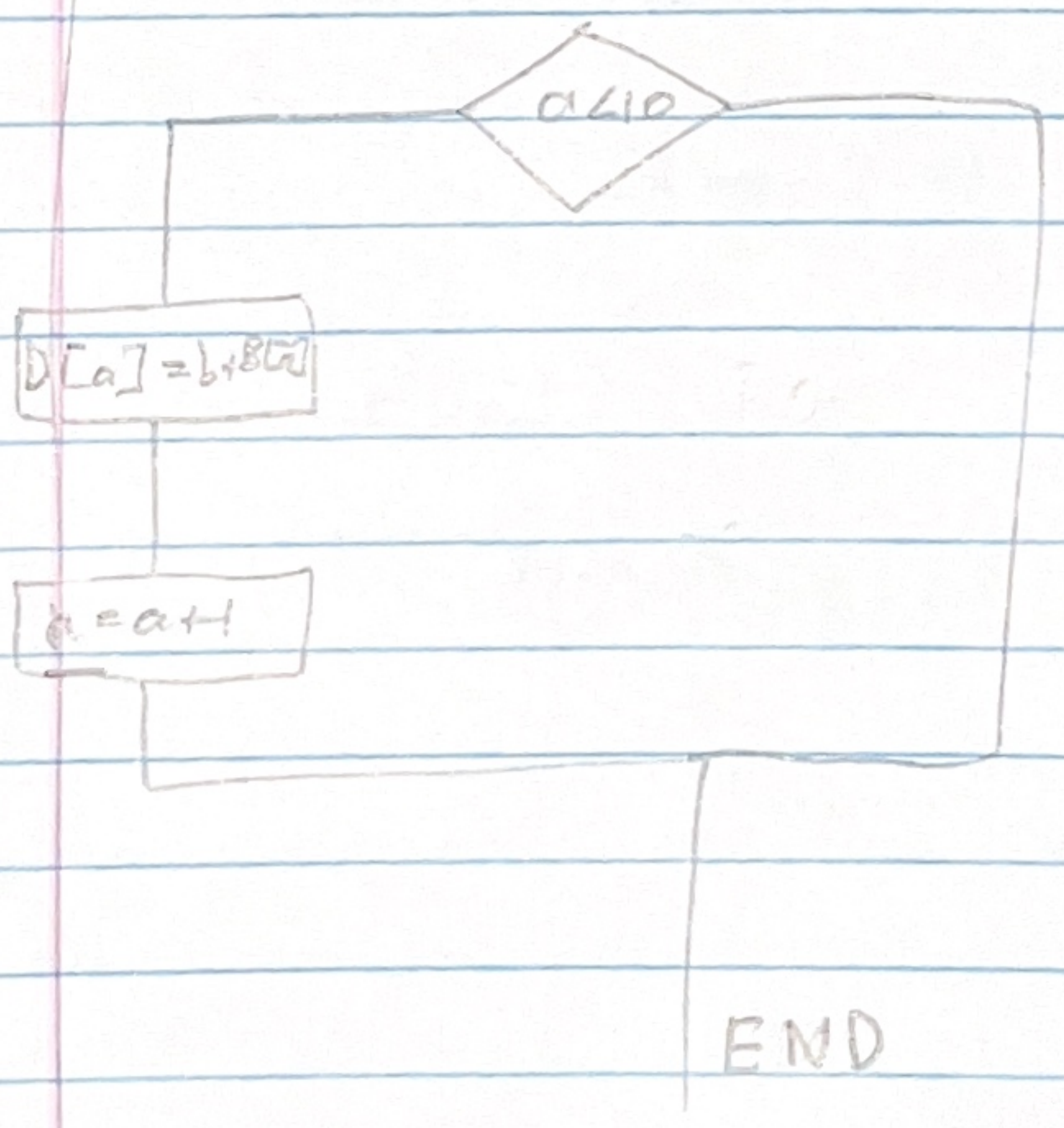
Casex → Can put covers in Casex statements

Datapath state

Clicker C



Flipped #4

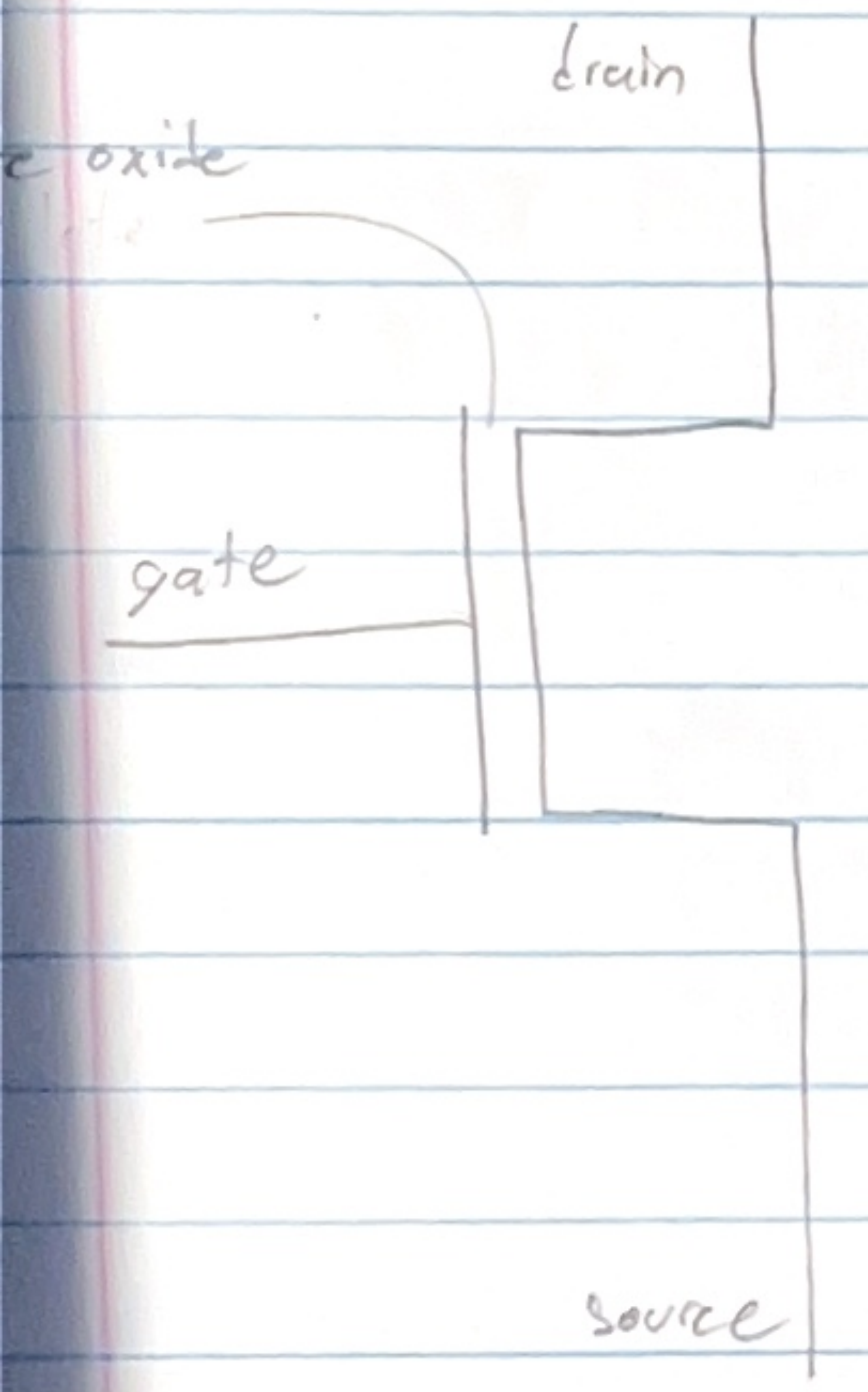


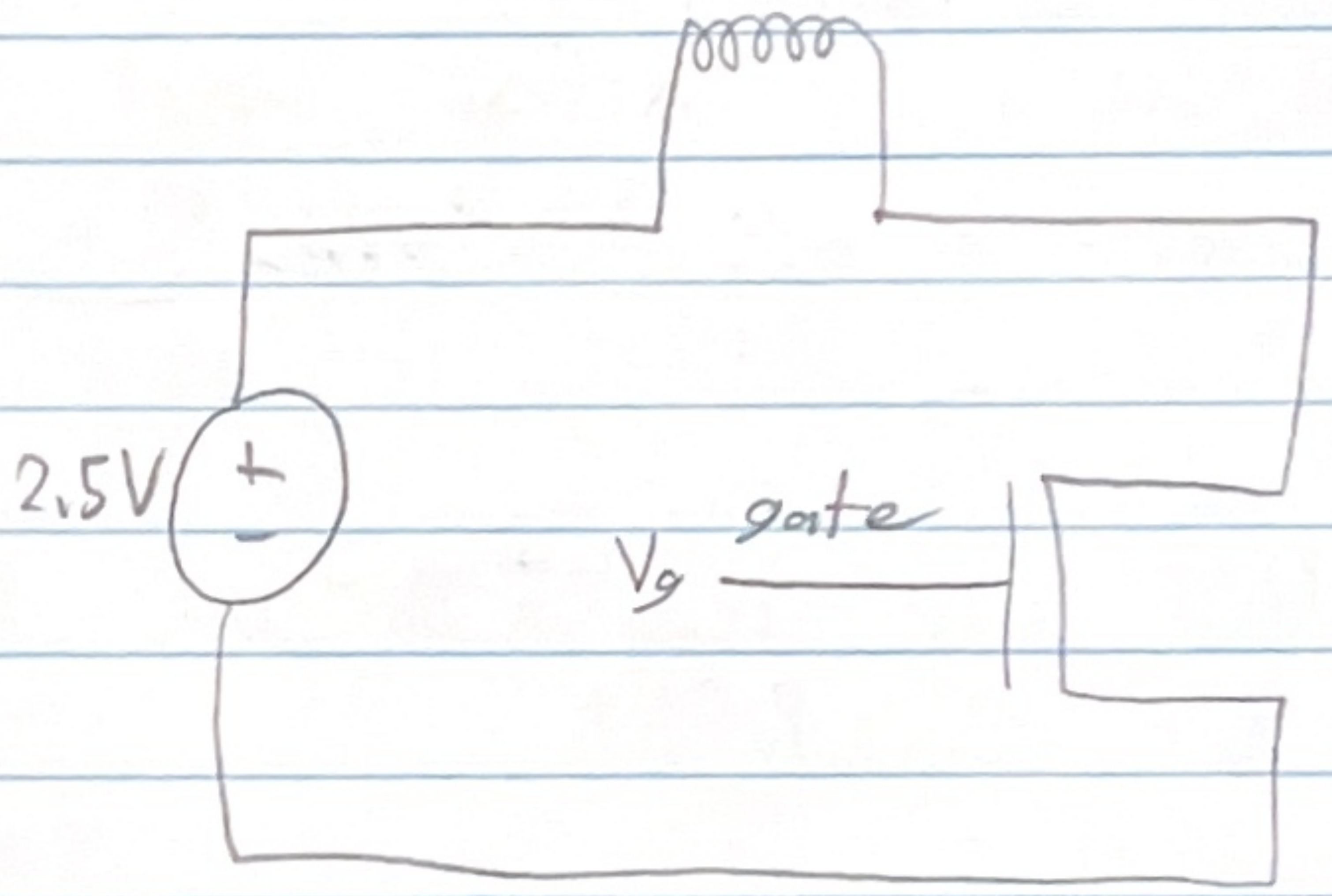
```
LDR r7,[r3,#a]
ADD r
ADD r0,r0,#1

MOV  r2,#0
CMP  r0,r1
BLT  Else
 B   DONE
MOV  r2,#2
```

Slide Set 8 CMOS Logic Gates

- How to build logic gates out of transistors

n-type MOS transistor (NFET)

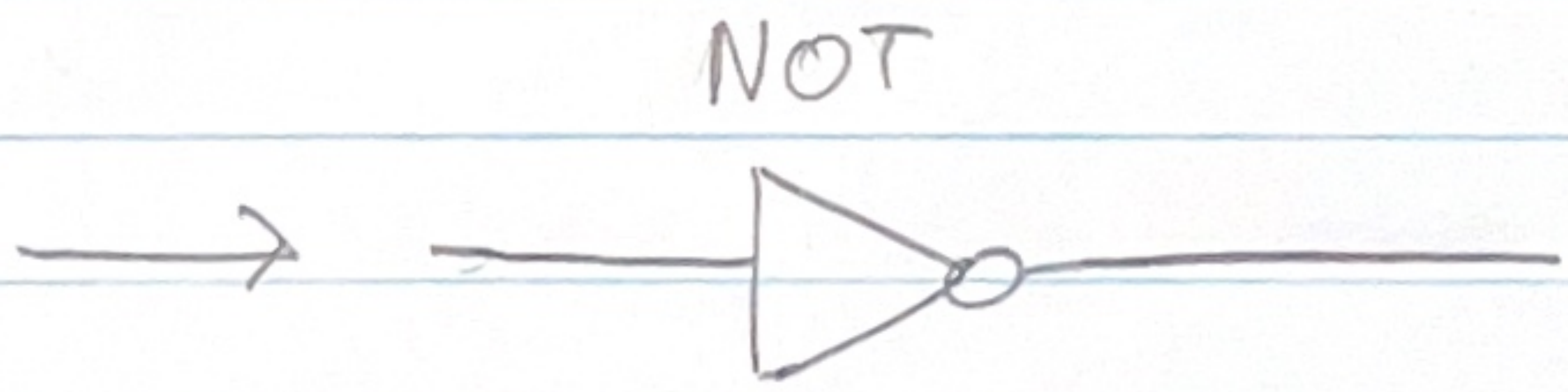




V_g = 0V, light off
V_g = 2.5V, light on

Truth table

Input V	output V
0	1
1	0



PUSH {R0-R7, LR}

STMFD R13!, {R0-R7, R14}

- Store these registers on the Stack
- replaces STR instructions

SUB, LR, #bytes

POP {R0-R7, LR}

- loads these registers from stack
- replaces LDR instructions

ADD, LR, #bytes

Returning from ISR

SUBS pc, lr, #4

Generic Interrupt Controller

- Registers (Not in DRAM)

ICCICR → (1 bit) Should we accept interrupts
ICCPMR → set priority. anyone requesting below this priority isn't allowed
ICCIAR → who is requesting the interrupt
ICCEOIR → end specific interrupt

LP

1

3 c1

↓

$* n-1-\frac{n}{3}$

8

Caches/Virtual memory

Direct Mapped Cache

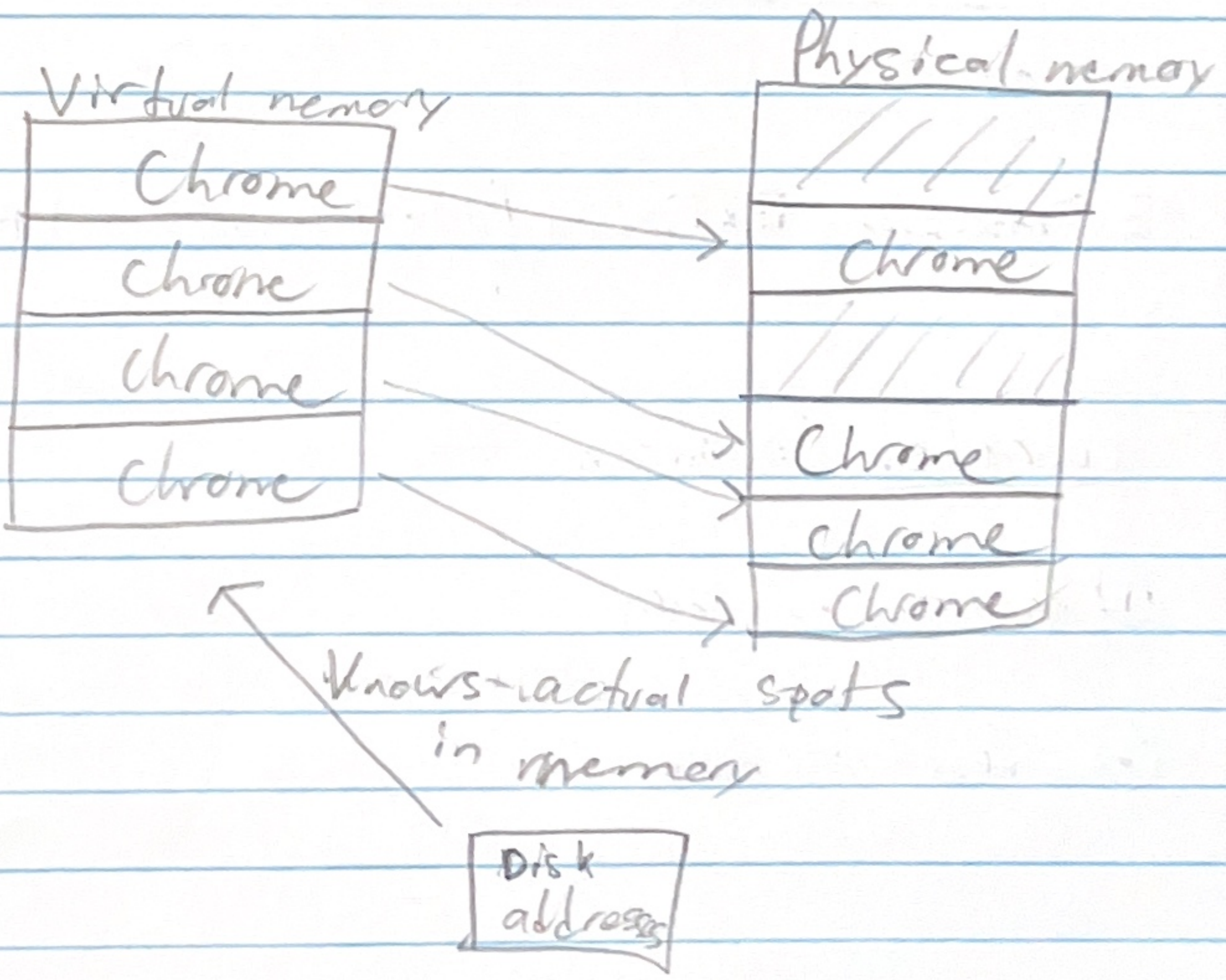
Address → | Block Address | Block offset |

index = (Block address) % (number of blocks)

What to do on a write (STR)?

1. write-through → No write allocate
2. write-back → write allocate

Virtual Memory



Slide Set 14 Parallel Computing

11/29/18
LEC

Problem → Two threads both try to modify the same Global variable at the same time

CPI → Cycles per Instruction

Sol'n → Synchronization

restrict access to threads until thread using that info is done

Pro	Con
Right result	lot of wasted time where threads are blocked

Amdahl's Law

$$ExTime_{new} = ExTime_{old}\left[(1-fraction_{enhanced}) + \frac{fraction_{enhanced}}{Speedup_{enhanced}}\right]$$

Parrallel Communication Models

① message Passing

② shared memory