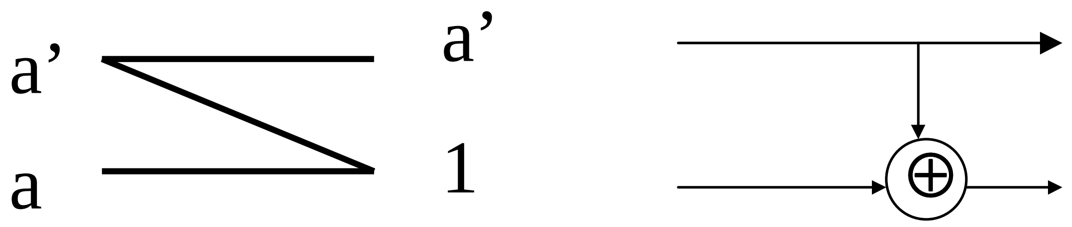
SOLUTION TO FINAL EXAM PROBLEMS

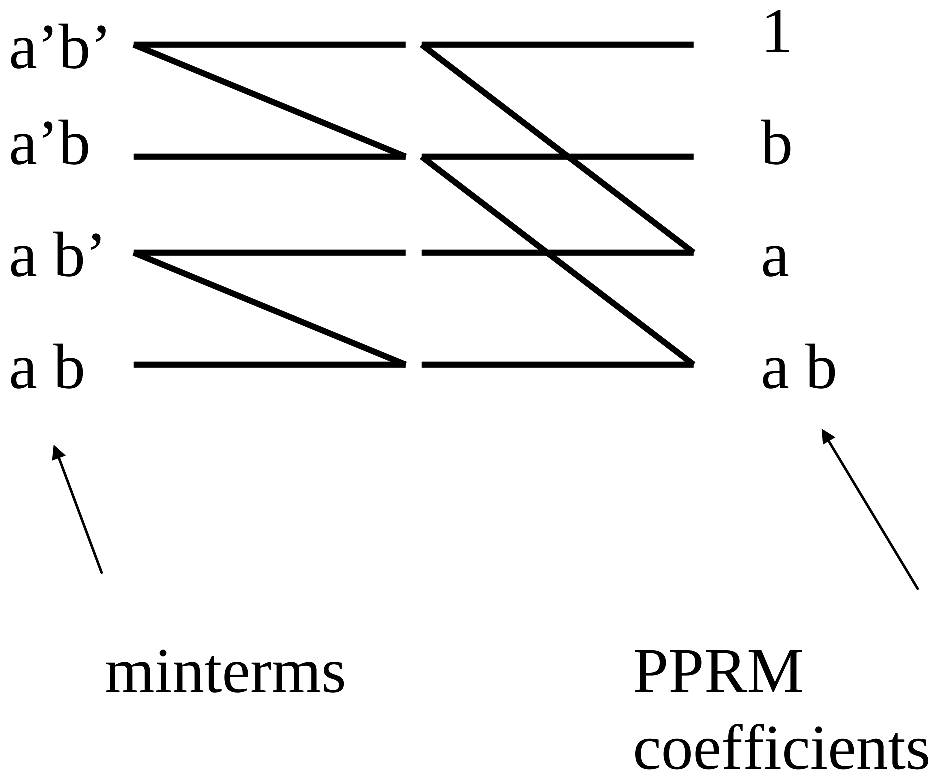
Problem 1. Fast Transforms and Butterflies.

- (a) Draw a kernel of a Fast Reed-Muller Transform. Explain on which formula of Boolean Algebra it is based.
- (b) Illustrate transformation of a symbolic 3 variable Karnaugh Map of Boolean Function to its Positive Polarity Reed-Muller Form using a butterfly based on the RM Kernel from point (a).
- (c) Repeat point (b) for function: f1 = AB + C. Show logic values in all nodes of the graph. Next repeat point (b) for function f2 = A'B'C' ⊕ A where symbol A' means negation of input variable A.
- (d) Draw a purely combinational realization of the Butterfly diagram from point (b).
- (e) draw a pipelined realization of the circuit from point (d), explain in your own words how it works, draw for this pipeline a timing diagram that is typical to illustrate operations of pipelines.

- (a) Draw a kernel of a Fast Reed-Muller Transform. Explain on which formula of Boolean Algebra it is based.

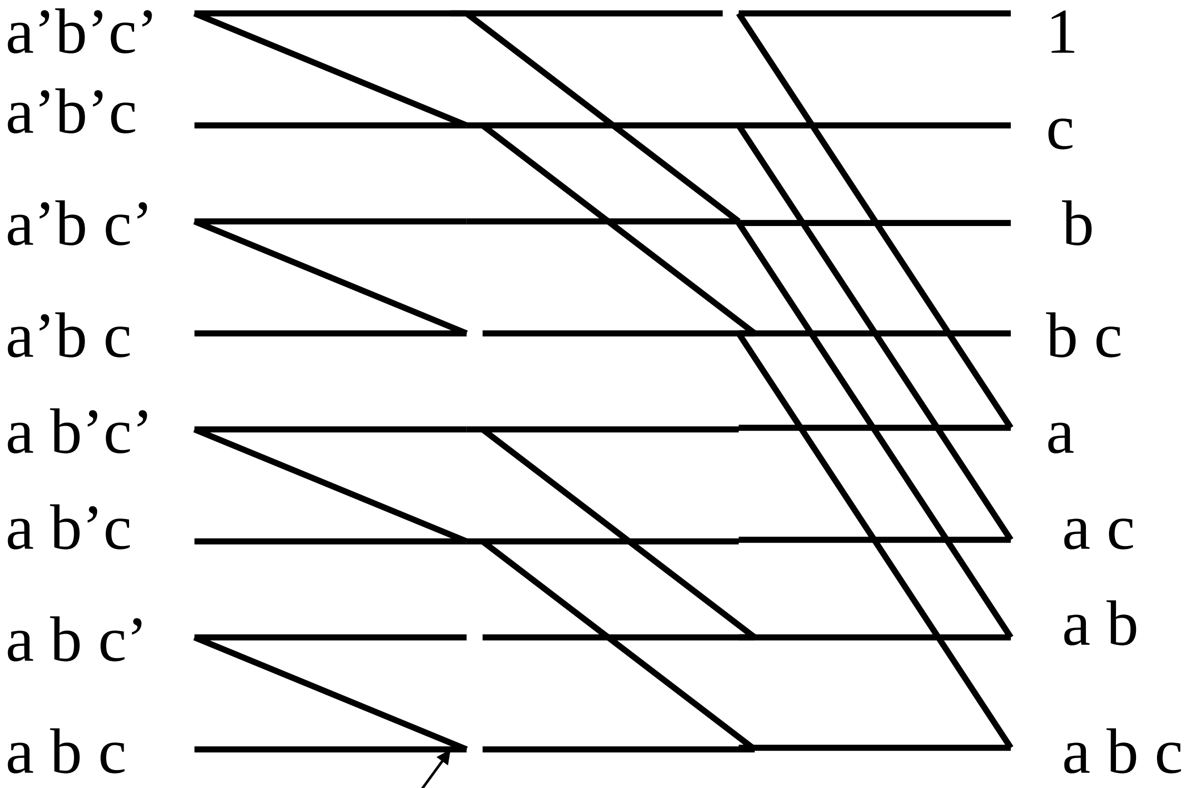


- (b) Illustrate transformation of a symbolic 3 variable Karnaugh Map of Boolean Function to its Positive Polarity Reed-Muller Form using a butterfly based on the RM Kernel from point (a). First we draw for two variables, as in class.



Please verify by yourself

Now, understanding order of variables we can draw for three variables, if in doubt of the order of coefficients, we can always verify - the number of possibilities is not that high so you can find the correct graph quickly.



abc'=ab(1⊕c)=ab ⊕ ac

a'bc'=(1 ⊕ a)b(1⊕c)=b(1 ⊕ a⊕ c⊕ ac)=b ⊕ab ⊕bc ⊕abc

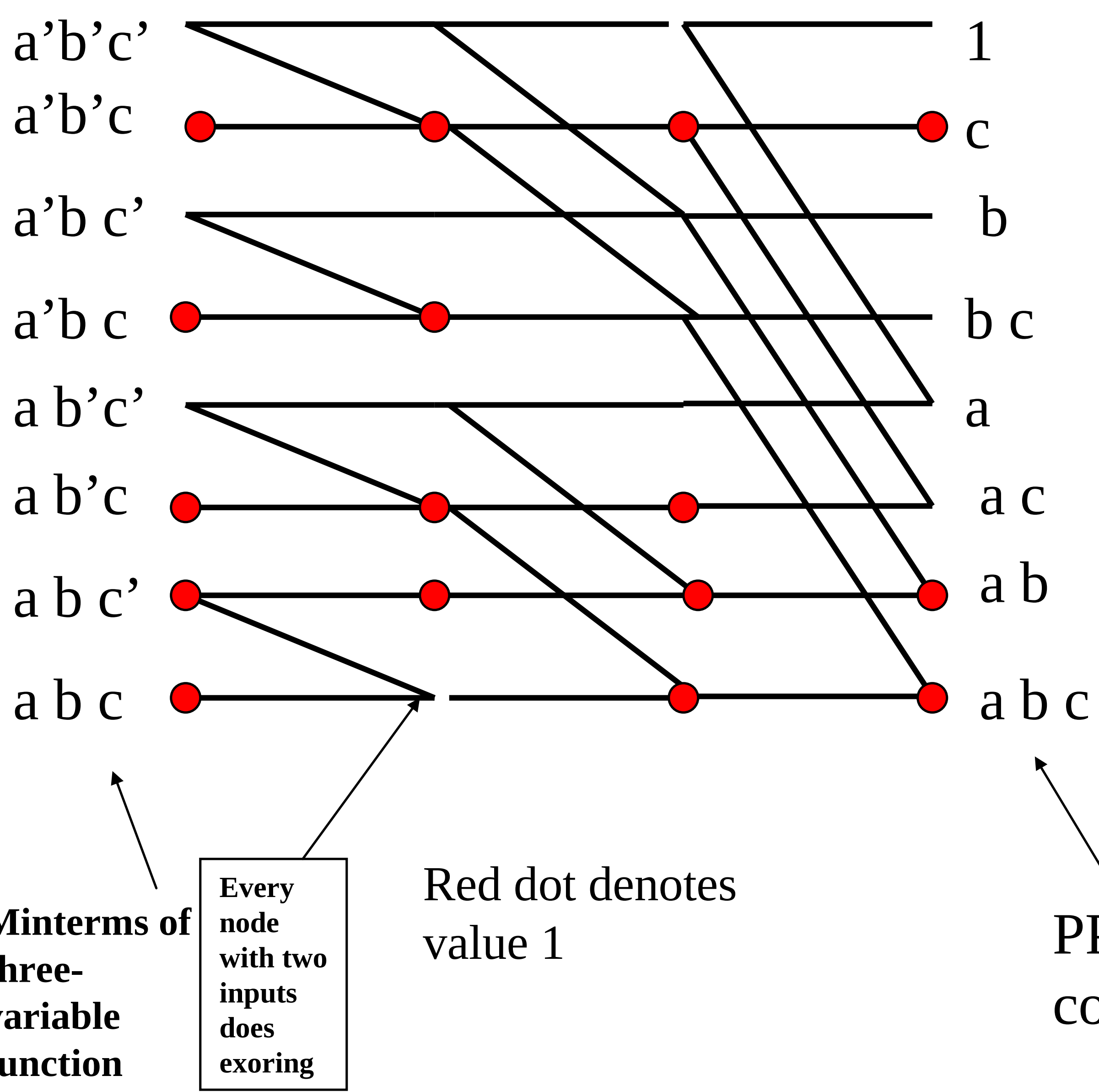
You can write formula like this for every minterm. Think how this formula relates to the graph

Minterms of three-variable function

Every node with two inputs does exoring

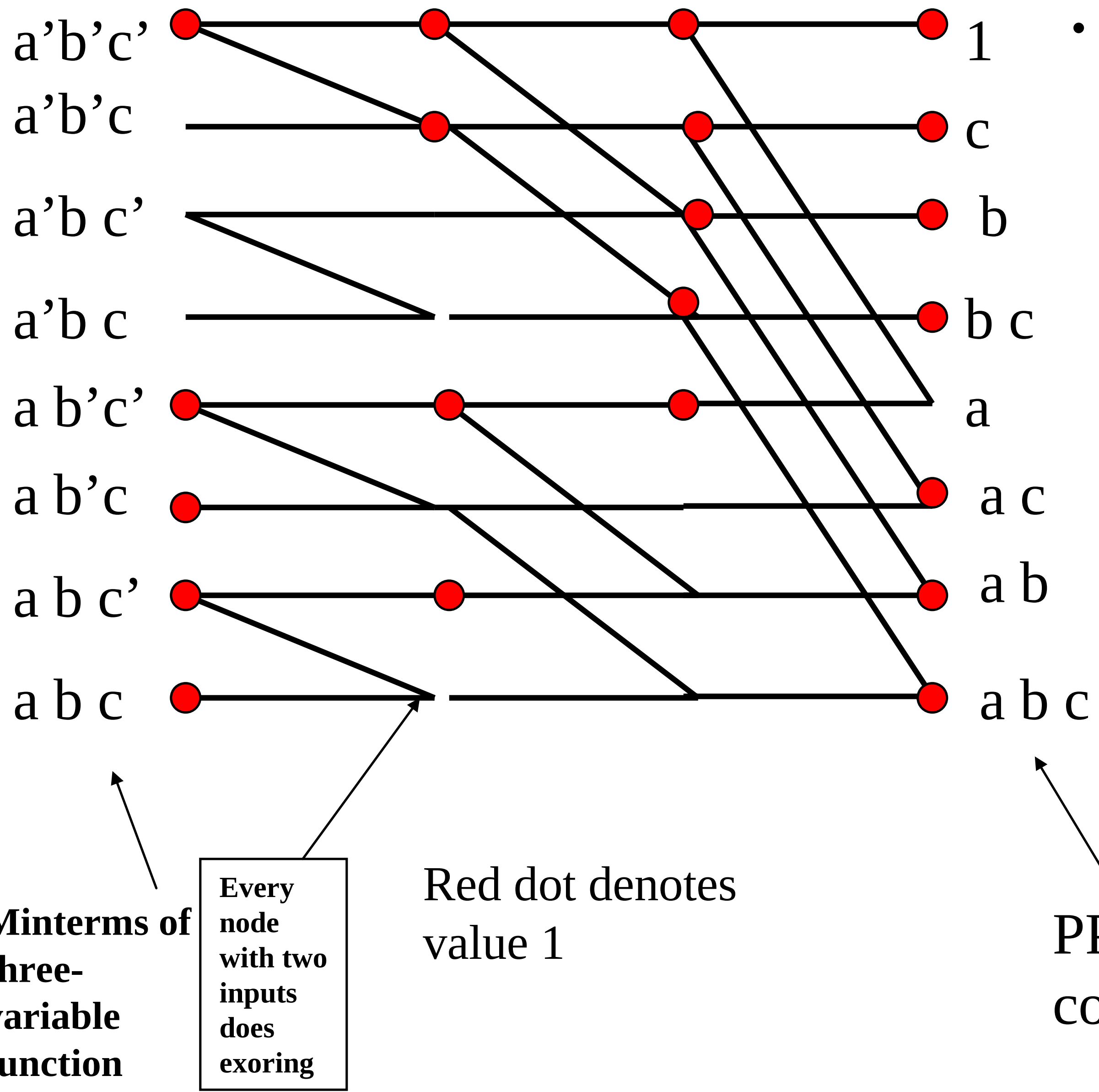
PPRM coefficients

- f1 = AB + C = AB(C+C')+(A+A')(B+B')C=ABC+ABC'+A'BC+AB'C+A'B'C



- Verification from definition
- f1 = AB + C = AB ⊕ ABC ⊕ C

Next repeat point (b) for function f2 = A'B'C' ⊕ A where symbol A' means negation of input variable A.



- Verification from definition
- f2 = A ⊕ A'B'C' = A ⊕ 1 ⊕ A ⊕ B ⊕ C ⊕ AB ⊕ AC ⊕ BC ⊕ ABC

f2 = A ⊕ A'B'C' = 1 ⊕ B ⊕ C ⊕ AB ⊕ AC ⊕ BC ⊕ ABC

- (d) Draw a purely combinational realization of the Butterfly diagram from point (b).

- **Replacing symbols od modulo-2 addition with exor gates in graph we obtain directly the combinational circuit.It can be redrawn to emphasize that each block corresponding to a kernel is the same layout.**

- (e) draw a pipelined realization of the circuit from point (d), explain in your own words how it works, draw for this pipeline a timing diagram that is typical to illustrate operations of pipelines.
- **It is sufficient to insert D type flip-flops after every block.**

Problem 2. Pipeline design, retiming and a controller.

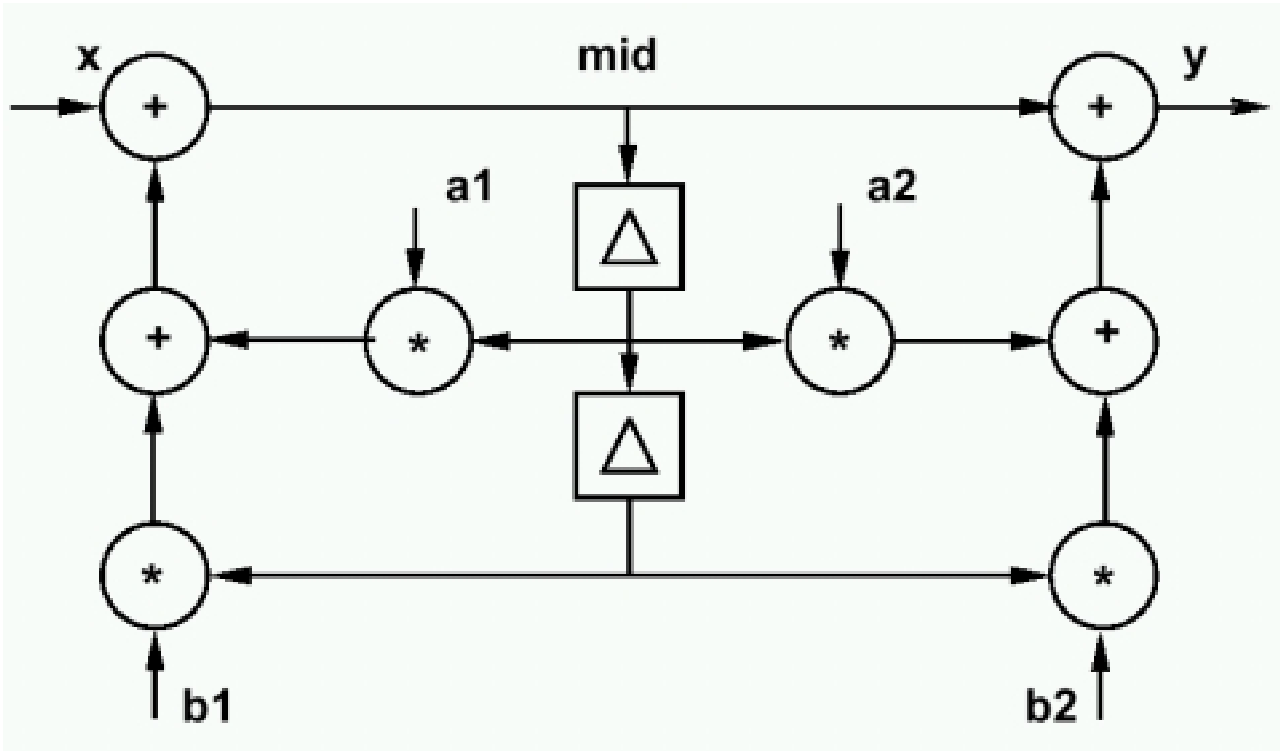
- (a) Start from circuit shown here. Design a pipelined circuit for this graph. Retime if necessary (we started this project in the class and we spent much time!).
- (b) Design the controller FSM for this pipeline. It should be optimized and verified.
- (c)Draw small part of the Timing table for your pipeline to confirm that you understand timing relations in it. Remember that timing can be changed but the timing relations must be preserved

$$Y1=x0+a2x1+b2x2$$

Current input

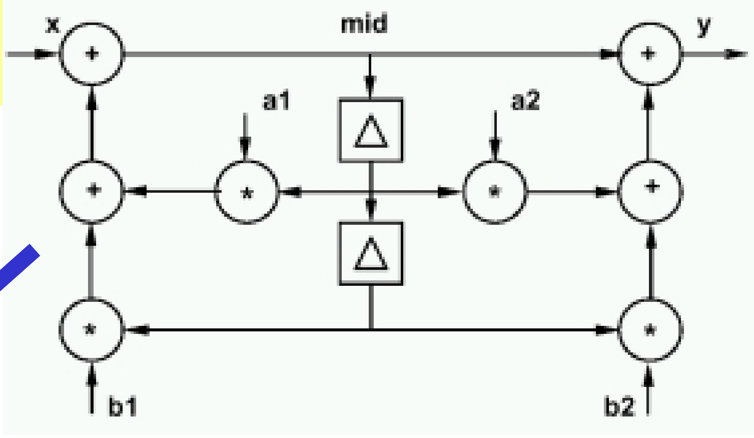
One delayed input (previo us)

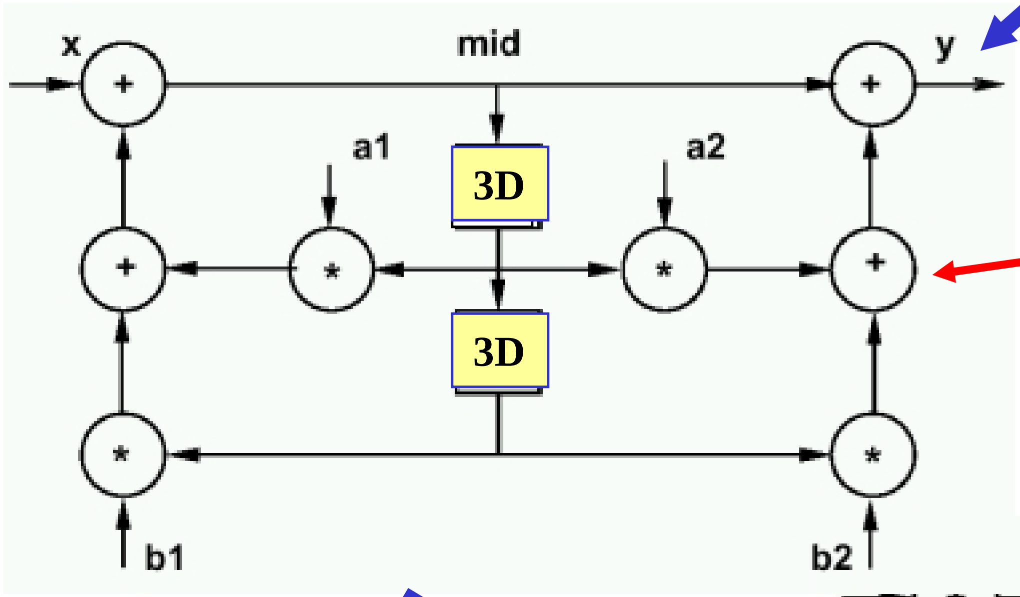
Twice delayed input



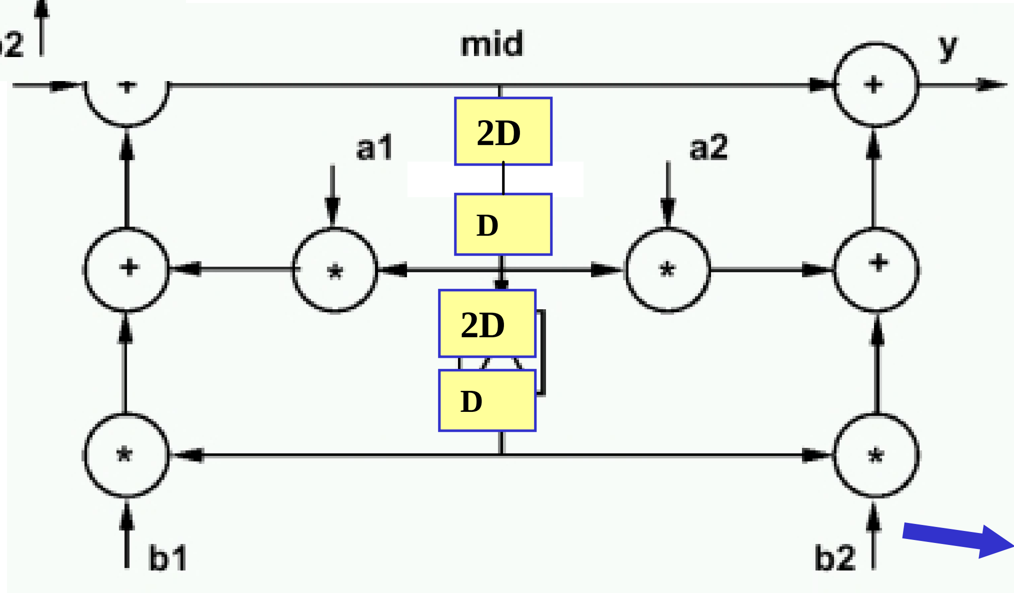
Solution to Problem 2.

- (a) Start from circuit shown here. Design a pipelined circuit for this graph. Retime if necessary.

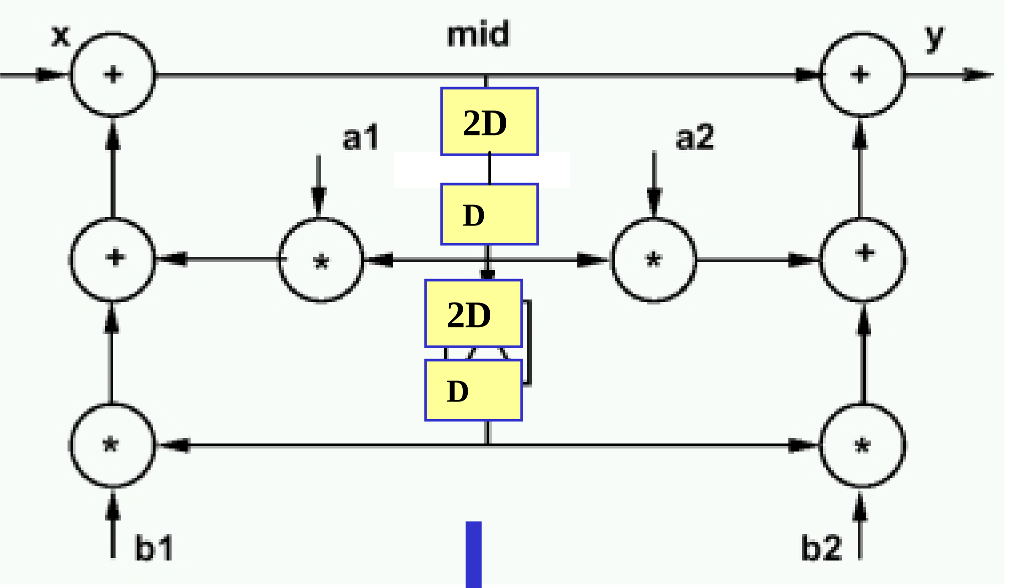


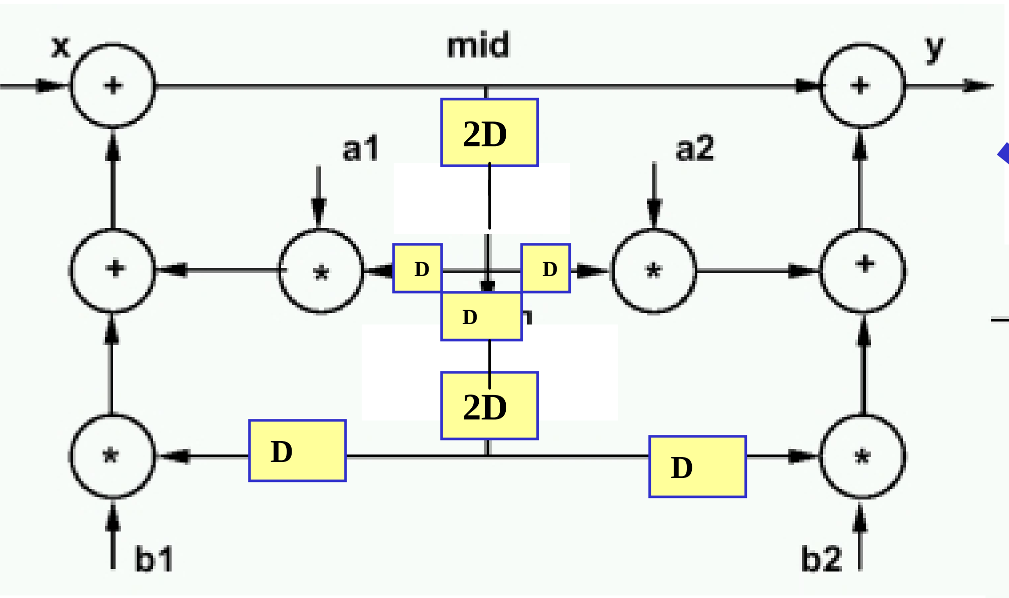


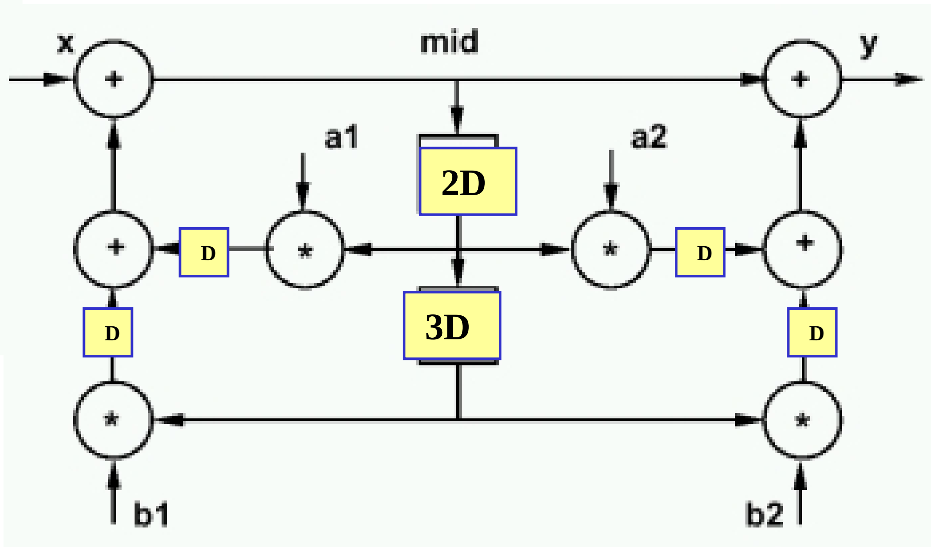
We slow down the circuit three times by replacing D by 3D

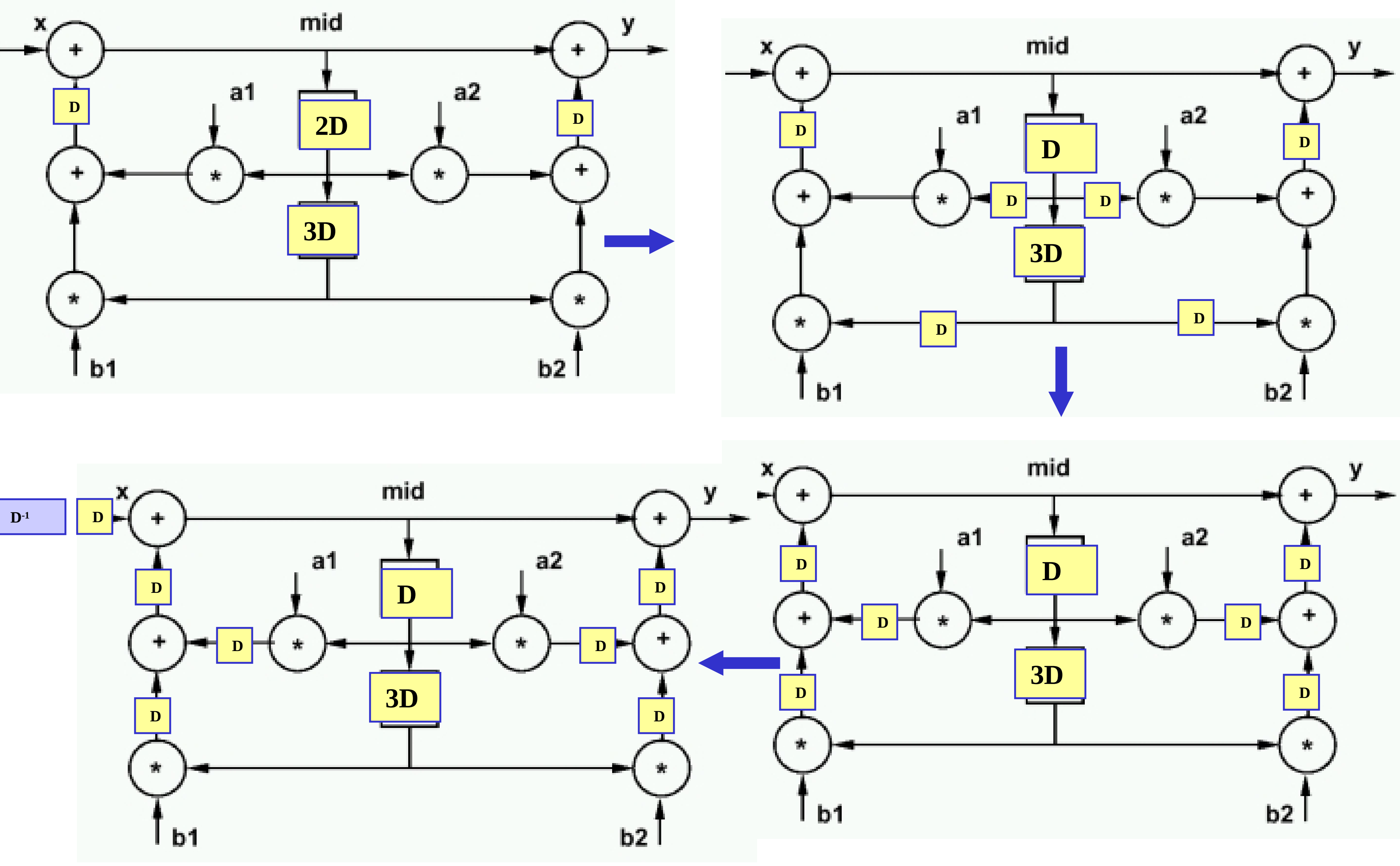


We will be retiming by moving delays through the circuit

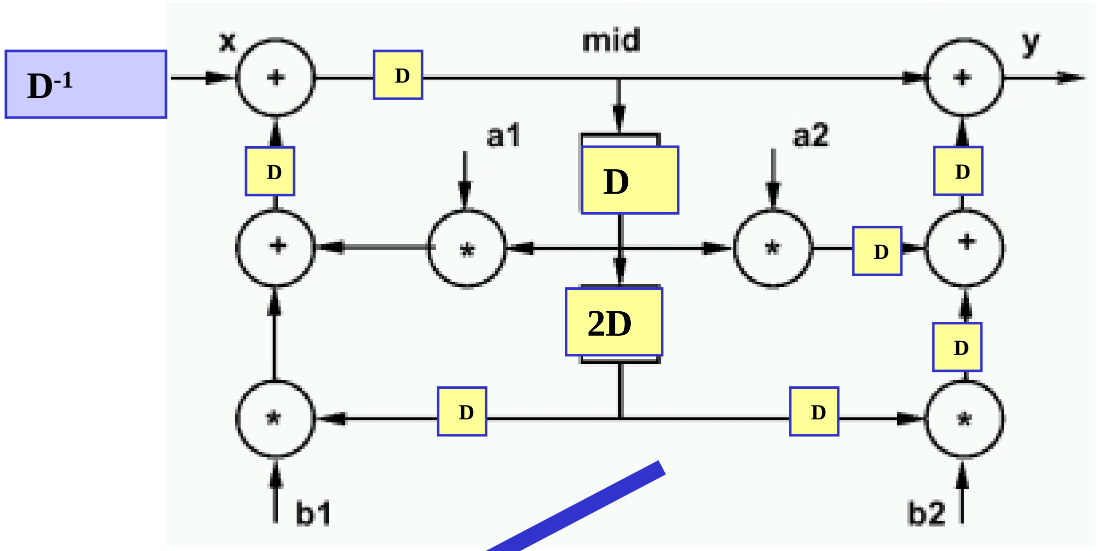

x
mid
y
a1
a2
2D
D
2D
D
b1
b2


x
mid
y
a1
a2
2D
D
D
D
2D
D
D
b1
b2

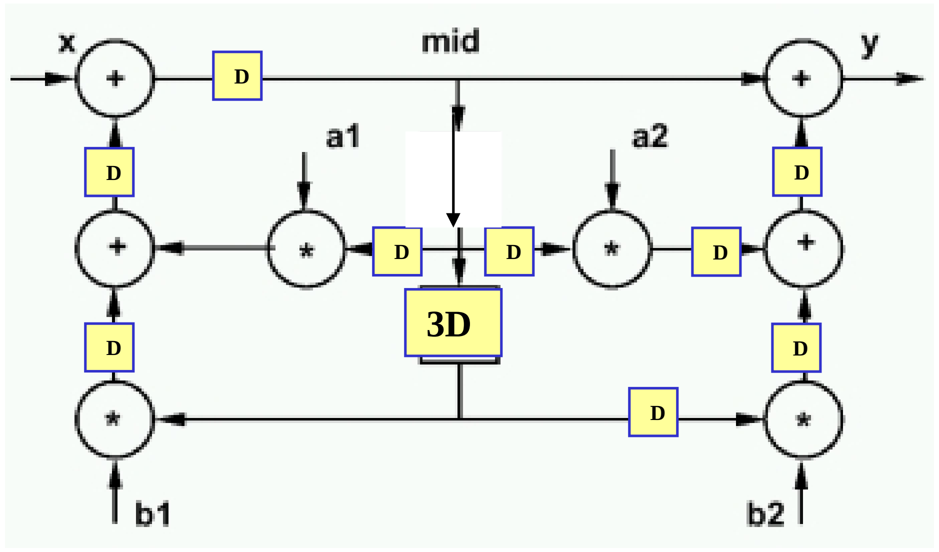

x
mid
y
a1
a2
2D
D
D
D
D
3D
b1
b2



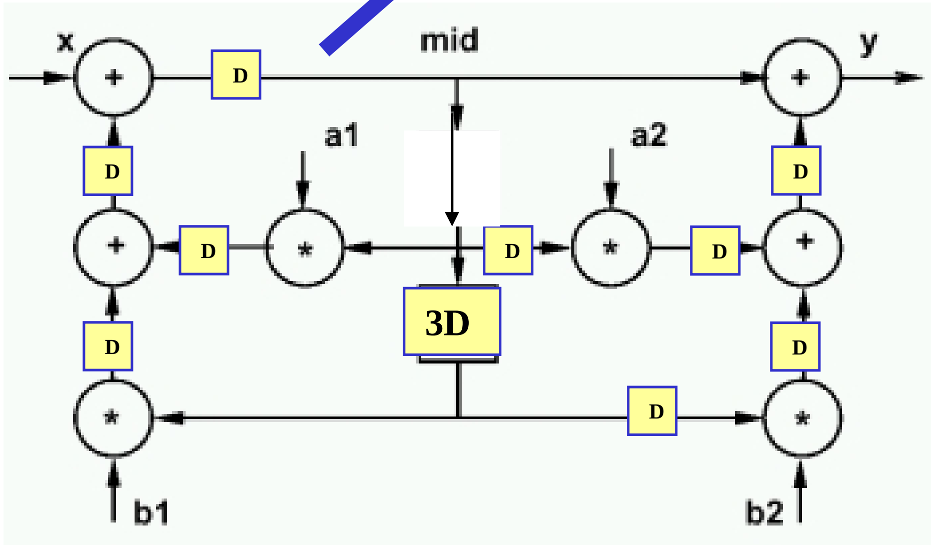
x
mid
y
a1
a2
b1
b2
2D
3D
D
D-1

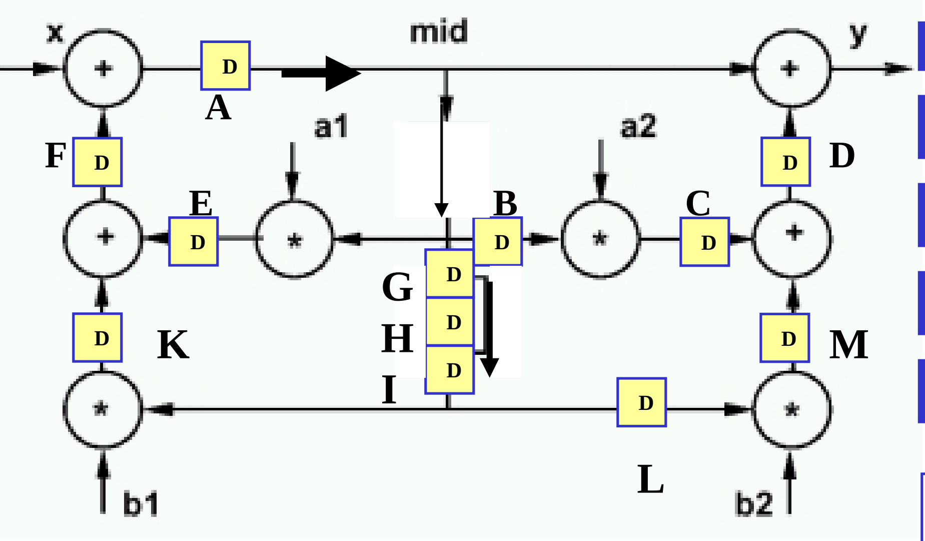


We can omit D^{-1} on input



And finally we have all logic blocks separated by delay registers





Now you can connect the same clock to all registers D and you do not need to design any additional Controller! **The whole trick was just smart retiming!!!!**

You do not have to draw the entire pipeline timing diagram, just show that you understand the principles.

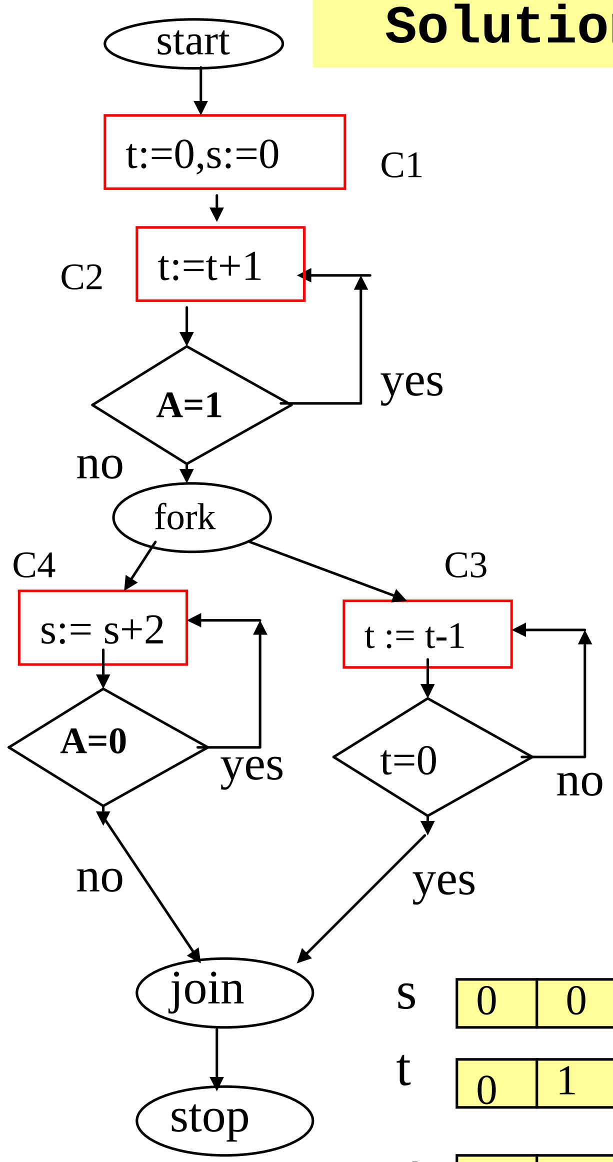
time registers	0	1	2	3	4	5	6	7	8	9
A		A1=X0+F0	A2=X1+F1	A3=X2+F2	A4=X3+F3	A5=X4+F4	A6=X5+F5			
B			B2=A1	B3=A2	B4=A3	B5=A4	B6=A5			
C				C3=a2*B2						
D					D4=C3+M3					
E										
F										
G										
H										
I										
J										
K										
L										
M										
y						y5=D4+A1				

Three delays, OK since we multiplied D by 3

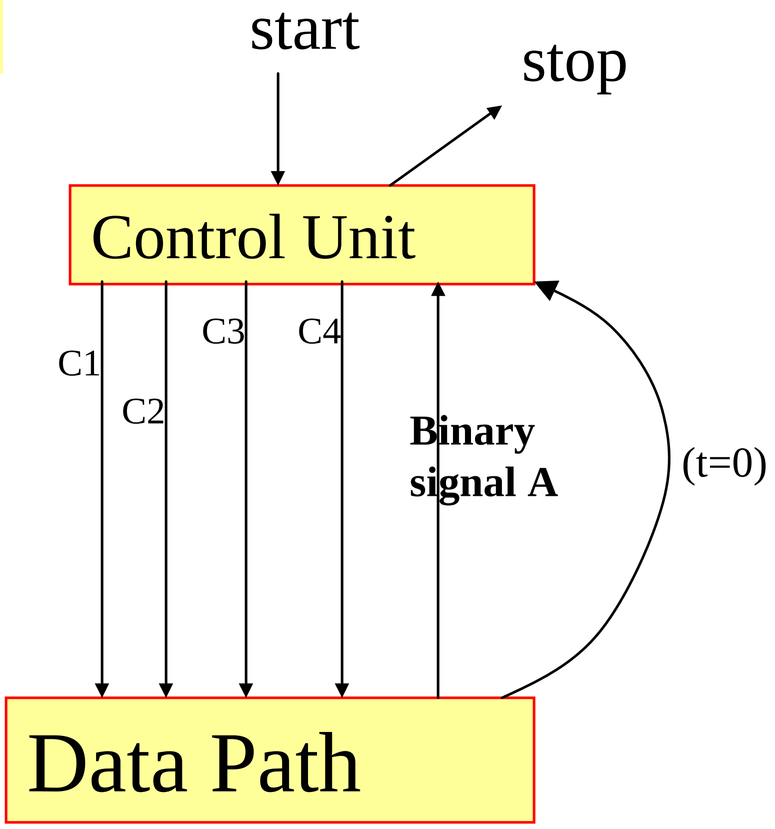
Problem 3. Design of parallel controllers.

- (a) Draw an arbitrary parallel flowchart that has some realistic (not necessarily practical) meaning, using parallel FORK and JOIN nodes. You can use other parallel nodes if you wish.
- (b)Illustrate realization of control using arbitrary method shown in the class.
- (c) Draw schematically a complete system of control unit and data path and analyze its behavior graphically (a timing diagram for data and most important controls) for one set of input data.

Solution to Problem 3

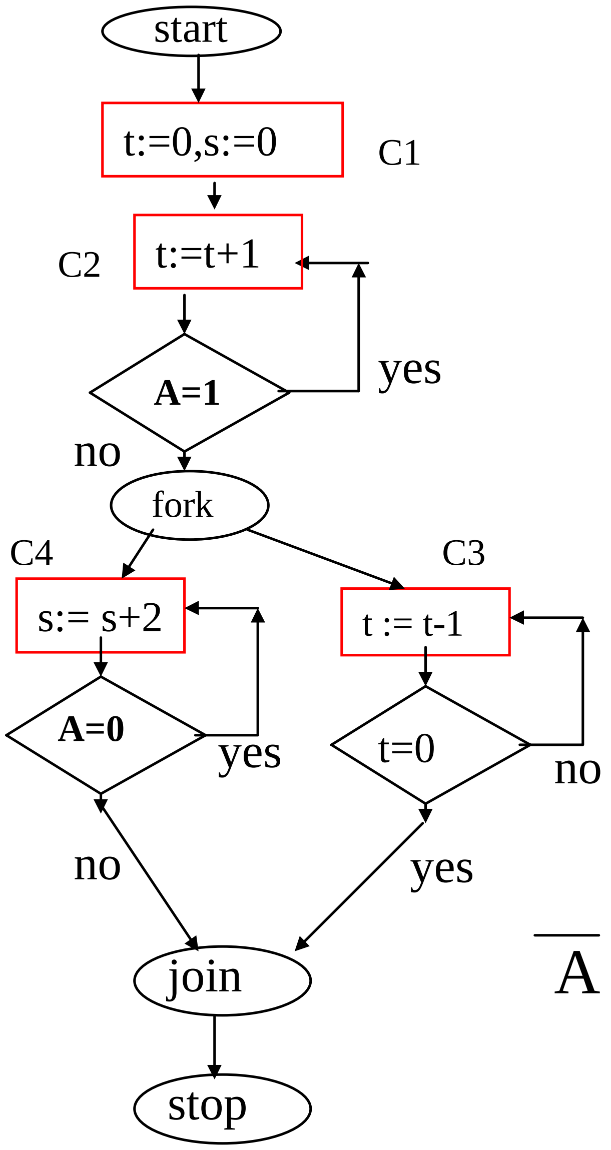


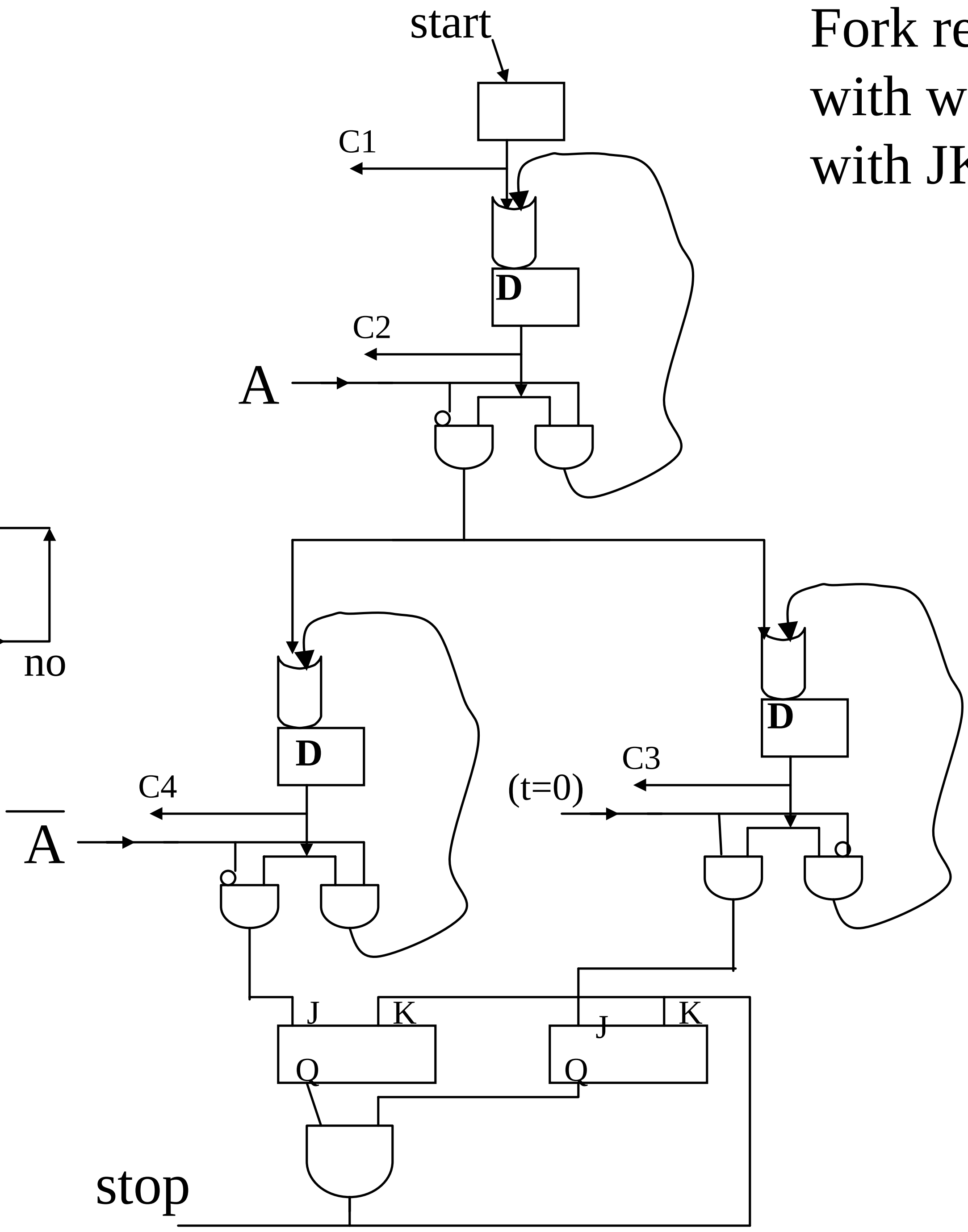
Standard synchronization circuits not shown



s	0	0	0	0	0	2	4	6	8	10	12	12						
t	0	1	2	3	4	3	2	1	0	0	0	0						
A	1	1	1	1	1	0	0	0	0	0	1	1						
	C1	C2	C2	C2	C2	C3,C4	C3,C4	C3,C4	C4	C4		stop						

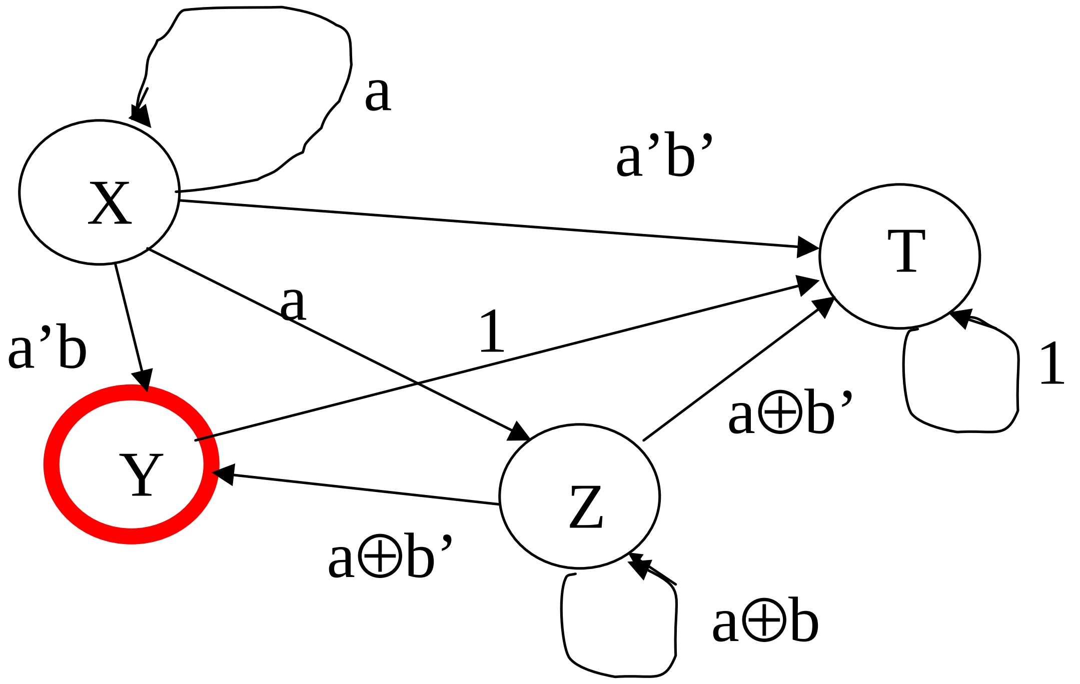
Fork realized with wire, Join with JK FF



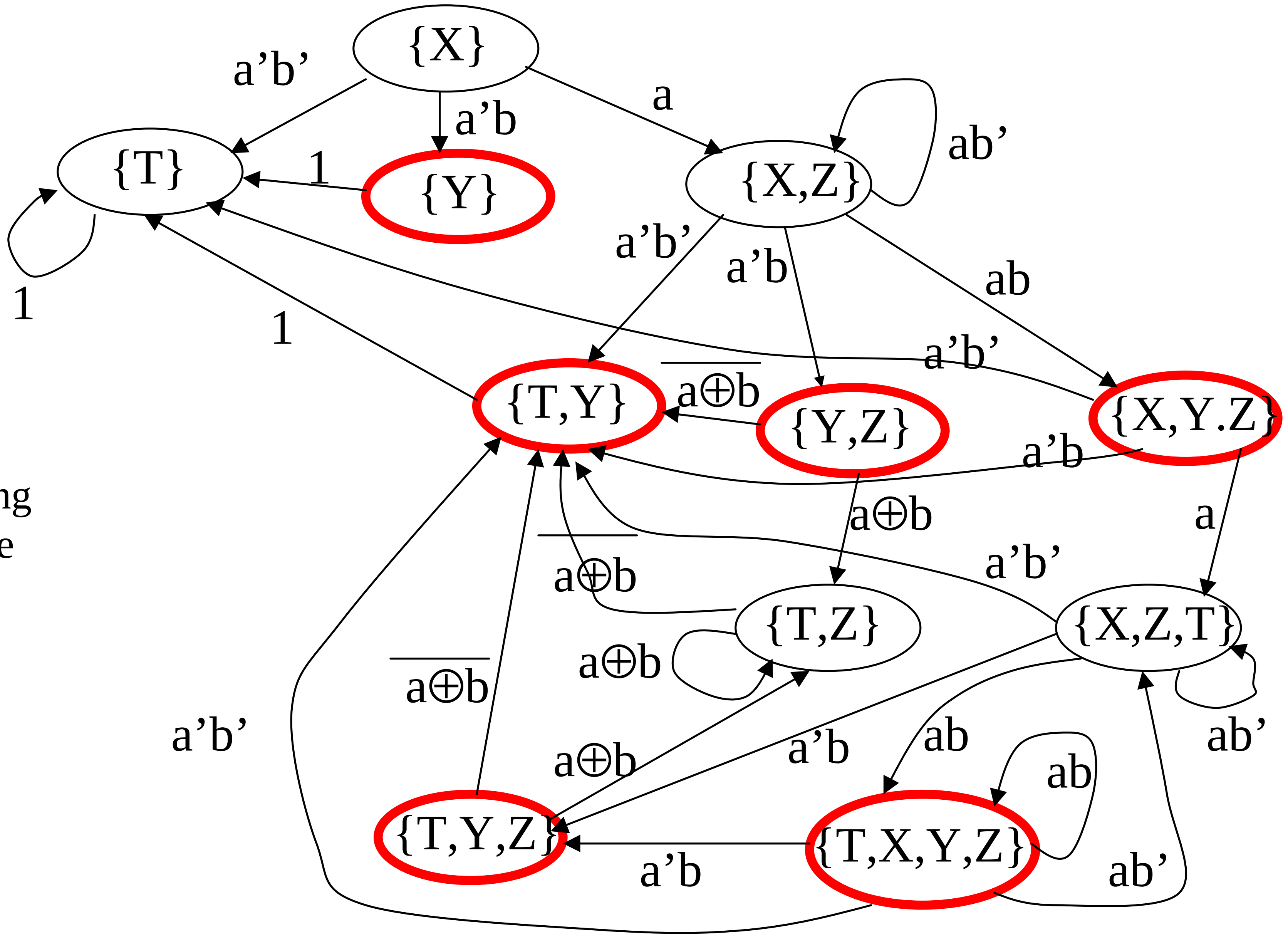


Problem 4. Reachability analysis.

- (a) Convert a non-deterministic FSM from Figure to an equivalent deterministic FSM using reachability analysis.Each state has its specific output with the same name, for instance being in state X is signalized by output X =1. Y is the accepting state.
- (b) Draw the schematics of the non-deterministic machine using D flip-flops
- (c) Draw the schematics of the deterministic machine using D flip-flops. Do not try to minimize the number of flip-flops
- (d) Can it be the same schematics? Can you explain why “yes” or why “not”?



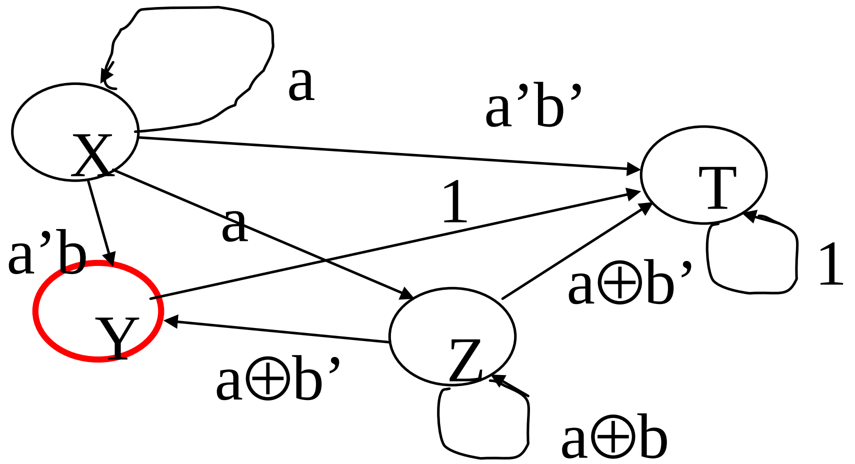
Solution to Problem 4.(a)

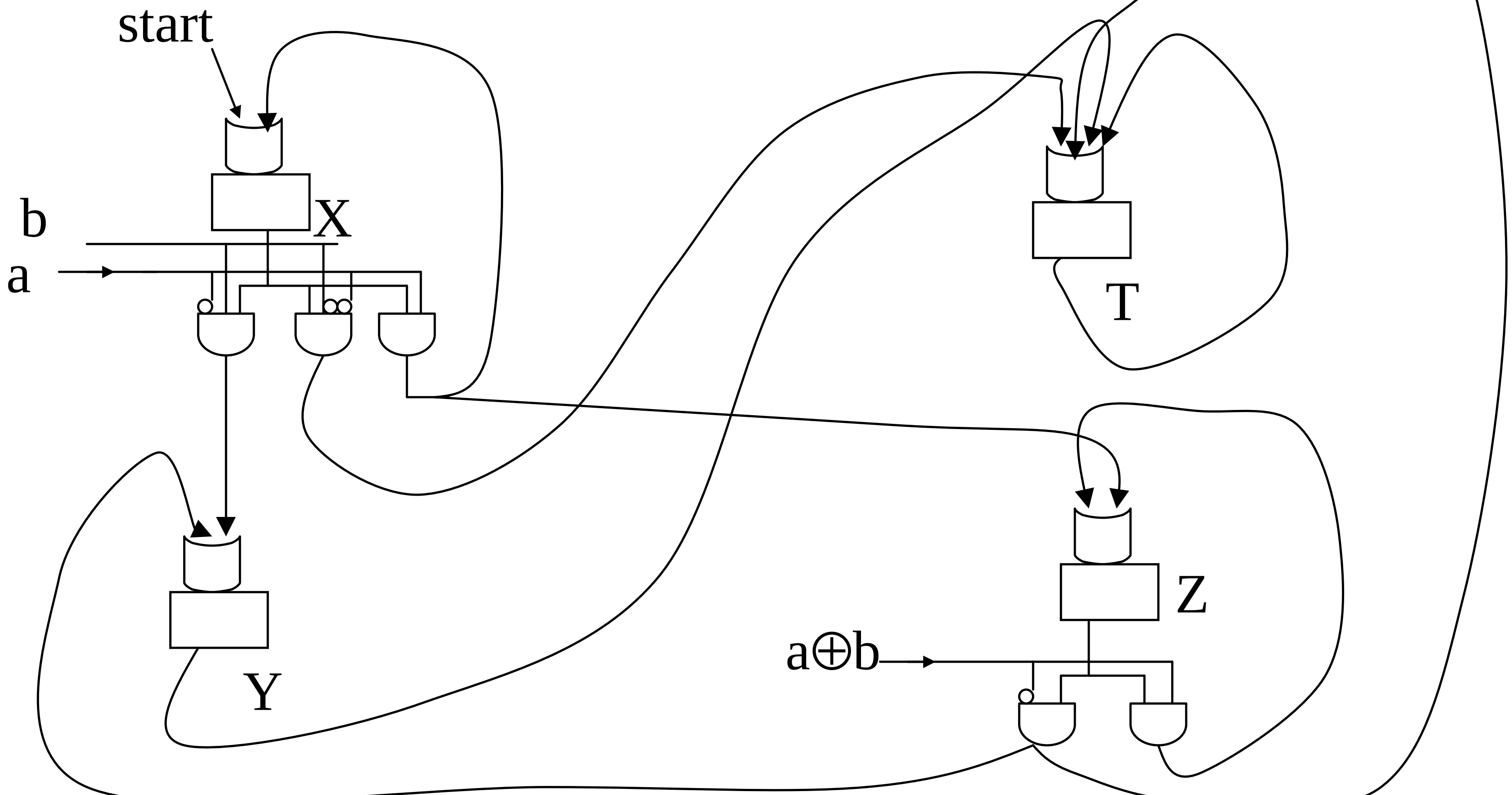


Accepting states are read

- (b) Draw the schematics of the non-deterministic machine using D flip-flops
- (c) Draw the schematics of the deterministic machine using D flip-flops. Do not try to minimize the number of flip-flops
- (d) Can it be the same schematics? Can you explain why "yes" or why "not"?

As an answer to points (b), (c), and (d) let us observe that if one uses one-hot coding of non-deterministic machine then the schematic realizes both the non-deterministic machine and equivalent deterministic machine since the conversion is done automatically by this synthesis method. It results directly from the operation of gates and the reachability analysis.


a
a’b’
X
T
1
a
a’b
1
Y
a⊕b’
Z
a⊕b’
a⊕b


start
b
X
a
T
Y
Z
a⊕b

Problem 5. Regular expressions.

- (a) Write a regular expressions of language **L** for the following event: *An even number of symbols **c** following symbol **b** or a divisible by three number of symbols **c** followed by an odd number of symbols **b**.*
- (b) Draw a graph of this regular expression
- (c)Convert the graph to a non-deterministic machine
- (d) Convert the non-deterministic machine to an equivalent deterministic machine.
- (e) Verification: For every possible sequence of letters b and c of length not larger than 3 ***analyze*** if it belongs to language L:
  - in the regular expression,
  - in the non-deterministic machine,
  - and in the deterministic machine.
- If it does not, what does it mean?

Solution to Problem 5.

- (a) Write a regular expressions of language **L** for the following event: *An even number of symbols **c** following symbol **b** or a divisible by three number of symbols **c** followed by an odd number of symbols **b**.*

$b(cc)^*$

Regular expression for "even number of symbols c following symbol b." We assume zero to be an even number.

$(ccc)^*b(bb)^*$

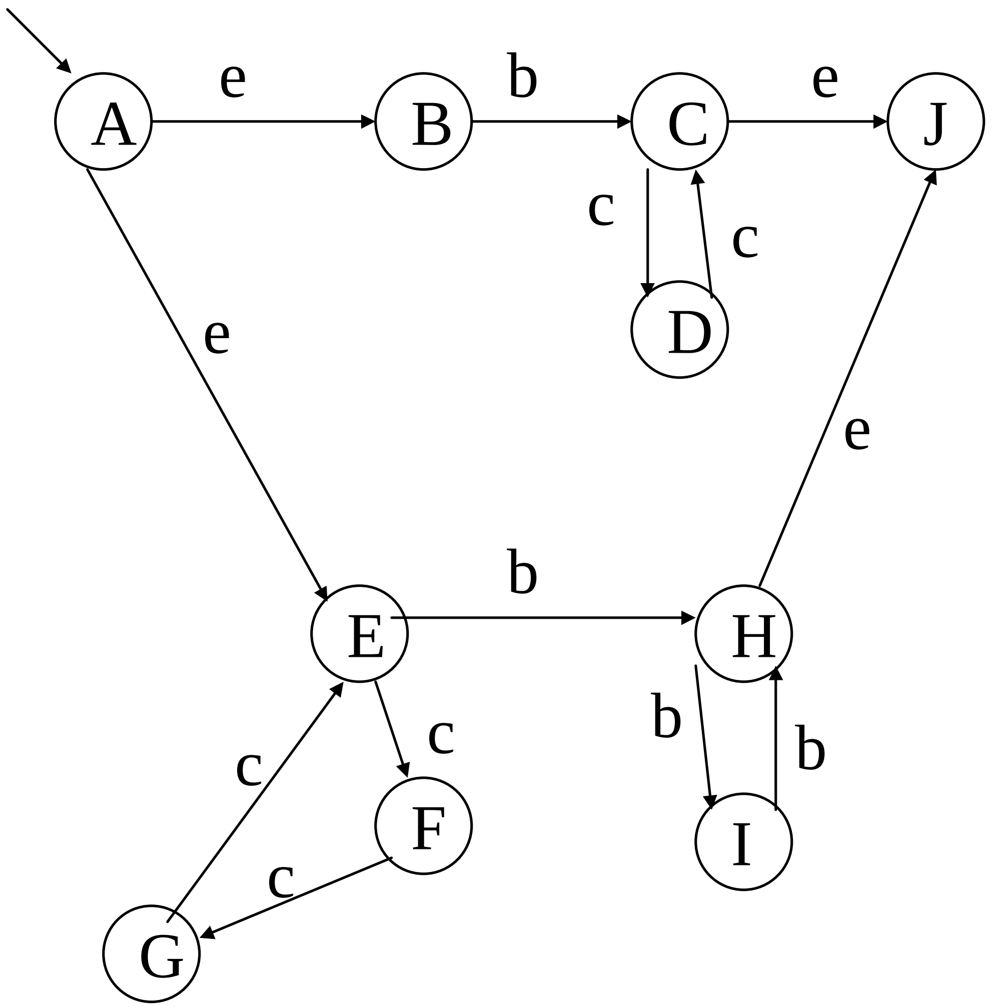
Regular expression for "divisible by three number of symbols c followed by an odd number of symbols b. We assume zero to be a number divisible by 3.

$b(cc)^* \cup (ccc)^*b(bb)^*$

Union of these two regular expressions realizes language L

Solution to Problem 5.

- (b) Draw a graph of this regular expression

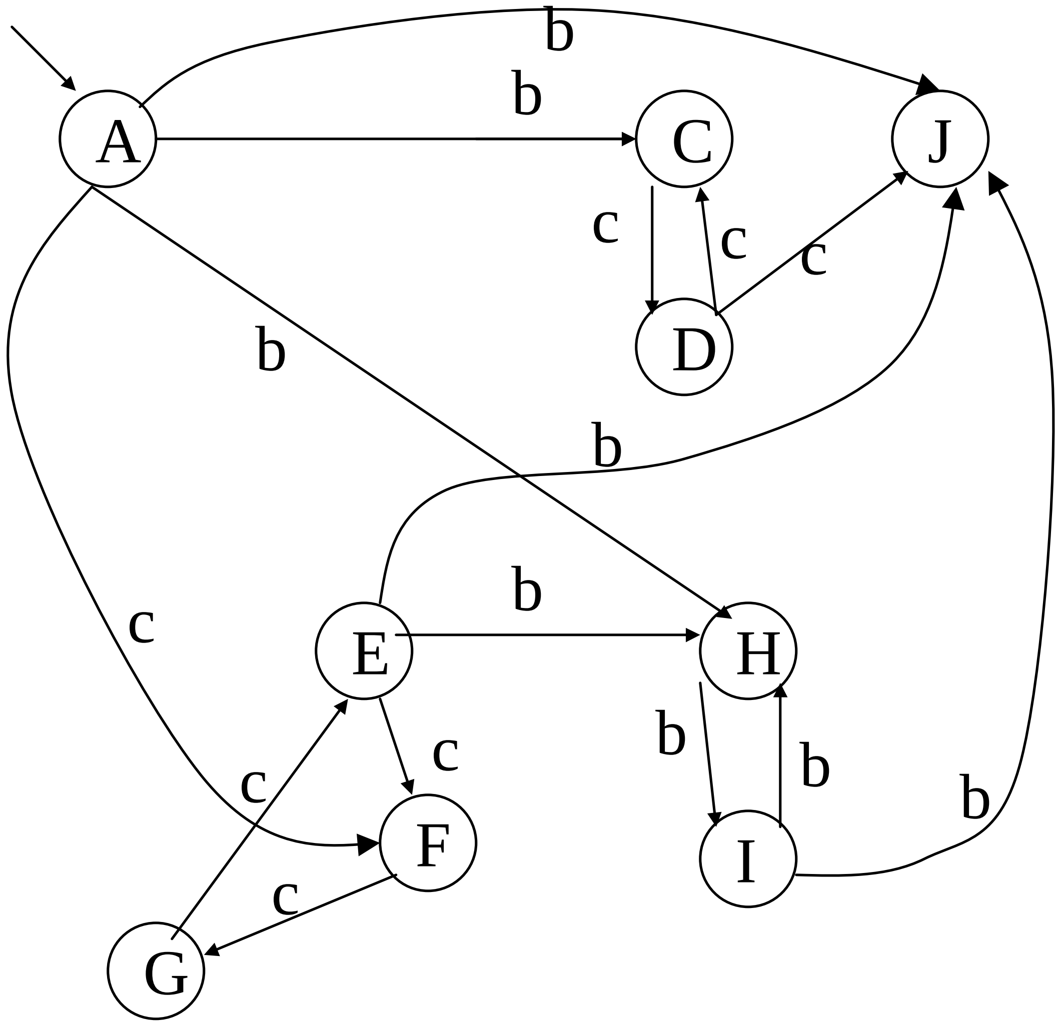


e denotes an empty symbol

I used a “safe” method from the class here. The first e on top left could be avoided.

Solution to Problem 5.

- (c)Convert the graph to a non-deterministic machine



All symbols e are removed and paths are adjusted to represent the same language

Solution to Problem 5.

- (d) Convert the non-deterministic machine to an equivalent deterministic machine.

$$\begin{tikzcd}
 & & \{A\} \arrow[dl, "b"'] \arrow[dr, "c"] & & \\
 & \{C,H,J\} \arrow[dl, "b"'] \arrow[dr, "c"] & & \{F\} \arrow[dr, "c"] \arrow[dddll, "b"] & \\
\{I\} \arrow[d, bend left, "b"] \arrow[ddr, "c"] & & \{D\} \arrow[d, bend left, "c"] \arrow[ddl, "b"'] & & \{G\} \arrow[d, "c"] \arrow[ddlll, "b"] \\
\{H,J\} \arrow[u, bend left, "b"] \arrow[dr, "c"'] & & \{C,J\} \arrow[u, bend left, "c"] \arrow[dl, "b"] & & \{E\} \arrow[uul, bend right, "c"'] \arrow[dlll, bend left, "b"] \\
 & \text{error} & & &
\end{tikzcd}$$

The table from next page verifies all stages. OK. If the columns were different there would be in error in calculations

All accepting states include J

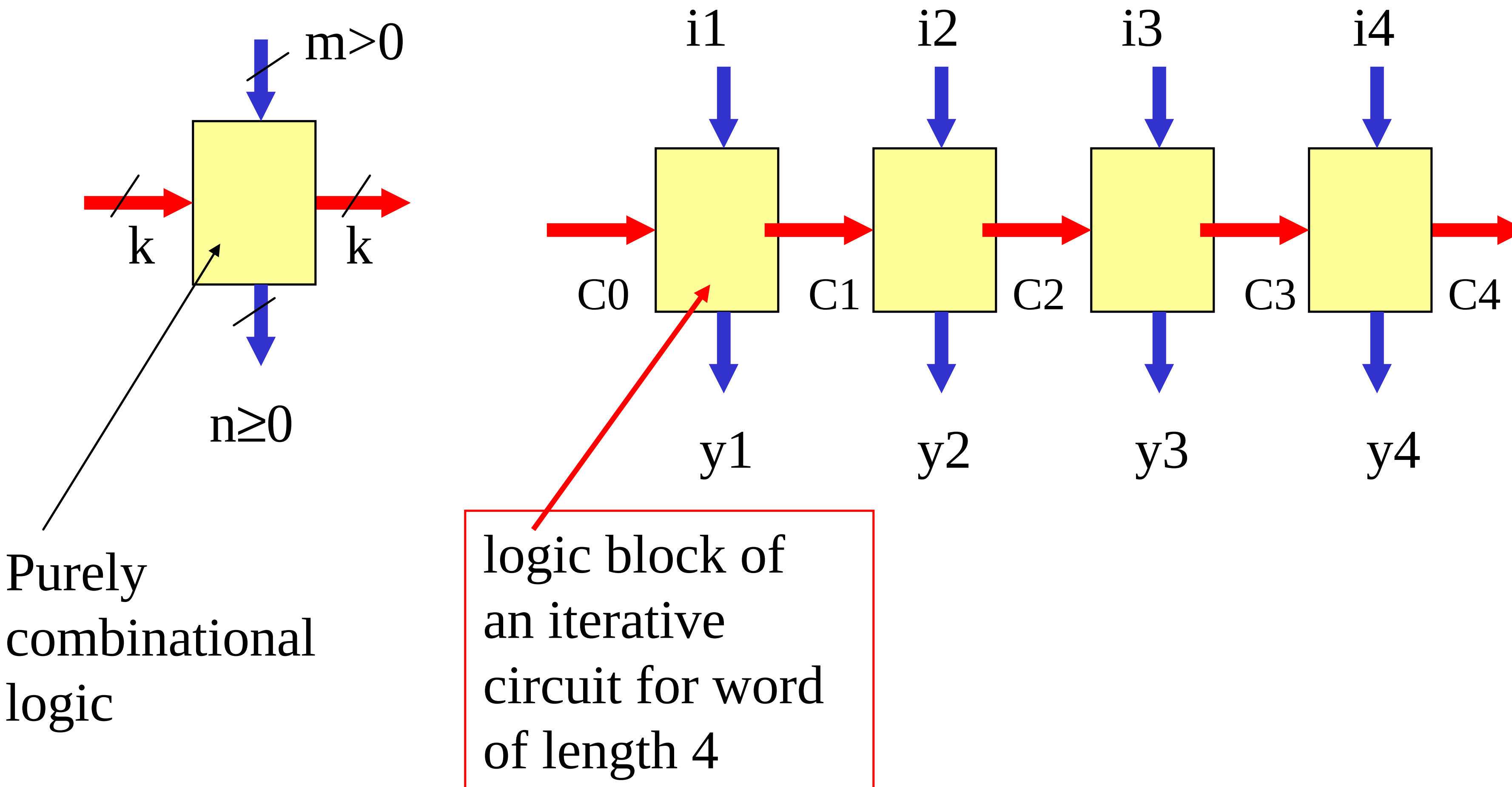
	Expression	Non-deterministic	Deterministic
Empty	no	no	no
b	yes	yes	yes
c	no	no	no
bb	no	no	no
bc	no	no	no
cb	no	no	no
cc	no	no	no
ccc	no	no	no
ccb	no	no	no
cbc	no	no	no
cbb	no	no	no
bcc	yes	yes	yes
bcb	no	no	no
bbc	no	no	no
bbb	yes	yes	yes

Problem 6. Iterative circuits.

- (a) Define what is an iterative circuit.
- (b) Write what is a relation between one-directional, one-dimensional iterative circuit and a Finite State Machine. Explain the Trade-off between speed and area in digital design and illustrate them on two versions of a circuit for comparison of two numbers - one iterative combinational and one a finite state machine.
- (c) Design an iterative combinational circuit with three outputs: $p= (A>B)$, $r=(A=B)$, $s=(A<B)$. Assume delay t1 for every logic gate with 2 inputs. Compare starting from the least significant bit. Draw the transition graph (state machine) for the single combinational block. Calculate the total delay of the circuit. Draw the schematics. Explain your design stages.
- (d) Use the transition graph from point (c) above to draw the sequential FSM realizing serial comparison for the same task.

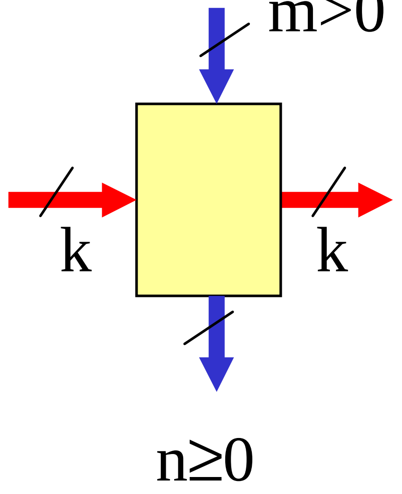
Solution to Problem 6.

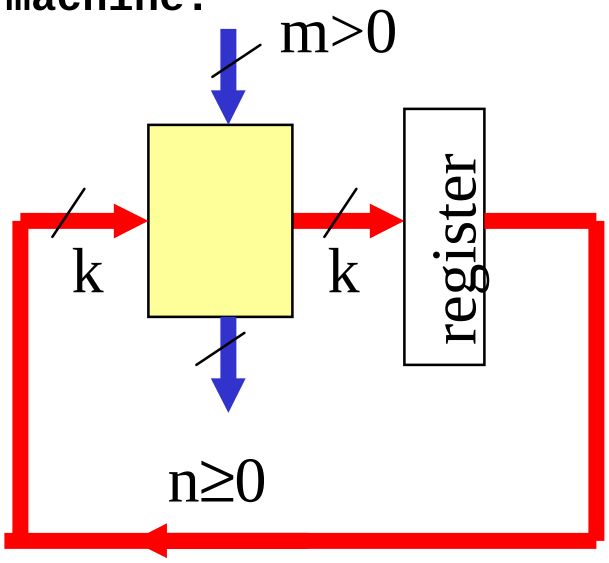
- (a) Define what is an iterative circuit.
  - Iterative circuit is a combinational circuit with a sequence of blocks. Each block has iterative (carry) inputs and iterative outputs. It has also direct inputs and may have direct outputs. All blocks (except of possibly the first and the last) as the same.


m>0
k
k
n≥0
Purely combinational logic
i1
i2
i3
i4
C0
C1
C2
C3
C4
y1
y2
y3
y4
logic block of an iterative circuit for word of length 4

Problem 6. Iterative circuits.

- (b) Write what is a relation between one-directional, one-dimensional iterative circuit and a Finite State Machine. Explain the Trade-off between speed and area in digital design and illustrate them on two versions of a circuit for comparison of two numbers - one iterative combinational and one a finite state machine.





In FSM the iteration is done in time, by storing intermediate signals in a register.

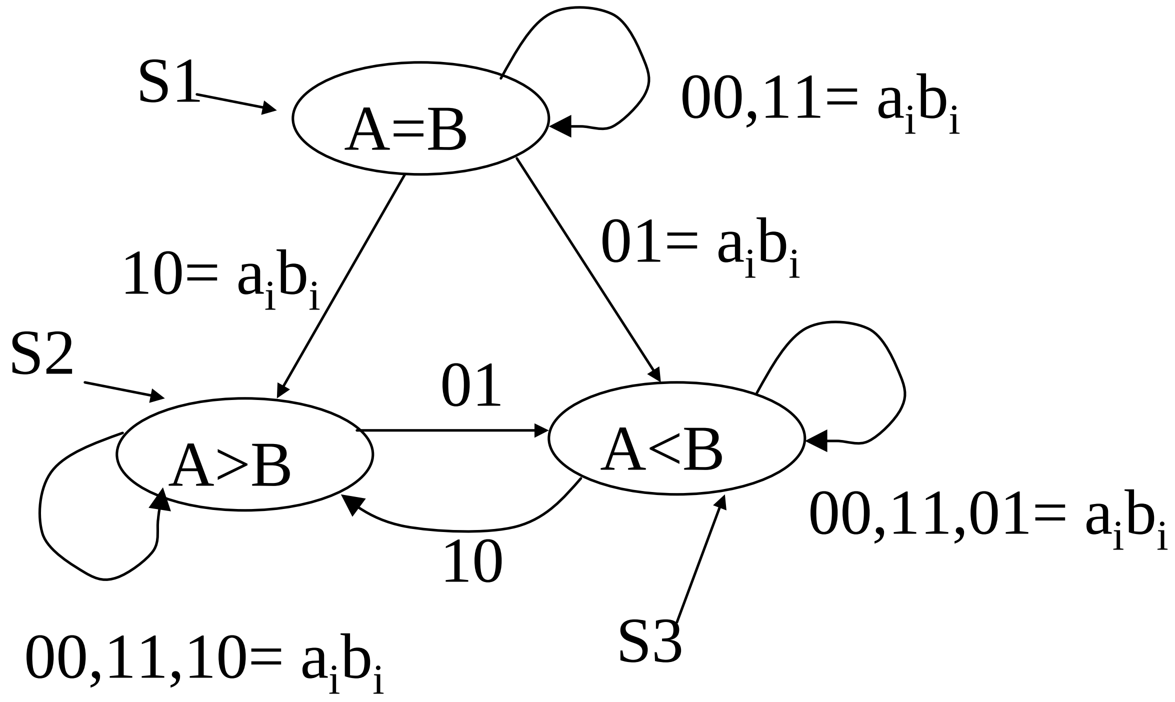
As shown in previous slide, in iterative circuit you iterate this combinational block in space, in one dimension.

For the word of length M, the delay is M*DB where DB is the delay of the block. The cost is M*DB

For the word of length M, the delay is M*(DB + reg-delay) where reg-delay is a total delay related to setting and reading the register. The cost is M + register-Cost. Thus FSM for the same task is cheaper but slower.

Problem 6. Iterative circuits.

- (c) Design an iterative combinational circuit with three outputs: p=(A>B), r=(A=B), s=(A<B). Assume delay t1 for every logic gate with 2 inputs. Compare starting from the least significant bit. Draw the transition graph (state machine) for the single combinational block. Calculate the total delay of the circuit. Draw the schematics. Explain your design stages.
- (d) Use the transition graph from point (c) above to draw the sequential FSM realizing serial comparison for the same task.



PS \ a_ib_i	00	01	11	10
S1=00	00	01	00	10
S3=01	01	01	01	10
-- =11	--	--	--	--
S2=10	10	01	10	10

Problem 6. Iterative circuits.

PS \ $a_i b_i$	00	01	11	10
S1=00	00	01	00	10
S3=01	01	01	01	10
-- =11	--	--	--	--
S2=10	10	01	10	10

$$Q1^+ = Q1\,(a'b)' + (ab')$$

$$Q2^+ = Q2\,(ab')' + (a'b)$$

Observe smart design based on reuse of groups and inhibition

PS \ $a_i b_i$ (Q1 Q2)	00	01	11	10
S1=00	0	0	0	1
S3=01	0	0	0	1
-- =11	-	-	-	-
S2=10	1	0	1	1

$Q1^+$

PS \ $a_i b_i$ (Q1 Q2)	00	01	11	10
S1=00	0	1	0	0
S3=01	1	1	1	0
-- =11	-	-	-	-
S2=10	0	1	0	0

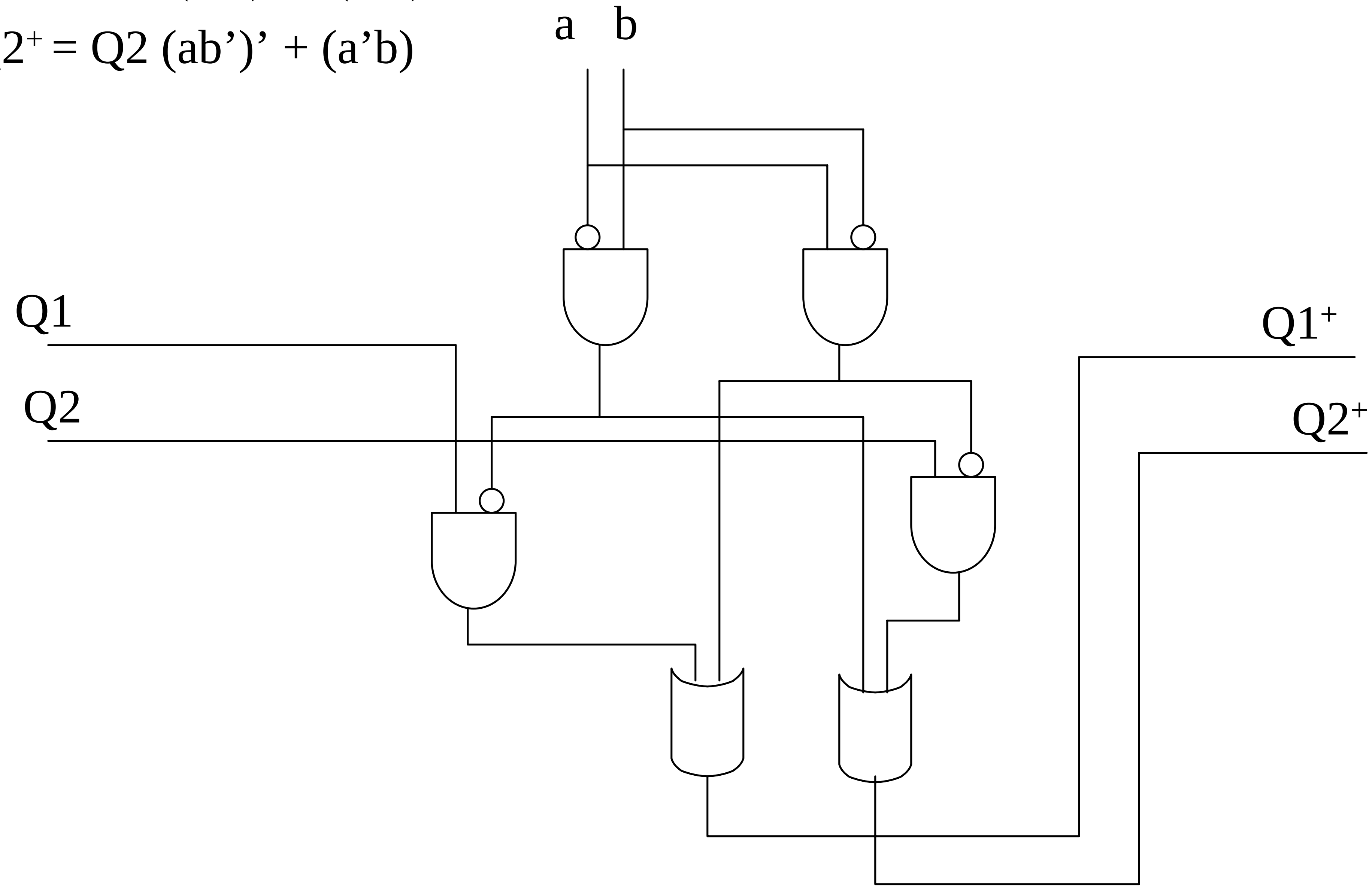
Group ab' used for inhibition

$Q2^+$

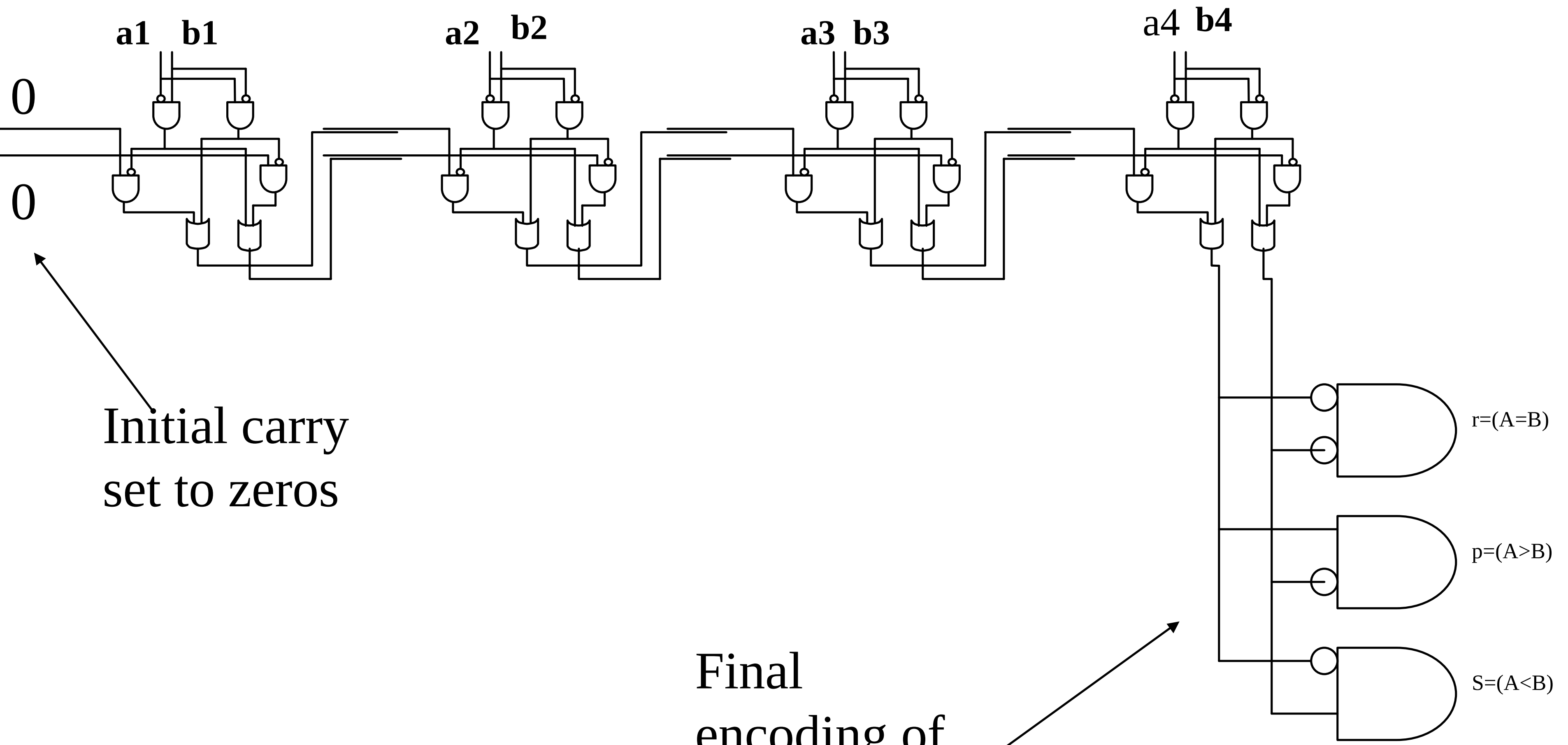
Problem 6. Iterative circuits.

$Q1^+ = Q1\,(a'b)' + (ab')$

$Q2^+ = Q2\,(ab')' + (a'b)$



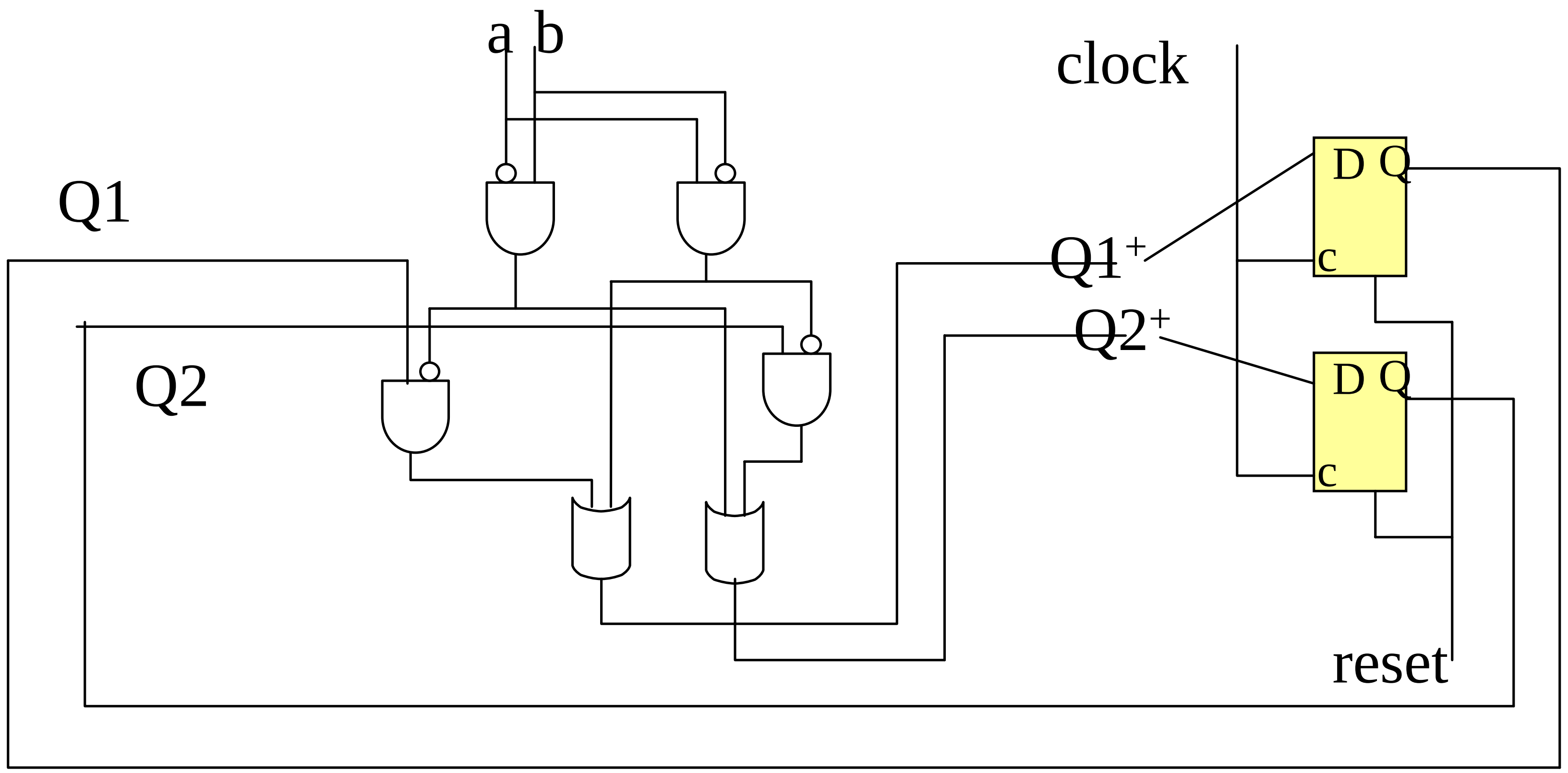
Problem 6. Iterative circuits.



Initial carry set to zeros

Final encoding of decision signals

Problem 6. Iterative circuits.



Realization of the Finite State Machine for the comparator

The output decoder with outputs p, r and s as in previous slide can be also added at the output of flip-flops.

Problem 7. Turing Machine.

- Design of a Turing machine to calculate number **2n** given number **n** on a tape. Both numbers are represented by subsequent ones.
- Example for n=3:
- 011100 --> 0111011111100

Initial head position   Final head position

<u>Perform the following:</u>

(a) draw the data path from functional blocks

(b) draw the control unit and how the data path and the control unit are connected.

(c) realize the control unit as any machine of your choice - Mealy, Moore, netlist of flip-flops, OR and branching gates, or a microprogrammed unit.

(d) verify using your schematics and the example above (n --> 2n) that your machine works correctly.

Solution to Problem 7.

- **Design of a Turing machine to calculate number 2n given number n on a tape. Both numbers are represented by subsequent ones.**
- **Example for n=3:**
- **011100 --> 0111011111100**

Instructions for Post Machine

S=stop

R=move head right

L=move head left

J(0)=Jump if zero to instruction shown by an arrow

J(1)=Jump if one to instruction shown by an arrow

(1) = write 1 to the tape

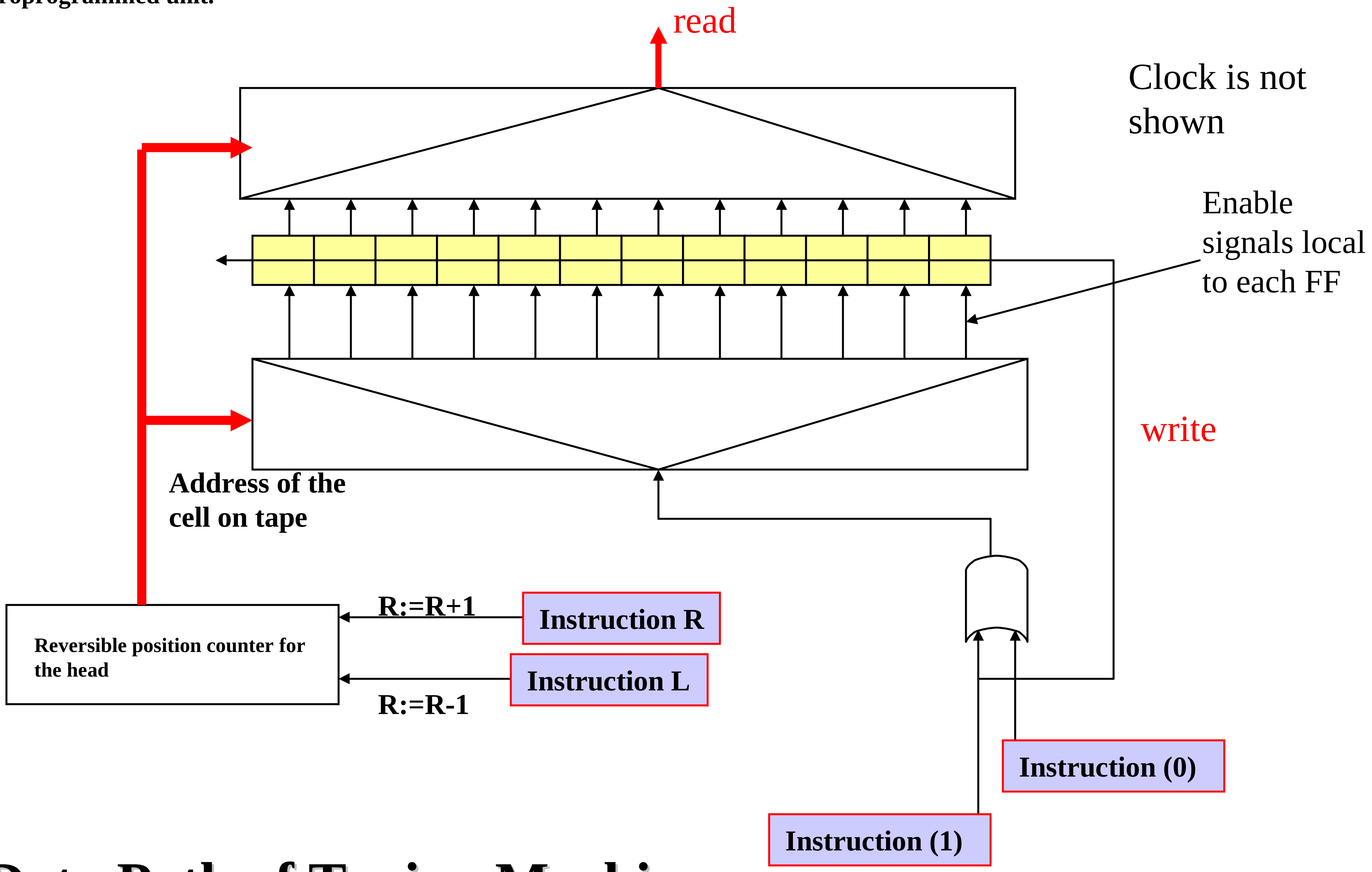
(0) = write 0 to the tape

Turing Machine and Post Machine are equivalent. Turing Machine has Finite State Machine Control and Post has a program with instructions and jumps. So it is easier to design Post Machine and next convert it to equivalent Turing Machine. Or we can design a microcontroller.

(a) draw the data path from functional blocks

(b) draw the control unit and how the data path and the control unit are connected.

(c) realize the control unit as any machine of your choice - Mealy, Moore, netlist of flip-flops, OR and branching gates, or a microprogrammed unit.

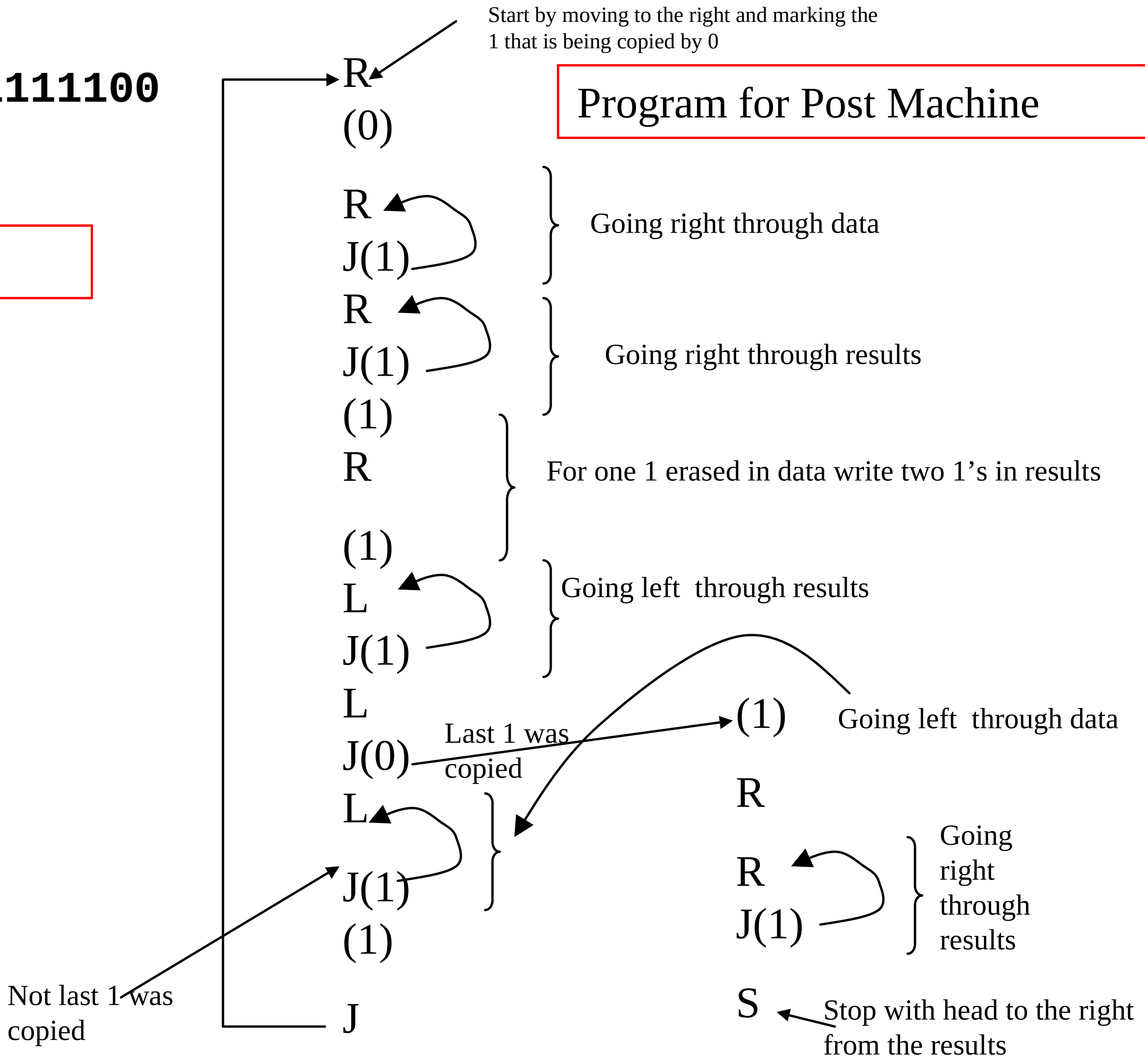

read
Clock is not shown
Enable signals local to each FF
write
Address of the cell on tape
R:=R+1
Instruction R
Reversible position counter for the head
Instruction L
R:=R-1
Instruction (0)
Instruction (1)

Data Path of Turing Machine

Solution to Problem 7.

- **Design of a Turing machine to calculate number 2n given number n on a tape. Both numbers are represented by subsequent ones.**
- **Example for n=3:**
- **011100 --> 0111011111100**

Program for Post Machine



Start by moving to the right and marking the 1 that is being copied by 0
R
(0)
R
J(1)
Going right through data
R
J(1)
Going right through results
(1)
R
(1)
For one 1 erased in data write two 1's in results
L
J(1)
Going left through results
L
J(0)
Last 1 was copied
L
J(1)
(1)
J
Not last 1 was copied
Going left through data
(1)
R
R
J(1)
Going right through results
S
Stop with head to the right from the results

Instructions for Post Machine

S=stop

R=move head right

L=move head left

J(0)=Jump if zero to instruction shown by an arrow

J(1)=Jump if one to instruction shown by an arrow

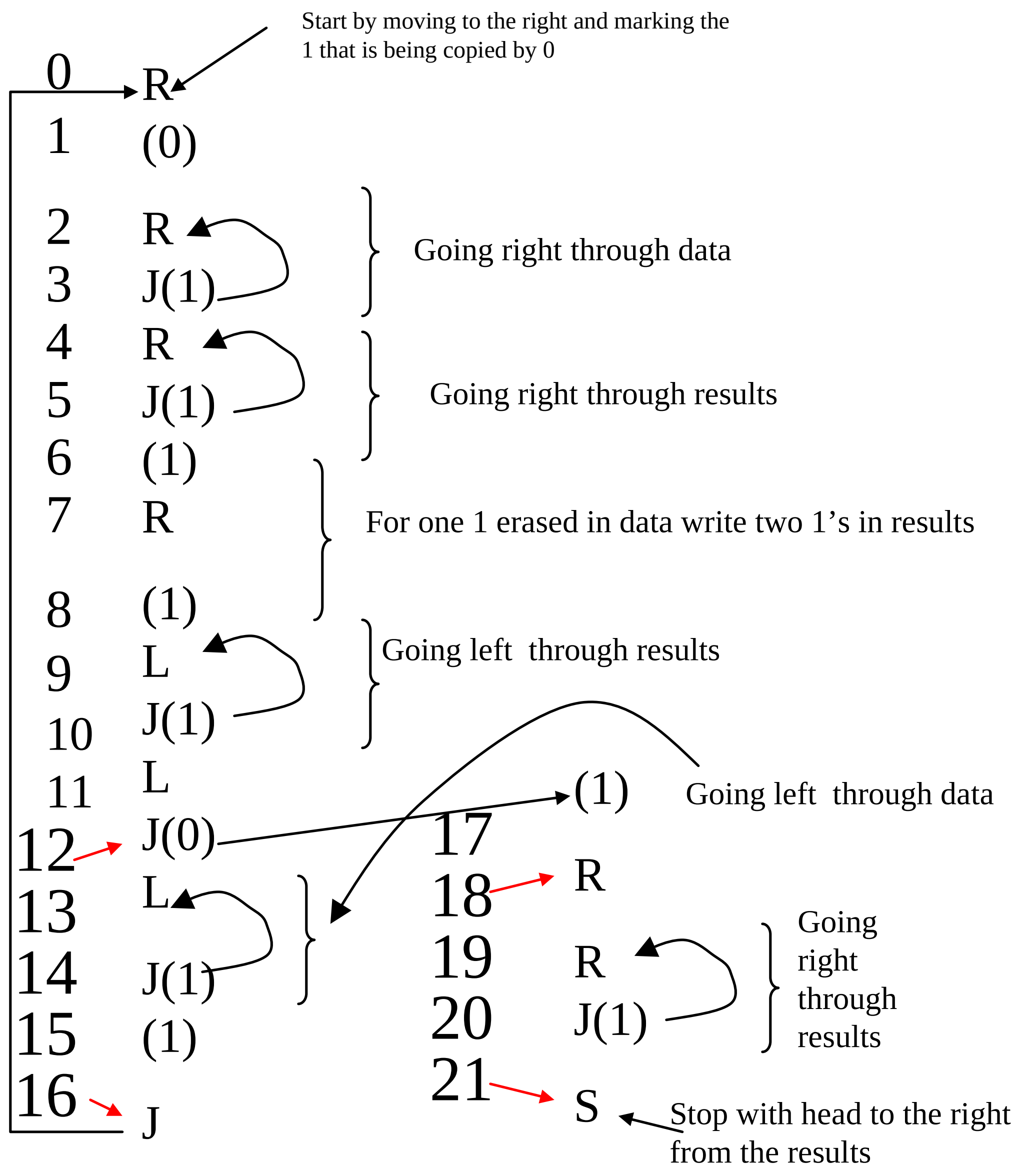
(1) = write 1 to the tape

(0) = write 0 to the tape

Enumeration for instructions is shown

Present state	Instruction	Condition to check	Jump address
0	R	0	-
1	0	0	-
2	R	0	-
3	J	t1	2
4	R	0	-
5	J	t1	4
6	1	0	-
7	R	0	-
8	1	0	-
9	L	0	-
10	J	t1	9
11	L	0	-
12	J	t0	17
13	L	0	-
14	J	t1	13
15	1	0	-
16	J	1	0
17	1	0	-
18	R	0	-
19	R	0	-
20	J	t1	19
21	S	0	-

t0=tape 0, t1 = tape 1,0=constant 0=counter increase, 1 = constant 1 = unconditional jump


Start by moving to the right and marking the 1 that is being copied by 0
0 R
1 (0)
2 R
3 J(1)
4 R
5 J(1)
6 (1)
7 R
8 (1)
9 L
10 J(1)
11 L
12 J(0)
13 L
14 J(1)
15 (1)
16 J
17 (1)
18 R
19 R
20 J(1)
21 S
Going right through data
Going right through results
For one 1 erased in data write two 1's in results
Going left through results
Going left through data
Going right through results
Stop with head to the right from the results

Present state	Instruction	Condition to check	Jump address
0	R	0	-
1	0	0	-
2	R	0	-
3	J	t1	2
4	R	0	-
5	J	t1	4
6	1	0	-
7	R	0	-
8	1	0	-
9	L	0	-
10	J	t1	9
11	L	0	-
12	J	t0	17
13	L	0	-
14	J	t1	13
15	1	0	-
16	J	1	0
17	1	0	-
18	R	0	-
19	R	0	-
20	J	t1	19
21	S	1	21

$\longrightarrow$

encoding

Present state	Instruction	Condition to check	Jump address

00000

binary encoding

…..

.

..

10101

1=000
0=010
L=100
R=110
S=001
J= --1

Encoding of instruction field

t1 = read = 00 address for mux

t0 = NOT(read)=01address for mux

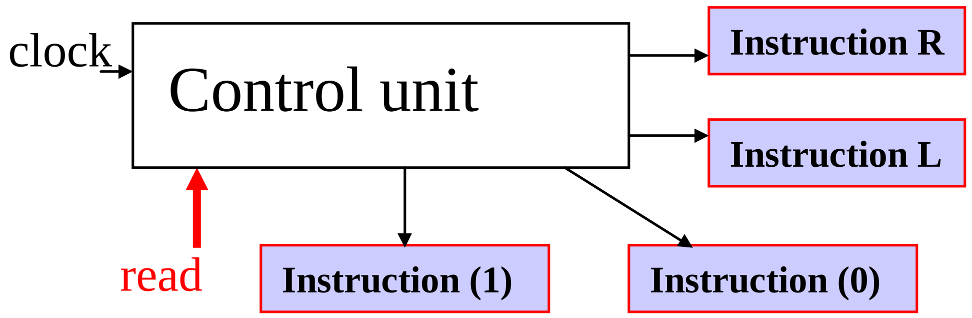
0 = constant one = 10 address for mux

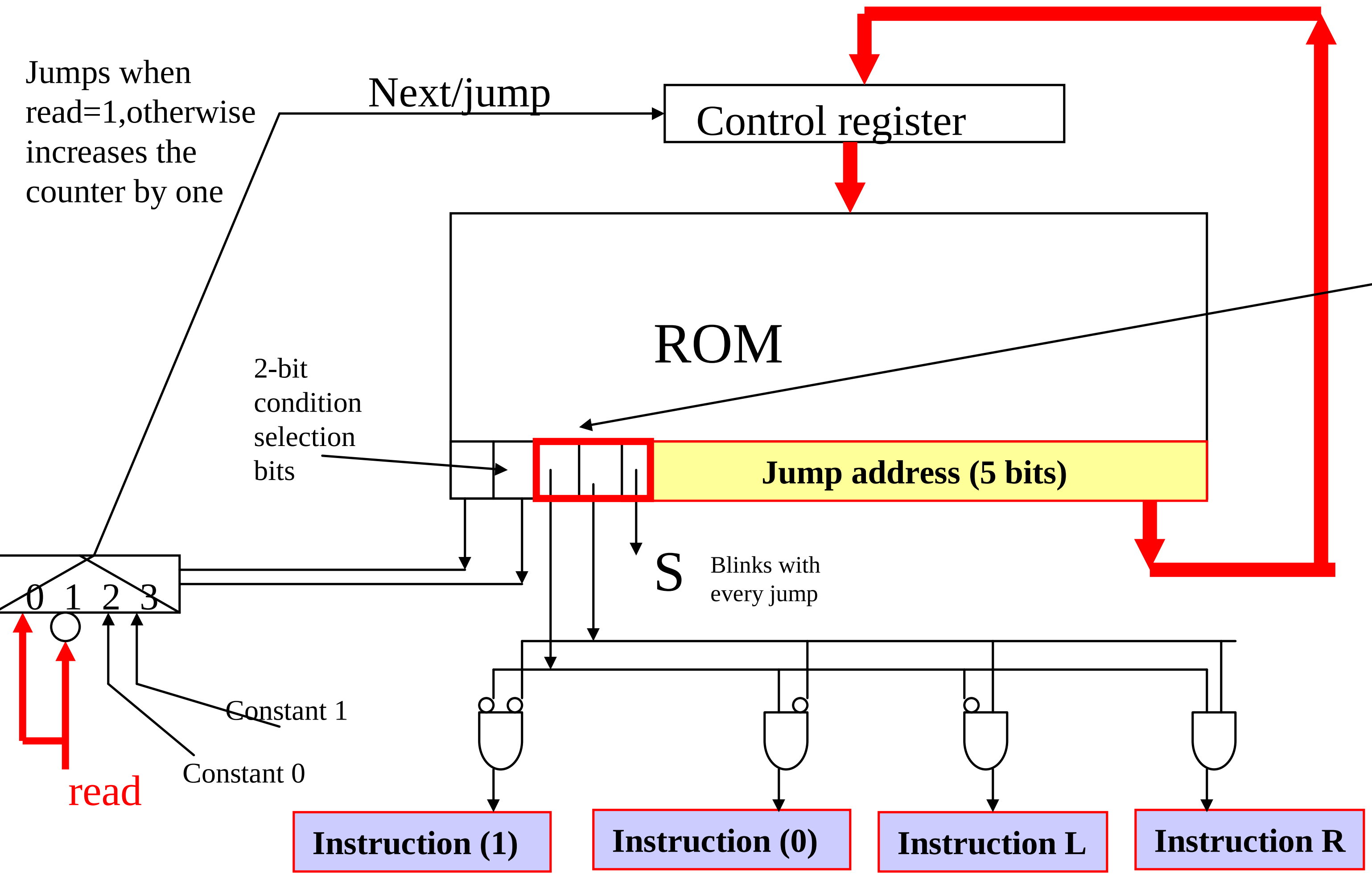
1= constant one selected=11 address for mux

Encoding of condition to check in mux

To complete the table just replace symbols with their binary codes

Microprogrammed Control Unit of Turing Machine





Problem 8. Machines with stack.

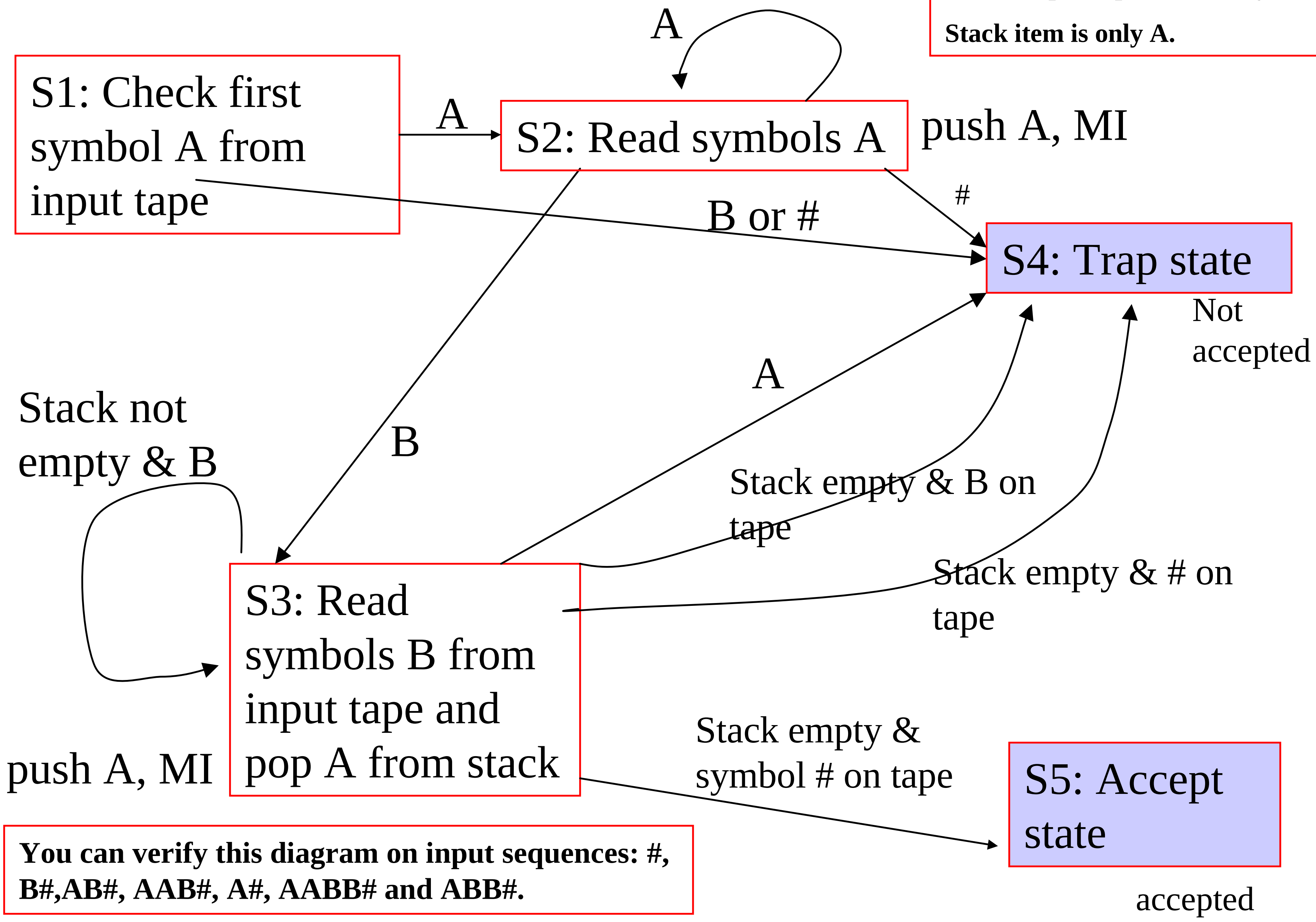
- (a)Design a machine with a control unit, an input shift register and a stack that accepts language $\cup A^n B^n = AB \cup AABB \cup AAABBB \ldots$
- (b) Show details of data path design.
- (c)Discuss the role of all signals.
- (d) Design a microprogrammed control unit for the stack and input register control.

Solution to Problem 8.First stage:draw diagram of controller

Input alphabet={A,B}

End of input tape marked by #.

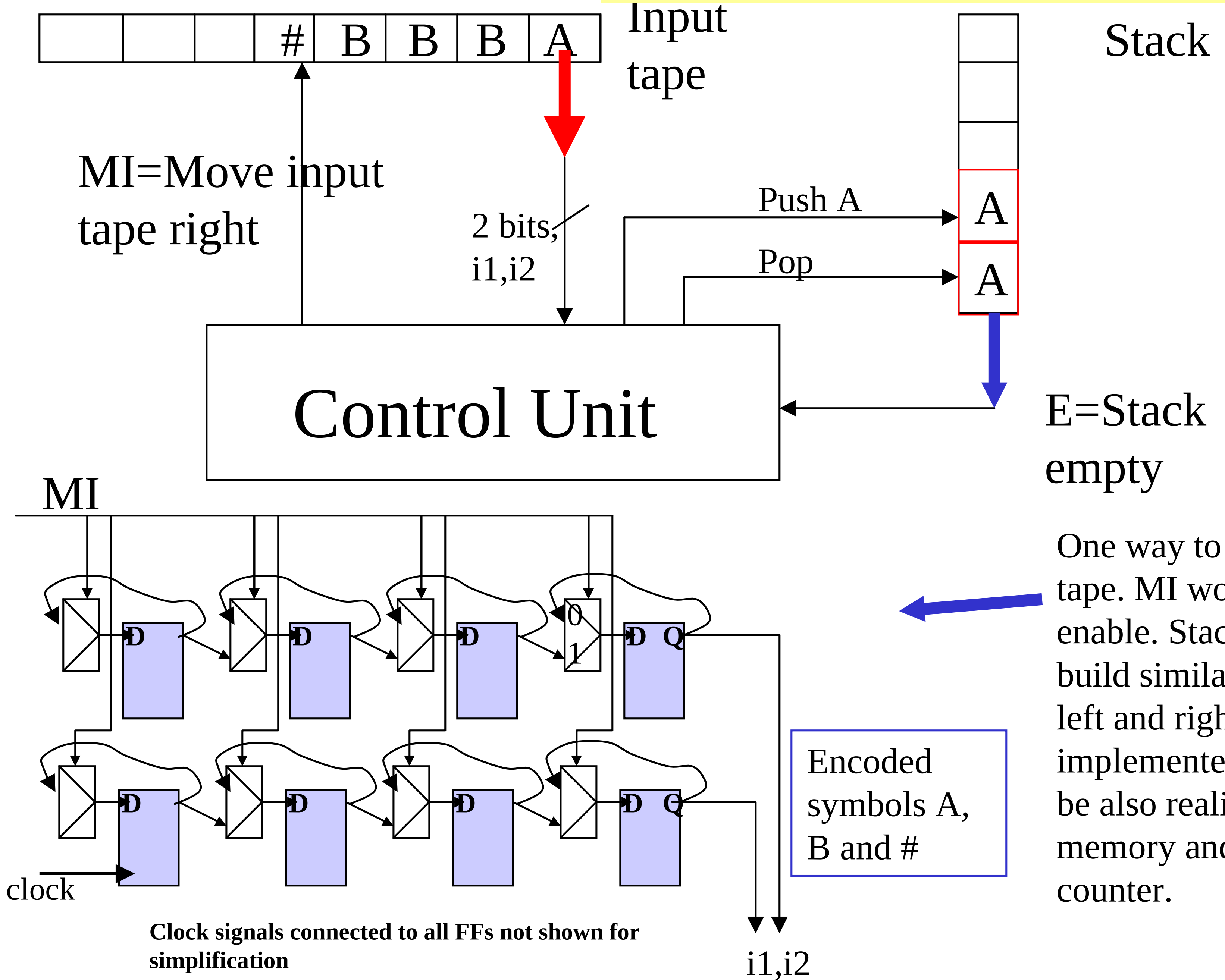
Stack item is only A.



You can verify this diagram on input sequences: #, B#,AB#, AAB#, A#, AABB# and ABB#.

Second stage:Encode symbols and draw diagram of data path and blocks

A encoded as 10, B as 01, # as 00



Clock signals connected to all FFs not shown for simplification

One way to realize input tape. MI works as enable. Stack can be build similarly. But shift left and right should be implemented. Stack can be also realized with a memory and reversible counter.

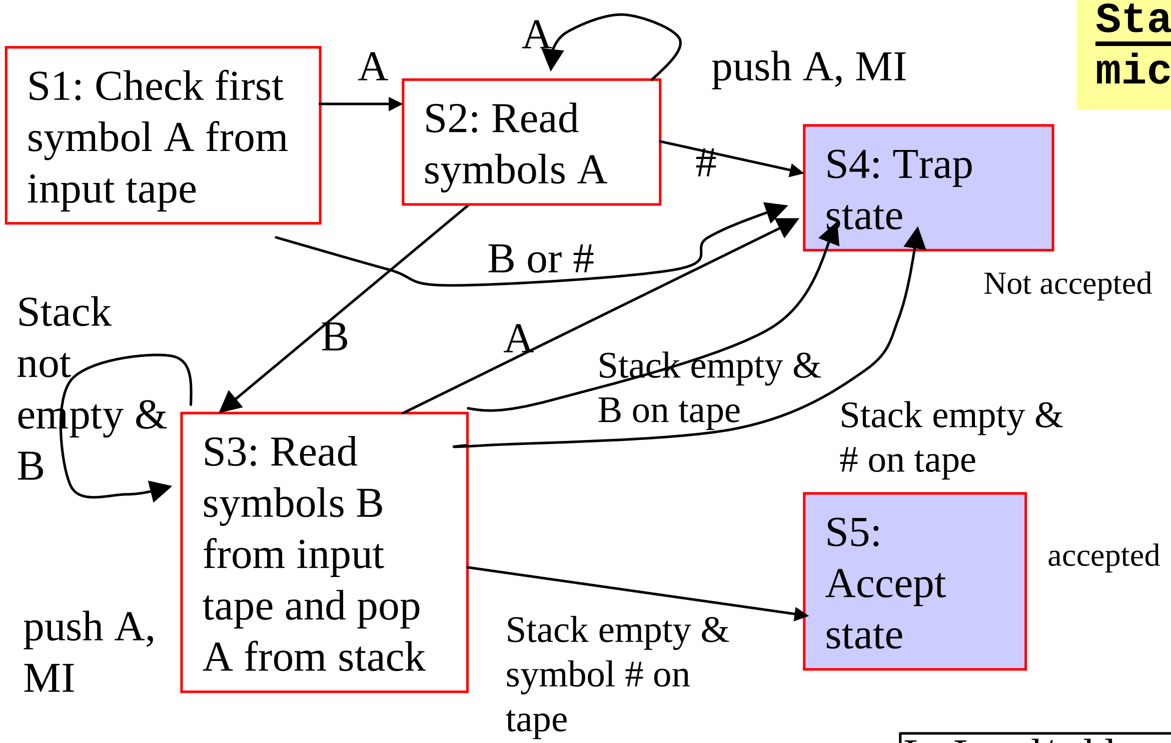
Stage 3:Create the table to program the ROM. It includes first only symbols and next they are encoded.Binary (encoded) data are used to program the ROM.

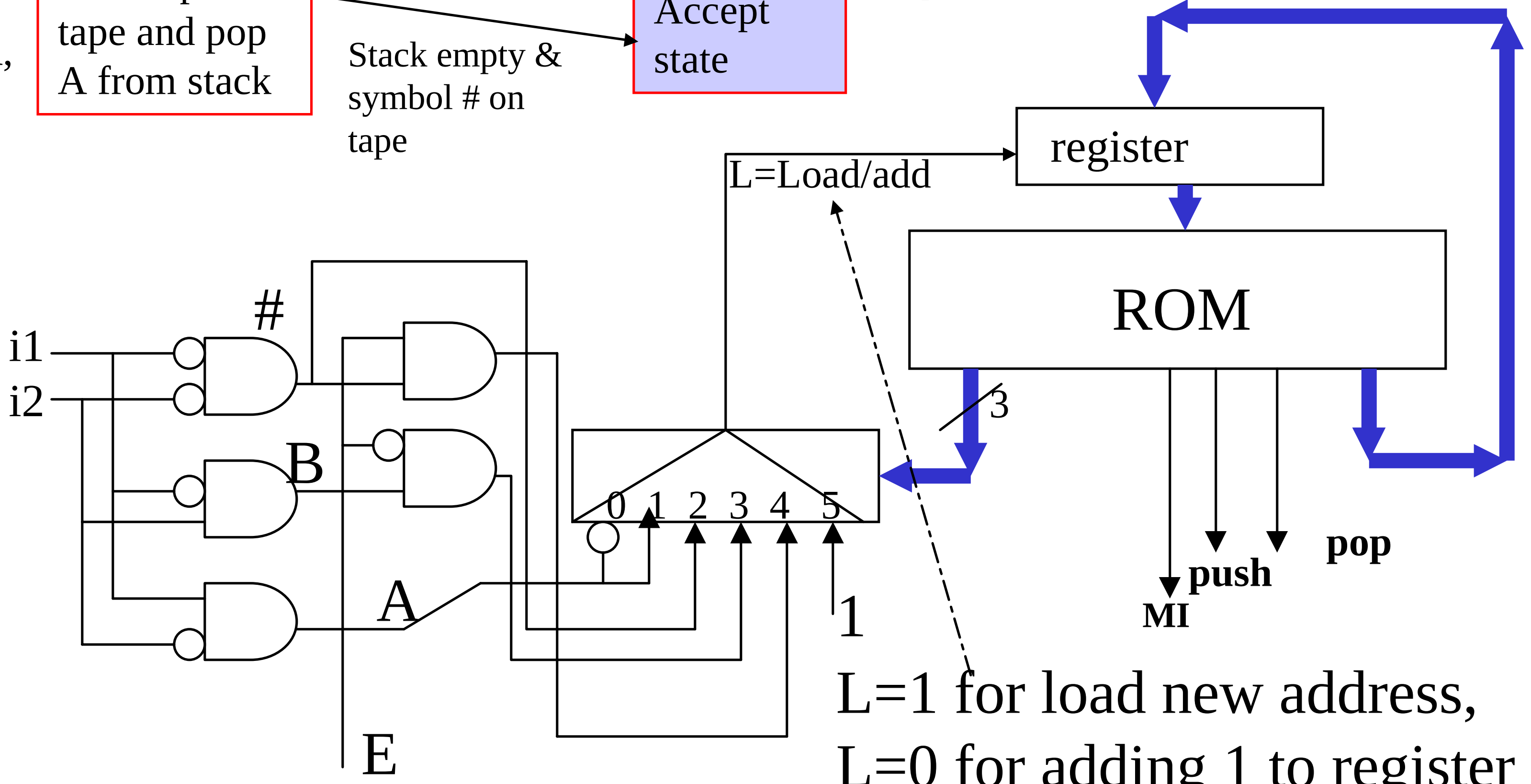
Present state	Encoded Present state	Condition Checked	Encoded Condition Checked	MI	push	pop	accept	Not accept	Jump state	Encoded jump state
S1	000	A’	0=000	0	0	0	0	0	S4	101
S2A	001	A	1=001	1	1	0	0	0	S2A	001
S2B	010	#	2=010	0	0	0	0	0	S4	101
S3A	011	B * E’	3=011	1	0	1	0	0	S3A	011
S3B	100	E * #	4=100	0	0	0	0	0	S5	110
S4	101	1	5=101	0	0	0	0	1	S4	101
S5	110	1	5=101	0	0	0	1	0	S5	110

Stage 4:Draw diagram of the microprogrammed controller

A encoded as i1 i2=10, B as 01, # as 00

This is not an optimized circuit



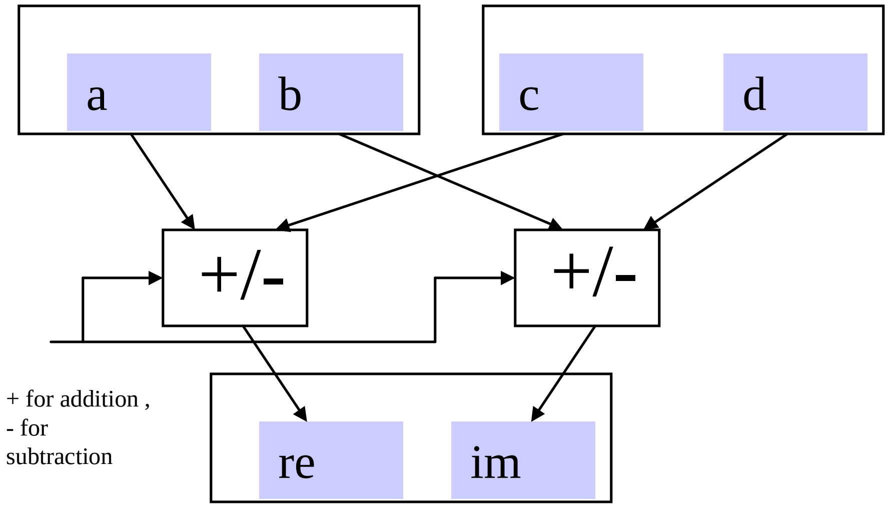


L=1 for load new address,
L=0 for adding 1 to register

Problem 9. Controller Design.

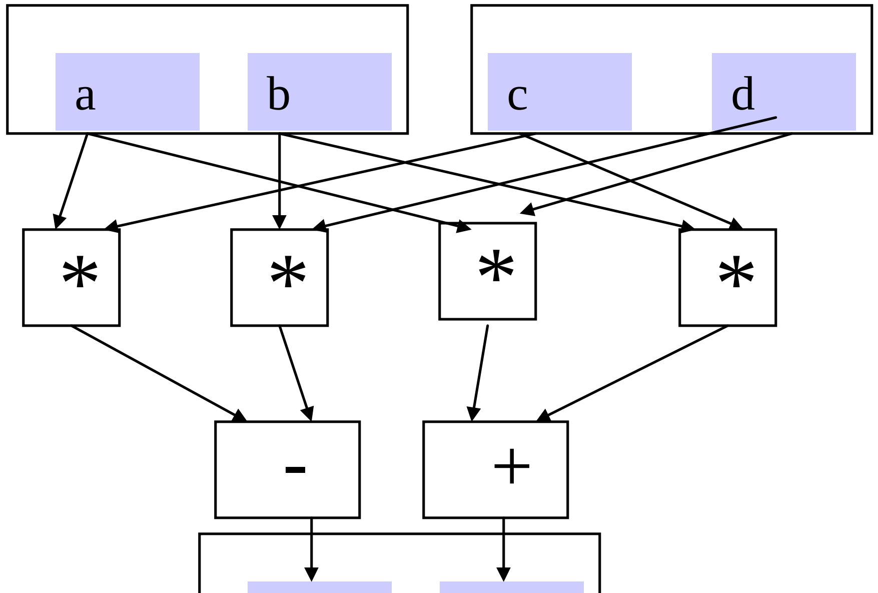
- (a) Design a sequential circuit with arbitrary blocks that executes operations of addition, multiplication, division and subtraction on ***complex numbers***.
- (b) Assume that you have available blocks that realize combinationally operations of addition, subtraction, multiplication and division of 8-bit registers.
- (c) Realize and draw the state graph of the controlling state machine and realize it using arbitrary method.
- (d) Draw the data path circuit. Show details of controlling registers. If necessary, optimize the controlling signals.
- (e) Verify your solution.

Realization of addition/subtraction



(a+jb) +/- (c+jd) = (a +/- c) + j(b +/- d)

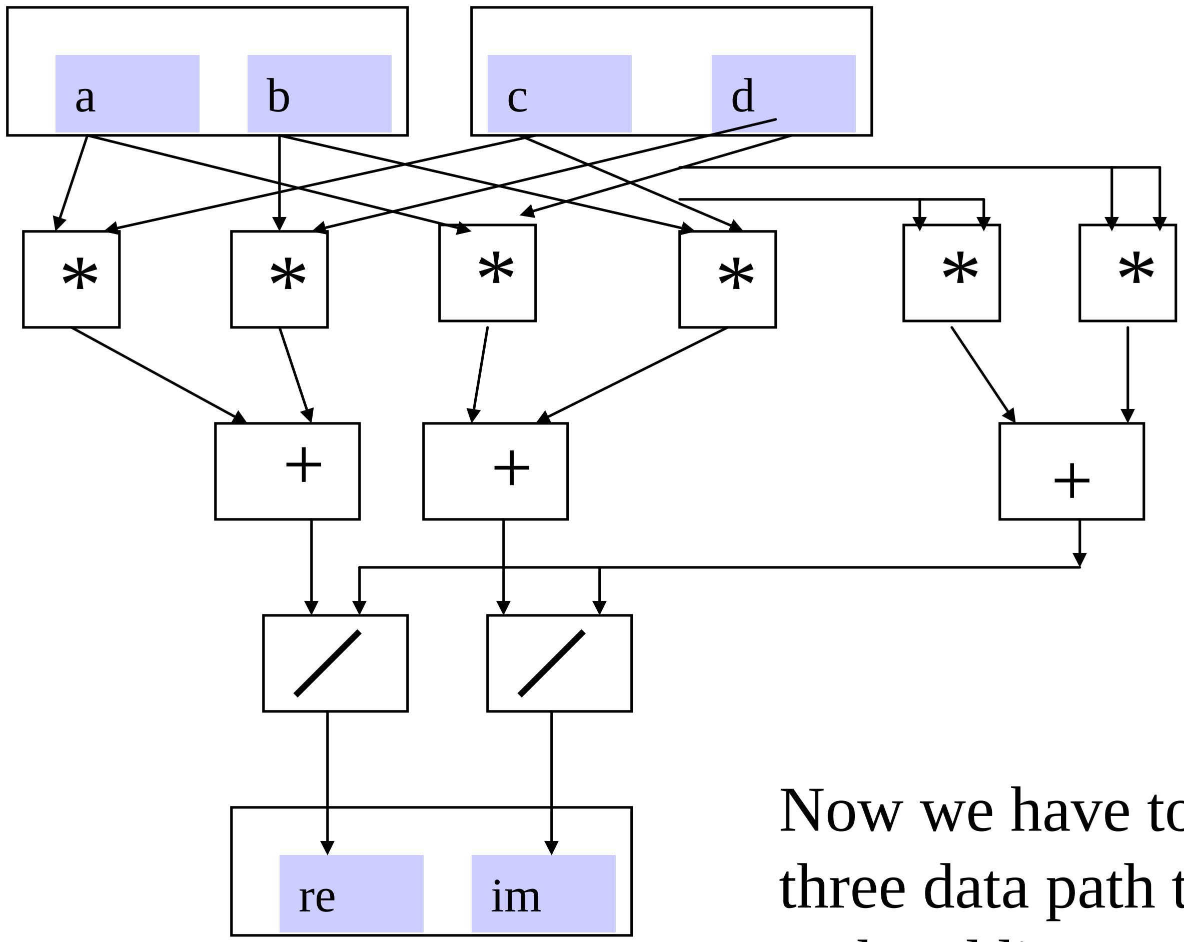
Realization of multiplication



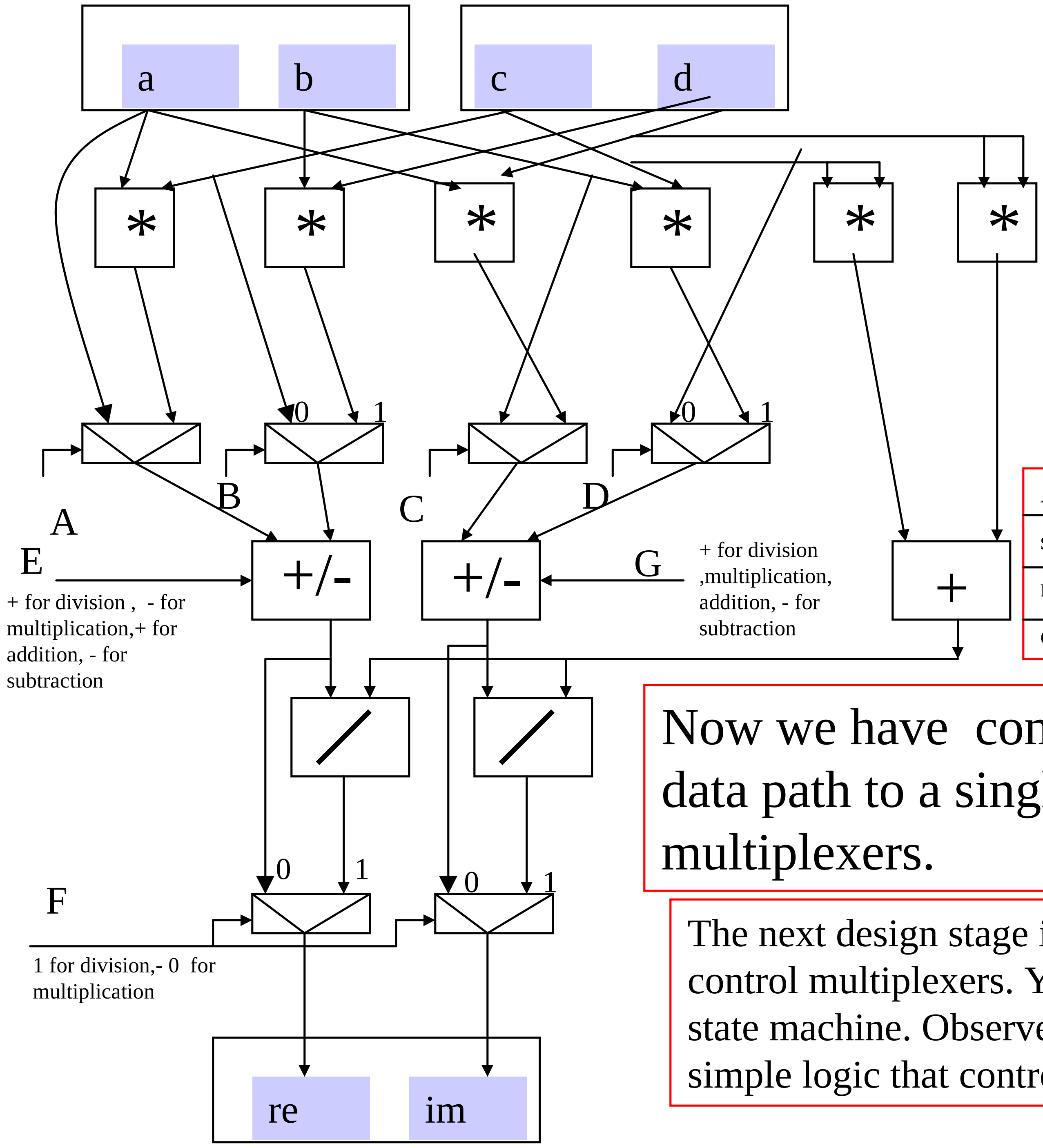
(a+jb) * (c+jd) = ac+ajd + jbc+ j^2 bd = (ac-bd)+j(ad+bc)

Realization of division

$$(a+jb) / (c+jd) = (a+jb)(c-jd)/(c+jd)(c-jd) = (ac+bd)+j(ad+bc)/(c^2+d^2)$$



Now we have to combine these three data path to a single data path, adding multiplexers and control.


a
b
c
d
*
*
*
*
*
*
0
1
0
1
A
B
C
D
E
+/-
+/-
G
+
+ for division , - for multiplication,+ for addition, - for subtraction
+ for division ,multiplication, addition, - for subtraction
/
/
0
1
0
1
F
1 for division,- 0 for multiplication
re
im

operation	controls						
	A	B	C	D	E	F	G
Addition	0	0	0	0	0	0	0
subtraction	0	0	0	0	0	0	0
multiplication	0	1	1	1	1	0	0
division	0	0	0	0	1	1	1

Now we have combined these three data path to a single data path, adding multiplexers.

The next design stage is to write tables of signals that control multiplexers. You do not need controlling state machine. Observe that the controller reduces to simple logic that controls muxes.