Problem 10. Minimization of Incompletely Specified Finite State Machines.

a	d/-	f/0
b	c/1	h/0
c	d/0	h/0
d	c/0	a/0
e	-/-	f/0
f	e/0	a/0
g	c/1	-/-
h	b/1	a/1

Machine M

- Given is Machine M
- (a) Find the minimal machine (in the number of states) that is equivalent to machine M
- (b) Draw the triangular table of machine M
- (c) Solve the triangular table
- (d) Find the maximal compatible groups of states
- (e) Solve graphically the covering/closure problem.
- (f) Formulate algebraically the binate covering problem.
- (g) Realize the machine using **JK flip-flops** and combinational gates.

Solution to Problem 10.

a	d/-	f/0
b	c/1	h/0
c	d/0	h/0
d	c/0	a/0
e	-/-	f/0
f	e/0	a/0
g	c/1	-/-
h	b/1	a/1

b	✗ cd, fh						
c	✗ fh	✗					
d	✗ cd, af	✗	✗ ah				
e	v	✗ fh	✗ fh	af			
f	de	✗	✗ de, ah	✗ ce	af		
g	✗ cd	v	✗	✗	v	✗	
h	✗	✗	✗	✗	✗	✗	✗ bc
	a	b	c	d	e	f	g

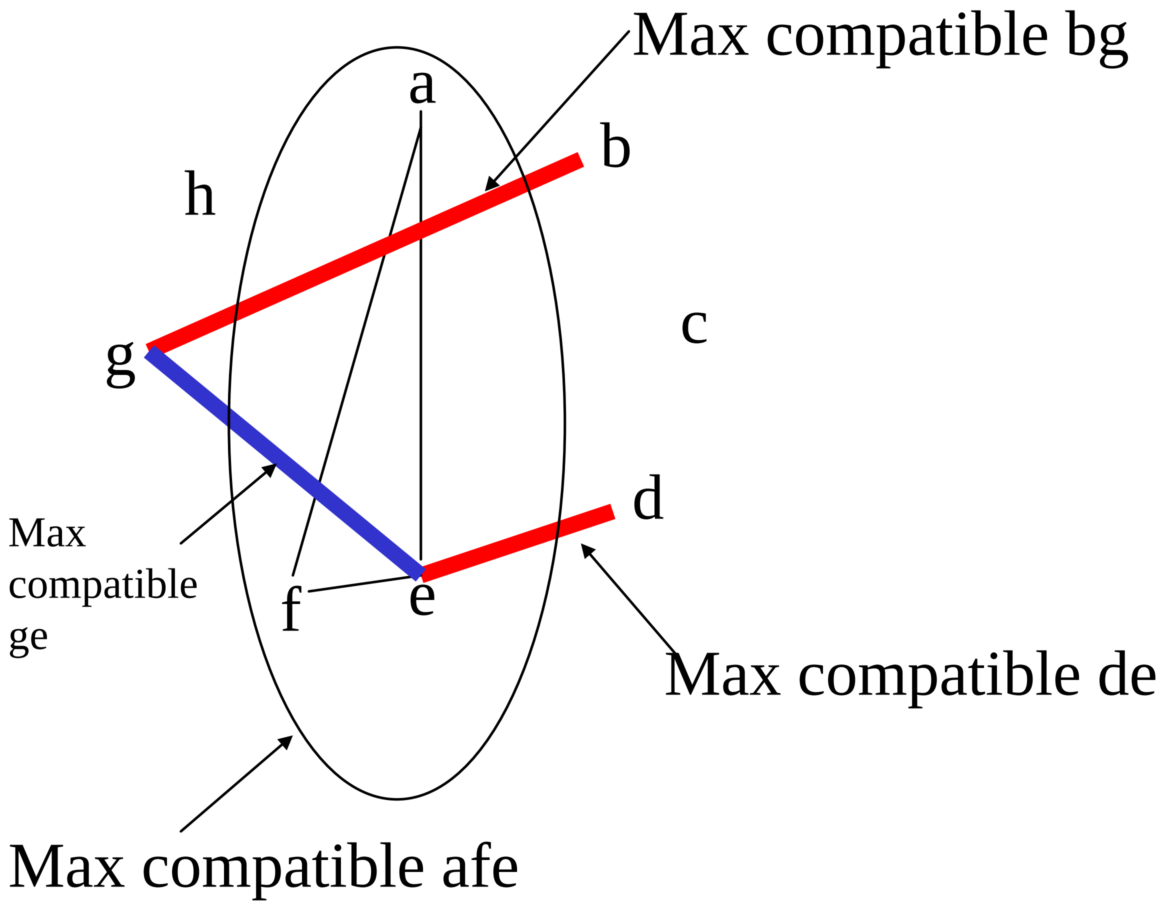
Compatibles = {ae, af, bg, de, ef, eg}

af compatible under de, de compatible under af, ef compatible under af

Solution to Problem 10.

Compatibles = {ae, af, bg, de, ef, eg}

a	d/-	f/0
b	c/1	h/0
c	d/0	h/0
d	c/0	a/0
e	-/-	f/0
f	e/0	a/0
g	c/1	-/-
h	b/1	a/1



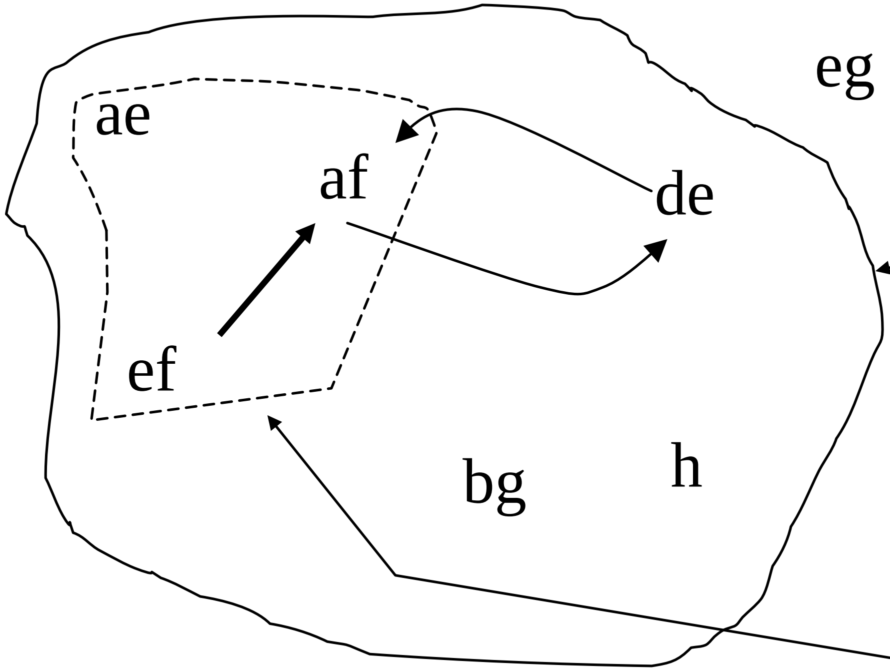
Solution to Problem 10.

a	d/-	f/0
b	c/1	h/0
c	d/0	h/0
d	c/0	a/0
e	-/-	f/0
f	e/0	a/0
g	c/1	-/-
h	b/1	a/1

Compatibles = {ae, af, bg, de, ef, eg}

af compatible under de, de compatible under af, ef compatible under af

Solving graphically the covering/closure problem



Complete and closed graph.

This graph has groups {aef, de, bg}.

Solution to Problem 10.

It can be observed that the machine is realized in the next problem since this is the same machine. So, you minimize the machine and next encode it and realize with JK ffs. This is a smarter approach that realizing the non-minimized machine which would lead to too big problem.

The binate covering Problem.

Given is a set of symbols S. Given is a set of groups G such that (for each group $g_j \in G$) [$g_j \subseteq S$]

Select set $G1 \subseteq G$ such that:

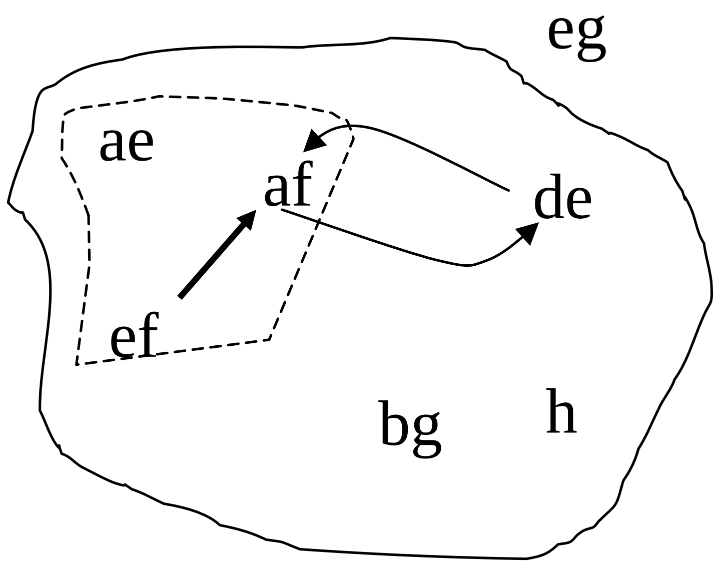
1) $\cup\, g_j \in G1 = S$

2) $[\,(g_j \in G1\,)$ and $\mathrm{IMPLY}(g_j\,,\, g_r)\,] \Longrightarrow g_r \in G1$

		a	b	c	d	e	f	g	h	af	de	aef
A	ae	x				x						
B	af	x				x					o	
C	ef					x	x			o		o
D	aef	x				x	x				o	
E	eg					x		x				
F	de				x	x				o		o
G	bg		x					x				
H	h								x			

Covering part

Closure part



Reduced to satisfiability formula

(A+B+D)G(A+F)(B+C+D+E+F)(C+D)
(E+G) H

* (B-->F)(C-->B+D)(D-->F)(F-->B+D)

Problem 11. Realization of Synchronous Finite State Machines.

a	d/-	f/0
b	c/1	h/0
c	d/0	h/0
d	c/0	a/0
e	-/-	f/0
f	e/0	a/0
g	c/1	-/-
h	b/1	a/1

- (a) Given is machine from the left. Realize this machine using D flip-flops and the excitation and output functions that would depend on the minimum total number of variables.
- (b) If you cannot minimize all these functions, try to minimize at least some and prove that you minimize them by some systematic method.
  - You do not have to prove that your solution is optimum but you must proceed rationally using the methods shown in class.
  - While solving this problem think about all FSM optimizing methods discussed in our class.
- (c)Using the final schematics demonstrate that you indeed minimized the number of arguments of <u>some</u> functions. Write specifically which ones. Prove with your comments that you understand the principles of state assignment and not only the procedure.

Solution to Problem 11.

a	d/-	f/0
b	c/1	h/0
c	d/0	h/0
d	c/0	a/0
e	-/-	f/0
f	e/0	a/0
g	c/1	-/-
h	b/1	a/1

{afe, bg, de} are groups of compatible states from the minimization of states. Encode A=afe, B=bg, C=de, D=c, E=h leads to the table below:

A	C/0	A/0
B	D/1	E/0
C	D/0	A/0
D	C/0	E/0
E	B/1	A/1

P0(0) = (BE, ACD)
P0(1) = (E, ABCD)

Pc(0) = BCD
Pc(1) = AE

Thus 1--> (BCD, AE) and this partition should be taken to simplify excitation function. The respective ff will depend only on input signals.

P(0)=(AD, BC, E), P(1)=(ACE, BD)

P(0) generates (AD,BCE), (ADE,BC) and (E,ABCD) which was already found. Assume (ACE, BD). Calculate product:

(BCD, AE) * (ACE, BD) = (C, BD, AE) thus B and D should be separated and states A and E should be separated. This can be done by P0(0) or (AD, BCE)

Solution to Problem 11.

A	C/0	A/0
B	D/1	E/0
C	D/0	A/0
D	C/0	E/0
E	B/1	A/1

T1= (BCD, AE)=(1,0)

T2= (AD, BCE)=(0,1)

T3= (ACE, BD)=(0,1)

A=000 since it occurs most times

A=000,
B=111,
C=110,
D=101,
E=010

We expect T1 to cause X $^+$ depending only on input signals because of 1-->T1

We expect T2 to simplify the second column

XYZ \ a	0	1
A=000	110,0	000,0
001	-	-
011	-	-
E=010	111,1	000,1
C=110	101,0	000,0
B=111	101,1	010,0
D=101	110,0	010,0
100	-	-

X $^+$ Y $^+$ Z$^+$ =
T1,T2,T3

X$^+$ = a’ which confirms our expectation for FF encoded with T1.

Y$^+$ =Za+ X’ a’ + Y’a’ which confirms our expectation for simple first column (for a’). This is because of P(0)

Z$^+$ =Y a’ which confirms our expectation for simple second column (for a).

Solution to Problems 11 and 10.

Realization with JK FFs.

Output z

Q1 Q2 Q3 \ a	0	1
A=000	110,0	000,0
001	-	-
011	-	-
E=010	111,1	000,1
C=110	101,0	000,0
B=111	101,1	010,0
D=101	110,0	010,0
100	-	-

$X^+ Y^+ Z^+ =$ T1,T2,T3

Using standard methods for JK FFs we get:

J1 = a’

K1 = a

J2 = a’

K2 = Q1 a’ + a Q3’

J3 = Q2 a’

K3 = a + Q2’ = (Q2 a’)’

Problem 12. State Assignment of Synchronous Finite State Machines.

a b	00	01	11	10
A	A	A	B	C
B	B	A	C	C
C	B	C	C	B
D	A	—	B	B

Transition table

a b	00	01	11	10
A	01	11	11	11
B	01	01	1-	01
B	01	11	-1	01
C	01	--	-1	11

Output table

Given is machine M2 described with the following transition and output tables

Solution to Problem 12.

Partitions from Transition Table

P(00)={AD,BC}

P(01)={AB,C,(D)}

P(11)={AD,BC}

P(10)={AB,CD}

a b

	00	01	11	10
A	A	A	B	C
B	B	A	C	C
C	B	C	C	B
D	A	—	B	B

a b

	00	01	11	10
A	01	11	11	11
B	01	01	1-	01
C	01	11	-1	01
D	01	--	-1	11

Observe that there is only one set of proper partitions that are determined by partitions determined from transition table:

T1 = {AB,CD}

T2 = {AD,BC}

Partitions from the Output Table:

Po(00)=1

Po(01)={AC,B,(D)}

Po(11)=1 (the simplest out of many)

Po(10)={AD,BC}

Thus select T2 and T3={AC,BD}

Concluding, I select partition T1 for sure and T2 since it will simplify two columns of transitions.

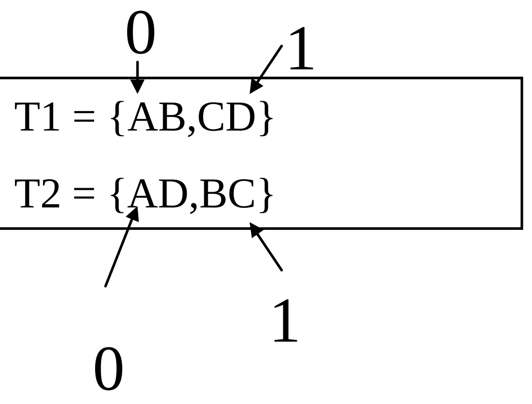
Another choice would be to select T1 and T3.

Solution to Problem 12.

T1 = {AB,CD}

T2 = {AD,BC}

selected



Thus encoding is:

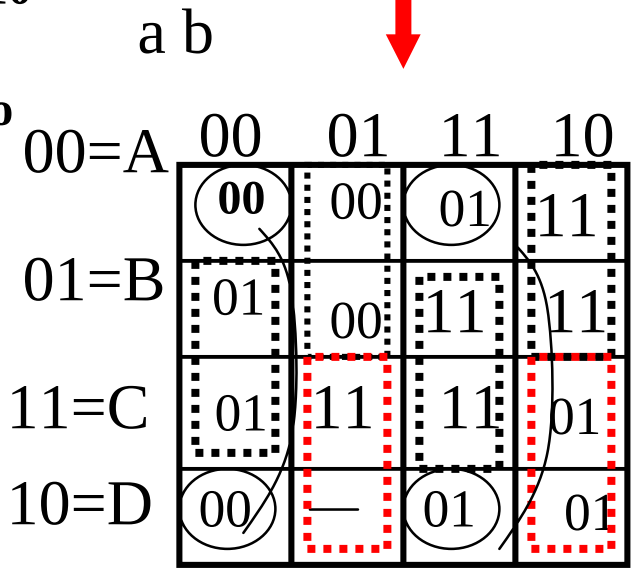
A=00,B=01,C=11,D=10

a b	00	01	11	10
A	A	A	B	C
B	B	A	C	C
C	B	C	C	B
D	A	—	B	B

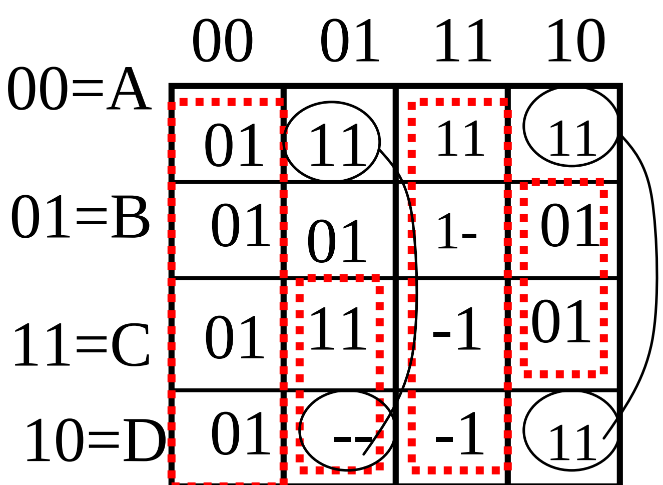
a b	00	01	11	10
A	01	11	11	11
B	01	01	1-	01
C	01	11	-1	01
D	01	--	-1	11

This encoding leads to maps at right:

The groups responsible for SOP minimization are shown



a b	00	01	11	10
00=A	00	00	01	11
01=B	01	00	11	11
11=C	01	11	11	01
10=D	00	—	01	01



	00	01	11	10
00=A	01	11	11	11
01=B	01	01	1-	01
11=C	01	11	-1	01
10=D	01	--	-1	11

Solution to Problem 12.

As we see, simplified groups are composed of groups from previous slide that are in turn found from partitions

In red

Q1+ = Q1’ab’+Q2ab+ Q1a’b

In blue

Q2+ = a + Q2b’+Q1a’b

repeated

In green

z1 = ab + (a+b)Q2’+Q1b

In black

z2 = 1

Q1+ Q2+

Q1 Q2 \ a b	00	01	11	10
00=A	**00**	00	01	11
01=B	01	00	11	11
11=C	01	11	11	01
10=D	00	—	01	01

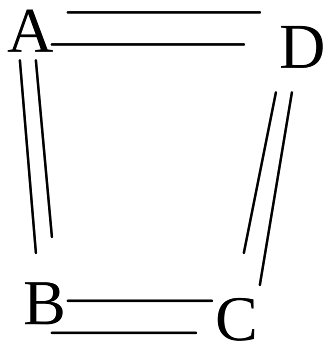
z1 z2

Q1 Q2 \ a b	00	01	11	10
00=A	01	11	11	11
01=B	01	01	1-	01
11=C	01	11	-1	01
10=D	01	--	-1	11

a b	00	01	11	10
00=A	**00**	00	01	11
01=B	01	00	11	11
11=C	01	11	11	01
10=D	00	—	01	01

a b	00	01	11	10
00=A	01	11	11	11
01=B	01	01	1-	01
11=C	01	11	-1	01
10=D	01	--	-1	11

Solution to Problem 12.rule-based

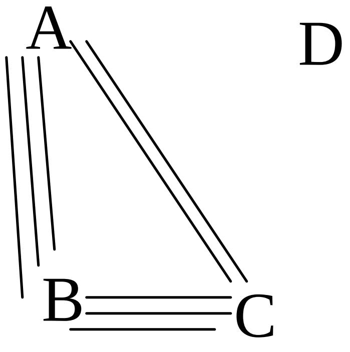


transitions

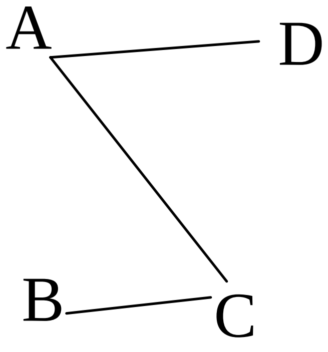
	ab 00	01	11	10
A	A	A	B	C
B	B	A	C	C
C	B	C	C	B
D	A	—	B	B

	ab 00	01	11	10
A	01	11	11	11
B	01	01	1-	01
C	01	11	-1	01
D	01	--	-1	11

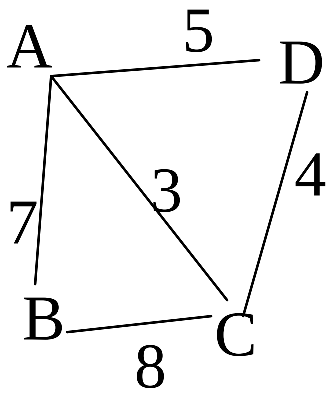
Assume that transitions are twice more important we get:



Next states



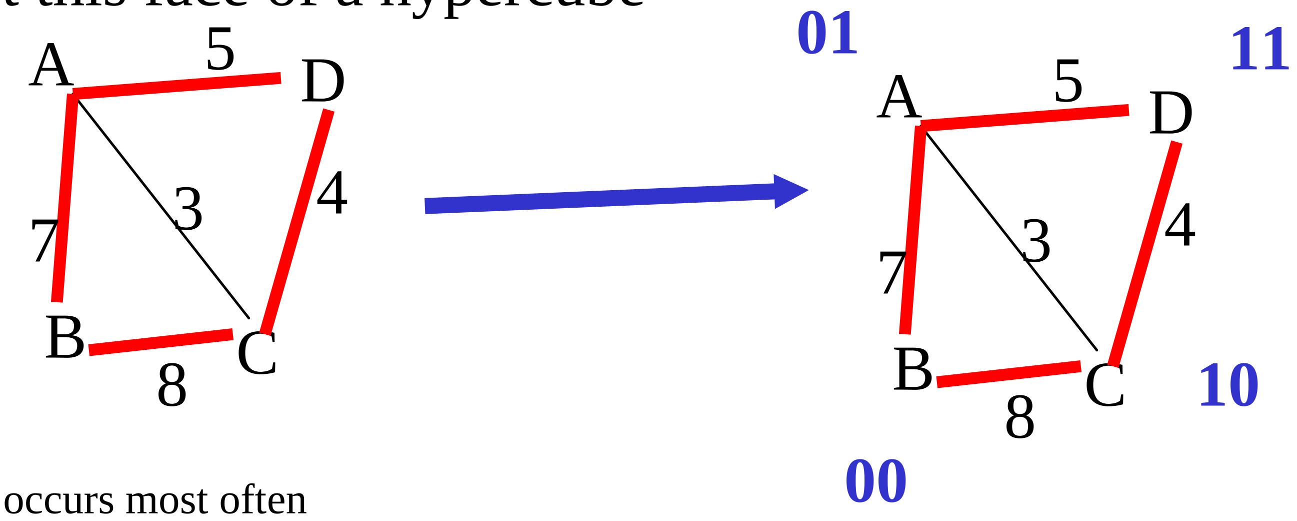
outputs



Total graph

Solution to Problem 12.rule-based

We get this face of a hypercube

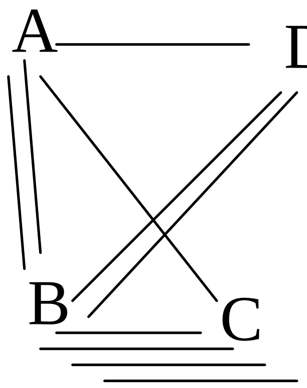


B occurs most often
so is encoded by 00

Which leads to encoding
A=01, B=00,C=10, D=11

This is the same set of partitions as before, so the result is very similar in terms of realization cost.

Solution to Problem 12.hypercube



a b

	00	01	11	10
A	A	A	B	C
B	B	A	C	C
C	B	C	C	B
D	A	—	B	B

a b

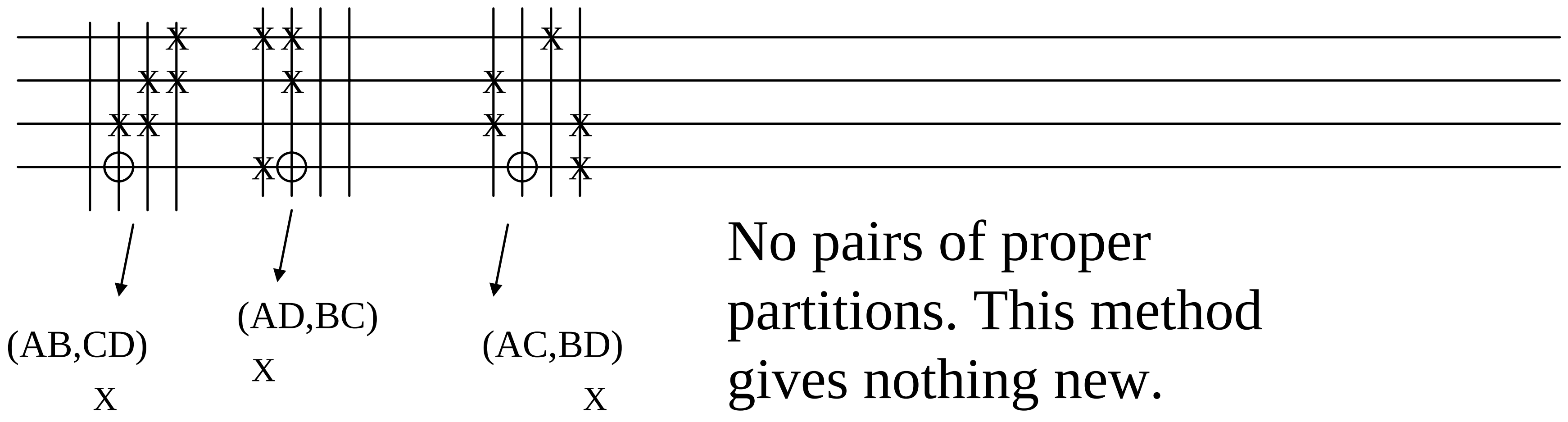
	00	01	11	10
A	01	11	11	11
B	01	01	1-	01
C	01	11	-1	01
D	01	--	-1	11

This leads again to solutions {T1,T2} and {T1,T3}

Solution to Problem 12.multi-line

a b	00	01	11	10
A	A	A	B	C
B	B	A	C	C
C	B	C	C	B
D	A	—	B	B

a b	00	01	11	10
A	01	11	11	11
B	01	01	1-	01
C	01	11	-1	01
D	01	--	-1	11



No pairs of proper partitions. This method gives nothing new. {AD,BC} is best.{AC,BD} worst. So solution is T1,T2

Solution to Problem 12.

Q1+ Q2+

Q1 Q2 \ a b	00	01	11	10
00=A	**00**	00	01	11
01=B	01	00	11	11
11=C	01	11	11	01
10=D	00	—	01	01

Q Q+	J K
0 0	0 -
0 1	1 -
1 0	- 1
1 1	- 0

Use method of bold symbols (shown in red)

J1=a(Q2+b')

K1=Q2'+b'

J1 K1

Q1 Q2 \ a b	00	01	11	10
00=A	**0**	0	0	1
01=B	0	0	1	1
11=C	0	1	1	0
10=D	0	—	0	0

K2=Q1'a'b

J2=a

Q1 Q2 \ a b	00	01	11	10
00=A	**0**	0	1	1
01=B	1	0	1	1
11=C	1	1	1	1
10=D	0	—	1	1

Explanation why this is good is the same as before

Problem 13. Scheduling and Register Allocation.

(a) How many registers to store the six variables (u1 to u6) ?

Inputs: v1, v2, v3, v4

Outputs: u5, u6

op1: u1 <-- v1 + v2

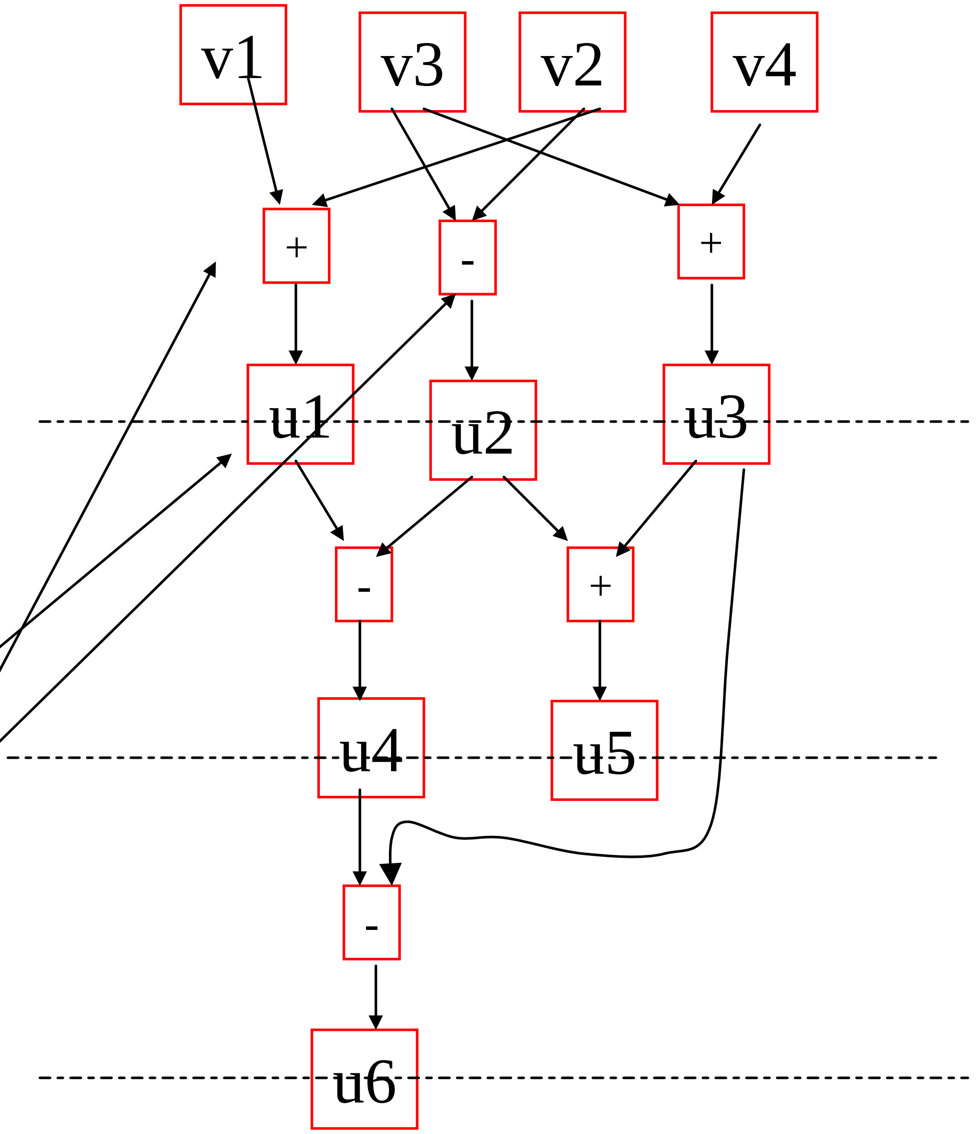
op2: u2 <-- v3 - v2

op3: u3 <-- v3 + v4

op4: u4 <-- u1 - u2

op5: u5 <-- u2 + u3

op6: u6 <-- u4 - u3

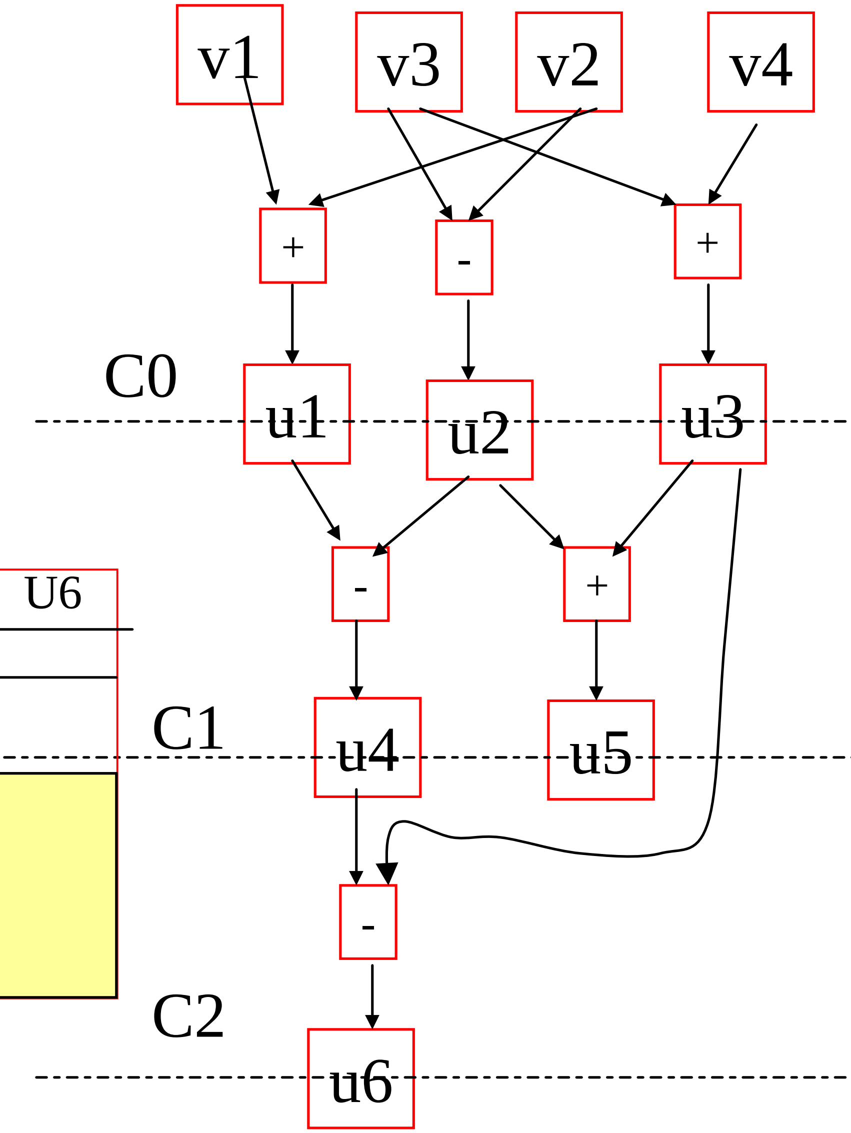


ASAP needs three clock cycles

3 registers, 2 adders, 1 subtractor.

Life time analysis of variables

cycle	U 1	U 2	U 3	U 4	U 5	U6
C 0	▨	▨	▨			
C 1			▨	▨	▨	
C 2					▨	▨



How many registers to store the six variables (u1 to u6) ?

Inputs: v1, v2, v3, v4

Outputs: u5, u6

op1: u1 <-- v1 + v2

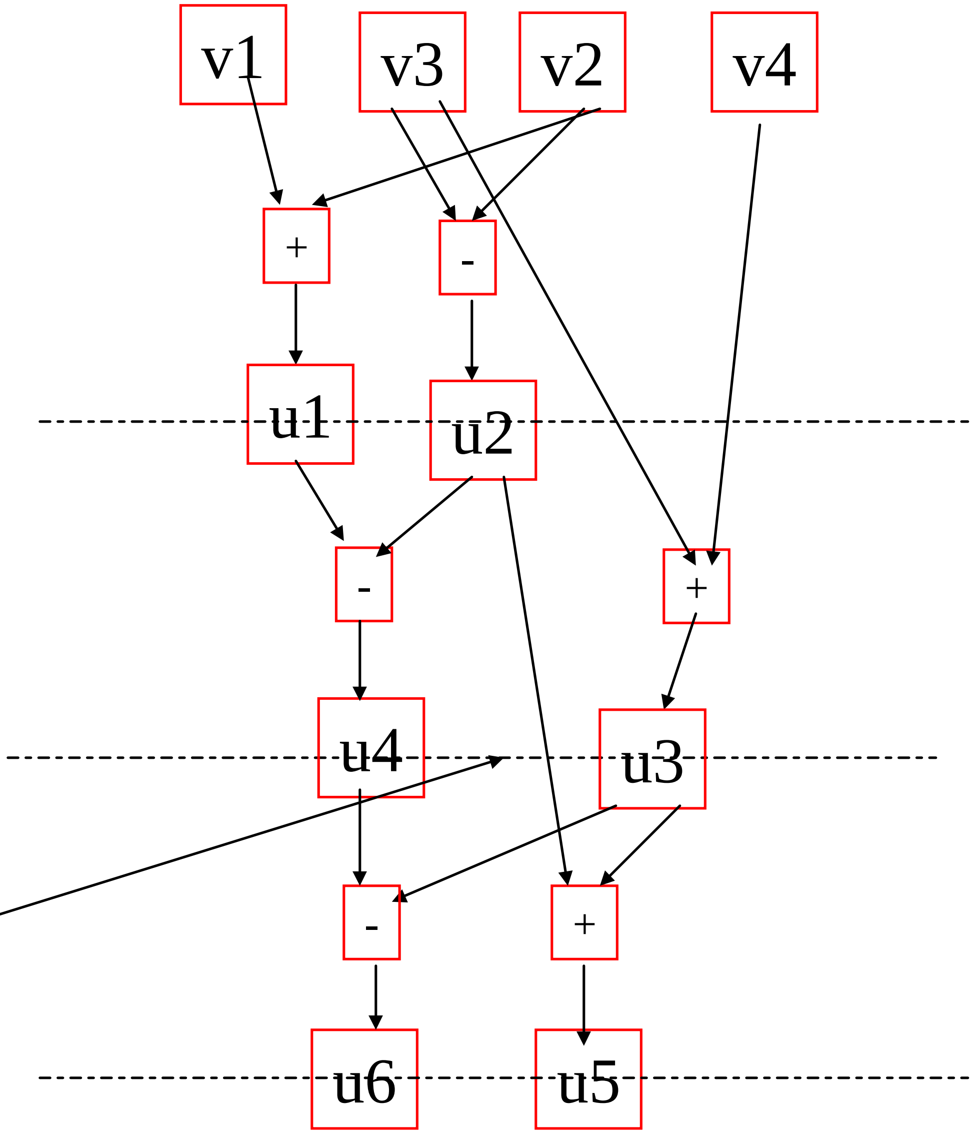
op2: u2 <-- v3 - v2

op3: u3 <-- v3 + v4

op4: u4 <-- u1 - u2

op5: u5 <-- u2 + u3

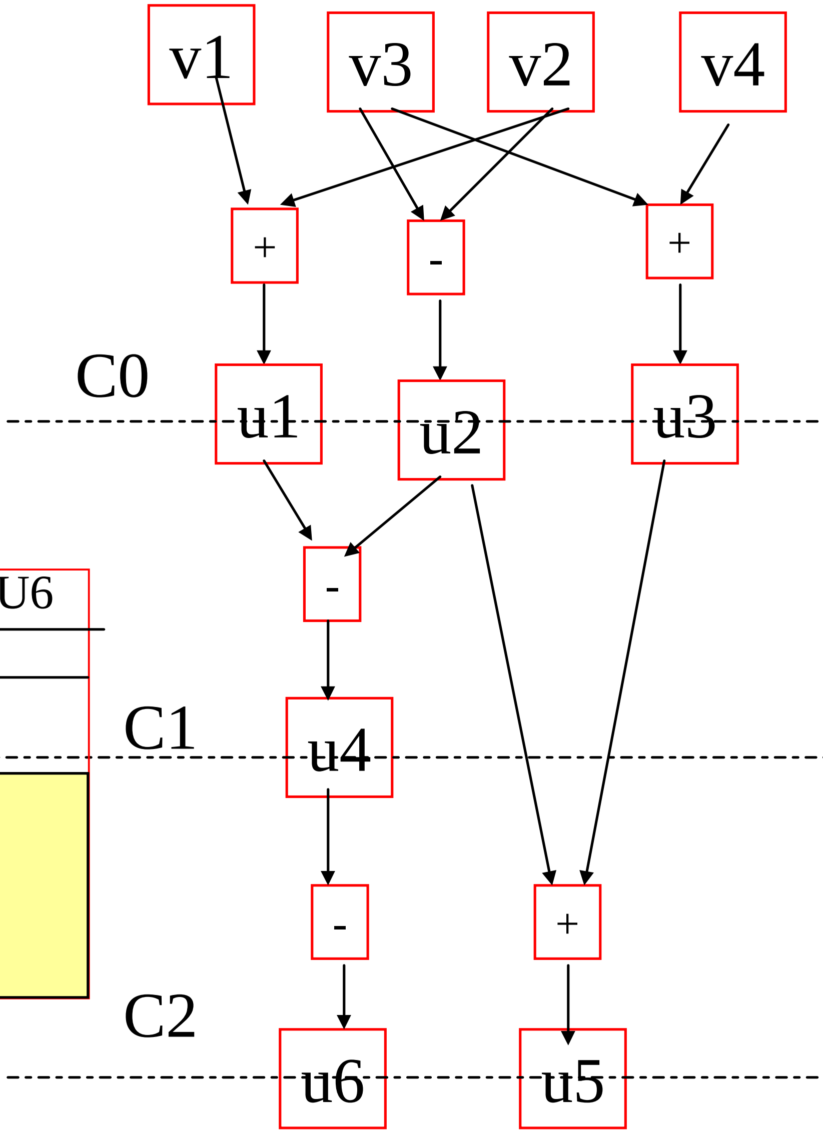
op6: u6 <-- u4 - u3



ALAP needs three clock cycles

3 registers, 1 adder, 1 subtractor.

cycle	U 1	U 2	U 3	U 4	U 5	U6
C 0	🟨	🟨	🟨			
C 1			🟨	🟨	🟨	
C 2					🟨	🟨



One more solution: 3 cycles, 3 registers, 2 adders, 1 subtractor.

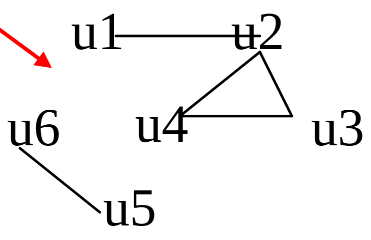
This solution is not better. From now on, several variants of solving this problem are possible and I just show one of them.

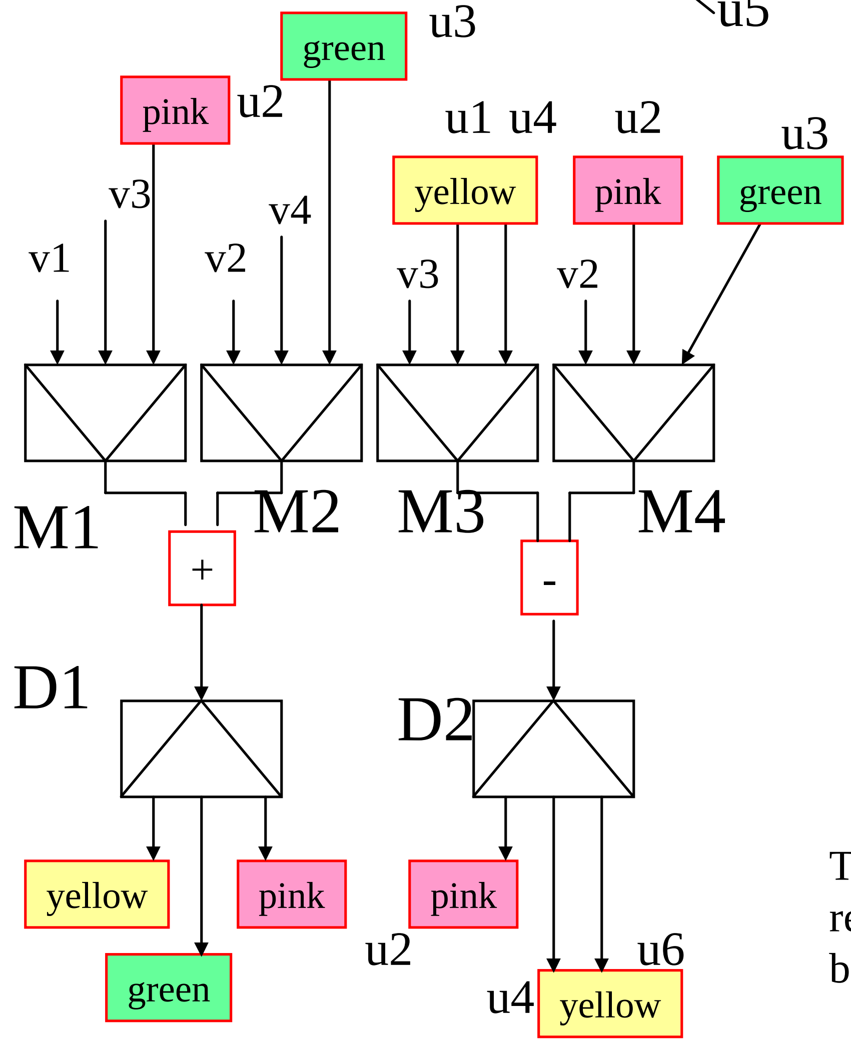
I select ALAP. Incompatibility graph is obviously 3-colorable, not less.

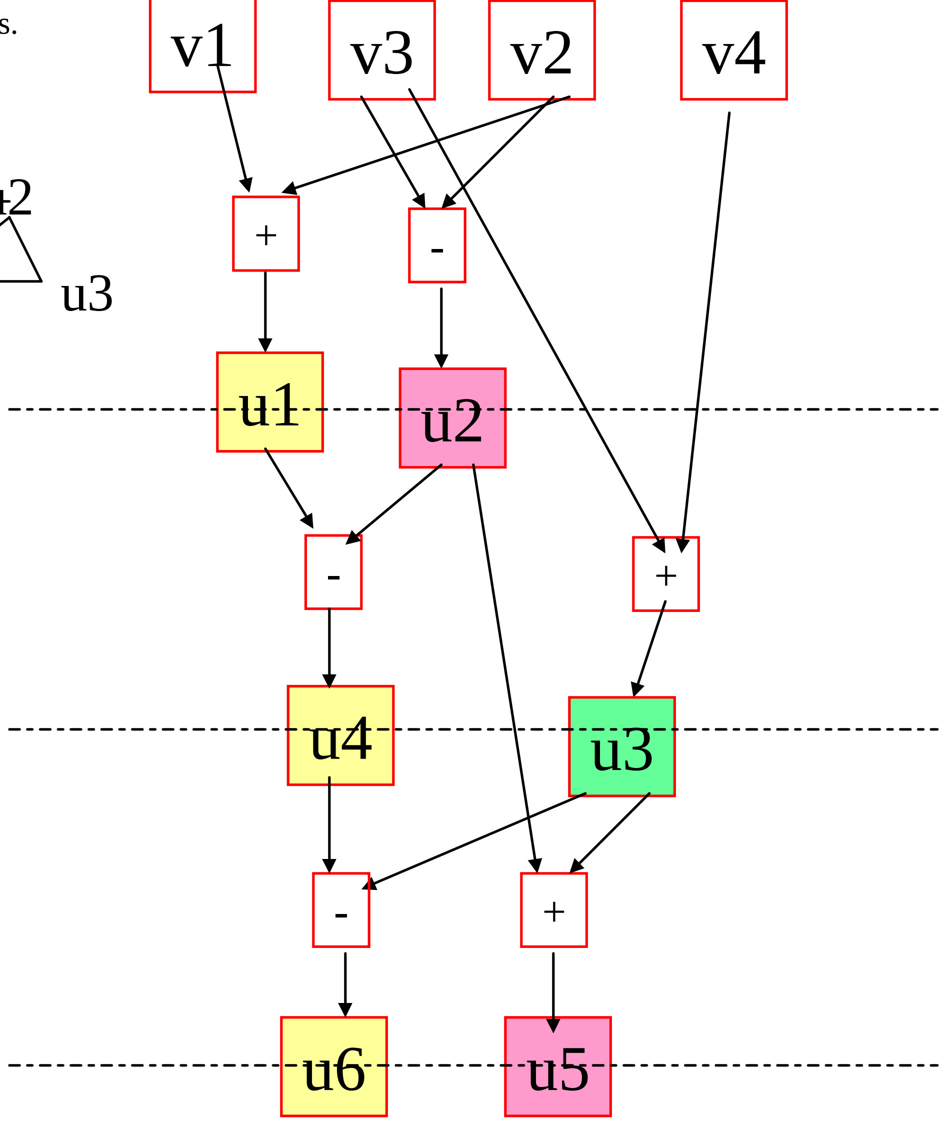
Registers u1, u4 and u6 are colored yellow.

Registers u2, u5 are colored pink

Register u3 is colored green.





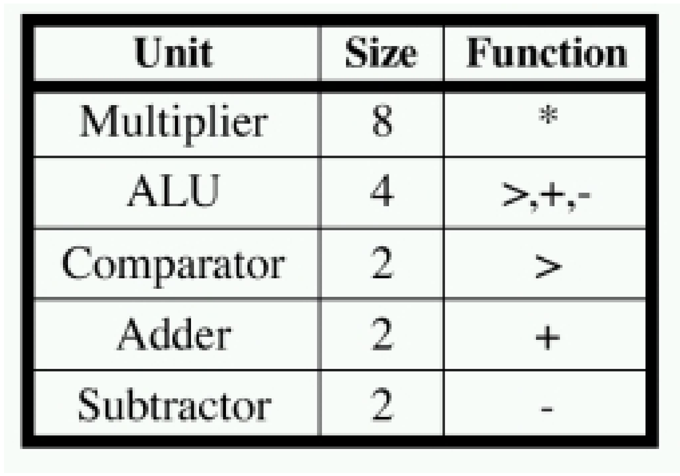


The schematics from left can be realized in memories, register files or registers. Mux M3 and Demux D2 can be simplified by reducing control variables to one each.

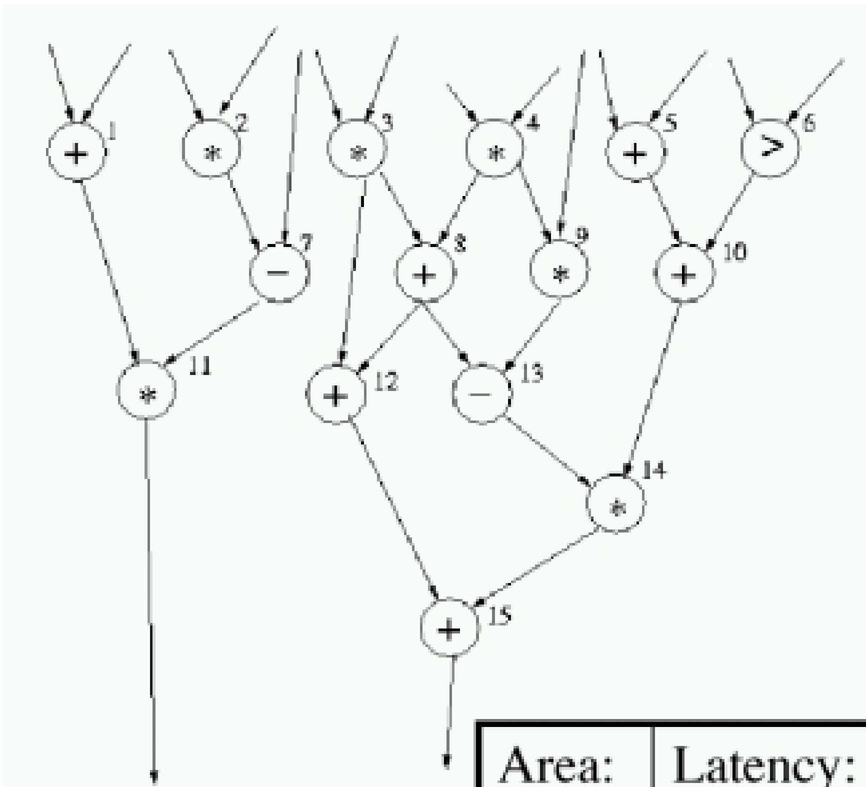
Problem 14

Schedule and allocate resources for each
operation, minimizing area and latency as much as possible.

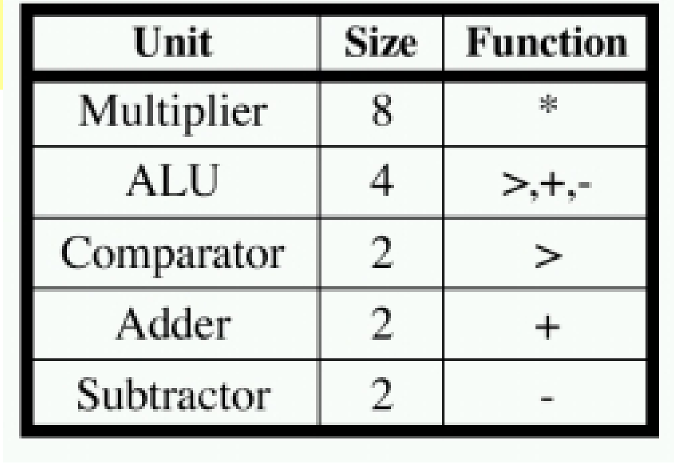
- (a) Find the solution with the smallest area
- (b) Find the solution with the smallest latency.
  - In points (a) and (b) you are not required to give an optimal solution, since that may prove to be more difficult than can be done in a reasonable amount of time.
  - But you have to demonstrate that your reasoning is correct, you understand the problem and know at least some scheduling and allocation methods.



Unit	Size	Function
Multiplier	8	*
ALU	4	>,+,-
Comparator	2	>
Adder	2	+
Subtractor	2	-



Solution to Problem 14



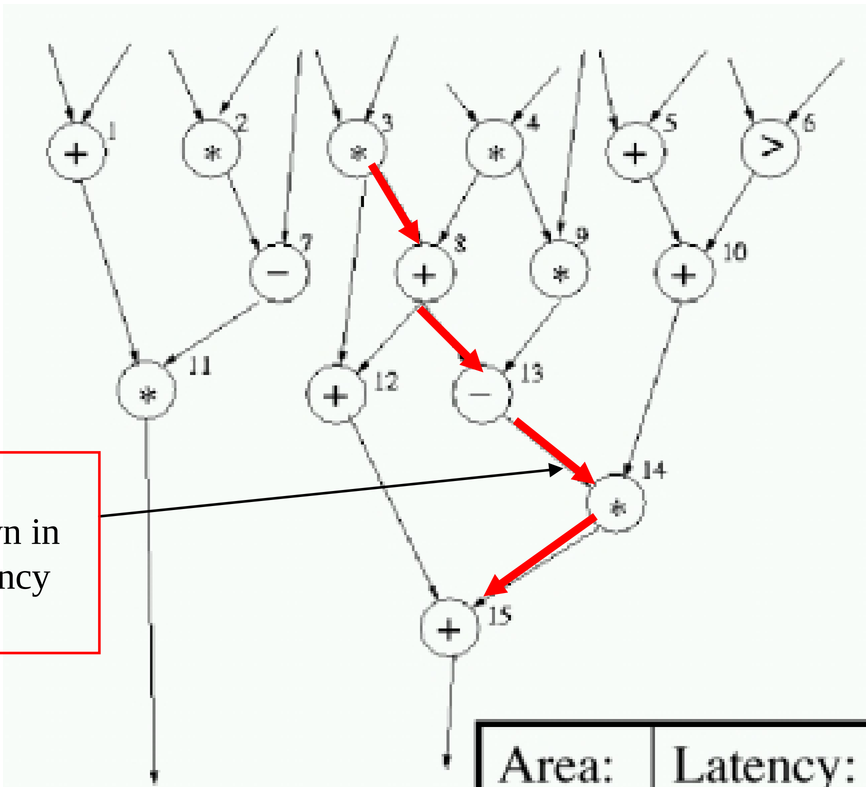
Unit	Size	Function
Multiplier	8	*
ALU	4	>,+,-
Comparator	2	>
Adder	2	+
Subtractor	2	-

Obviously only multiplier is the only operation for *. ALU withcost 4 does the same as comparator, adder and subtractor of cost 6. So the smallest area is 8+4=12.

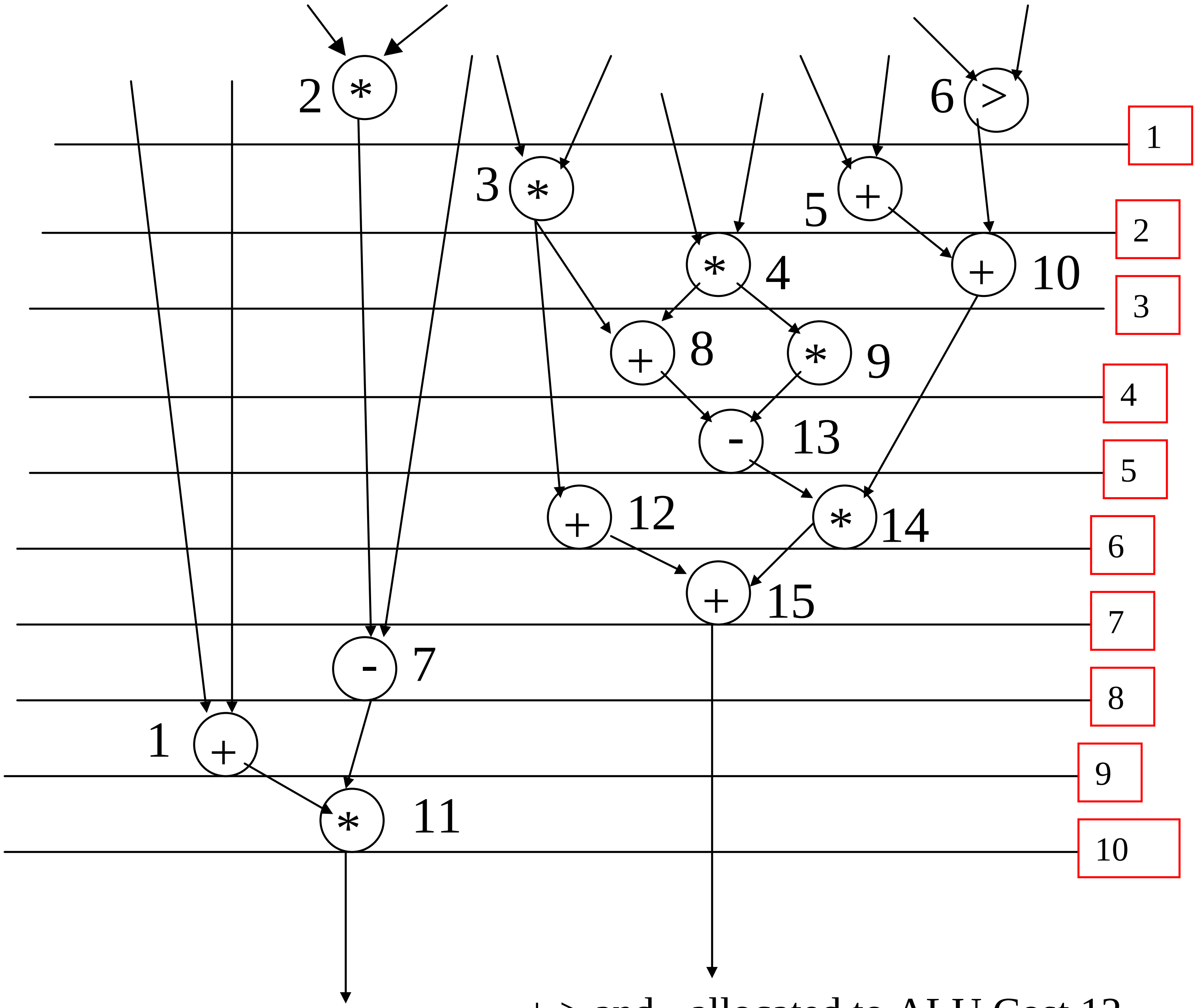
We will need muxes for scheduling. See next slide.

In this figure the latency is 5 (the shortest possible - ASAP), but the cost is 3multipliers*8+2adders*2+1comparator*2+1 subtractor*2 =24+4+2+2=32

Maximum dependency shown in red limits the latency to 5.

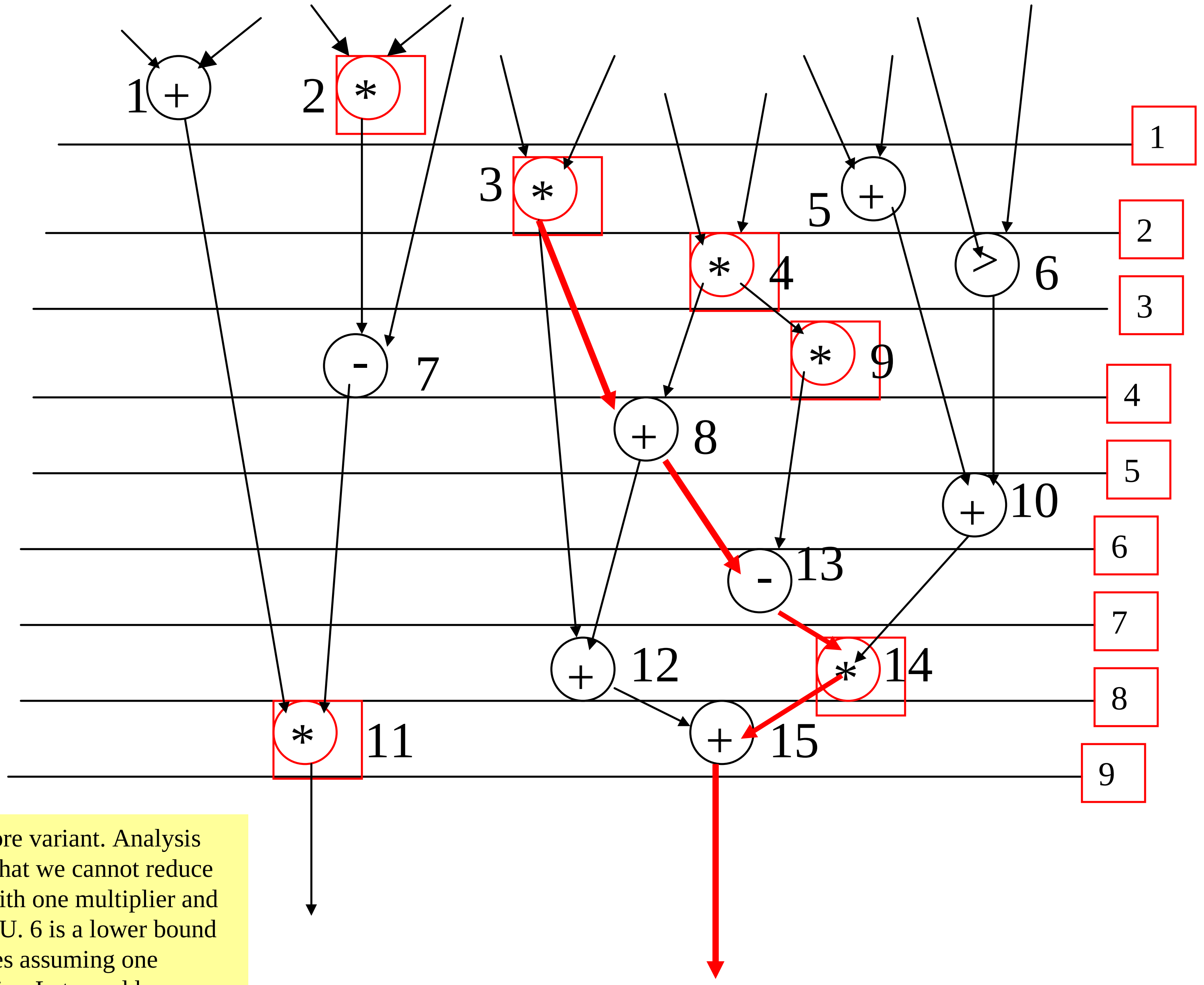


So our trade-off is to find not slower than 5 with cost better than 12. Let us first find small cost solution.



+,> and - allocated to ALU.Cost 12.

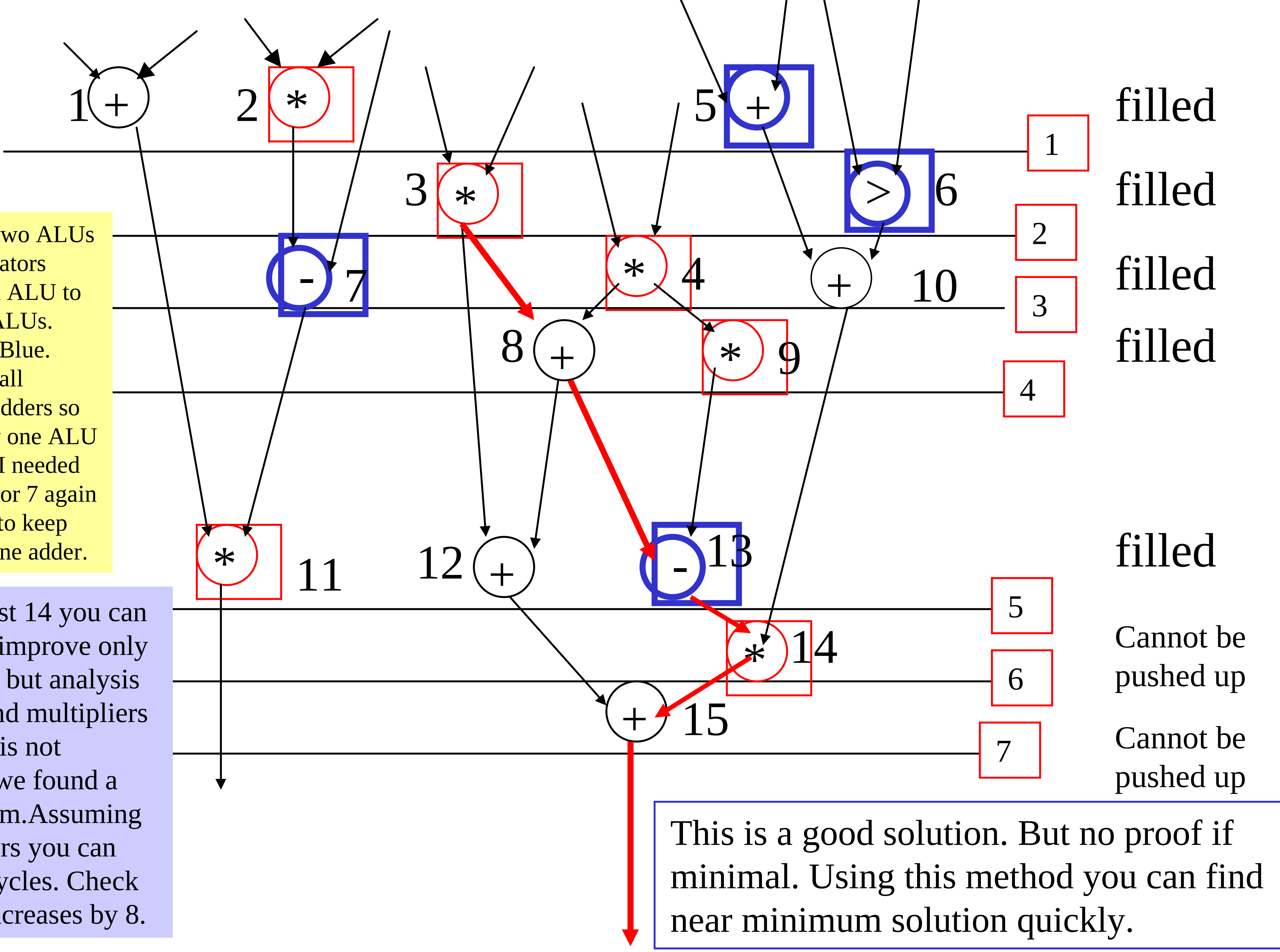
10 cycles.



One more variant. Analysis shows that we cannot reduce more with one multiplier and one ALU. 6 is a lower bound of cycles assuming one multiplier. Let us add more ALUs

+,> and - allocated to ALU.Cost 12.

9 cycles.



Now I assume two ALUs and I push operators implemented in ALU to top to fill two ALUs. ALU shown in Blue. Then I see that all remaining are adders so that I need only one ALU and one adder. I needed to shift subtractor 7 again to cycle below to keep one ALU and one adder.

Assuming cost 14 you can theoretically improve only by one cycle, but analysis of red path and multipliers shows that it is not possible. So we found a local minimum.Assuming two multipliers you can reduce to 5 cycles. Check it. But cost increases by 8.

This is a good solution. But no proof if minimal. Using this method you can find near minimum solution quickly.

+,> and - allocated to ALU.Cost 12+2=14.

7 cycles.

Problem 15.
From Verilog to sequential logic circuit

```
module DIFFEQ (x, y, u , dx, a, clock, start);
input [7:0] a, dx;
inout [7:0] x, y, u;
input clock, start;
reg [7:0] xl, ul, yl;
always
   begin
   wait ( start);
   while ( x < a )
       begin
        xl = x + dx;
       ul = u - (3 * x * u * dx) - (3 * y * dx);
       yl = y + (u * dx);
      @(posedge clock);
      x = xl; u = ul ; y = yl;
   end
endmodule
```

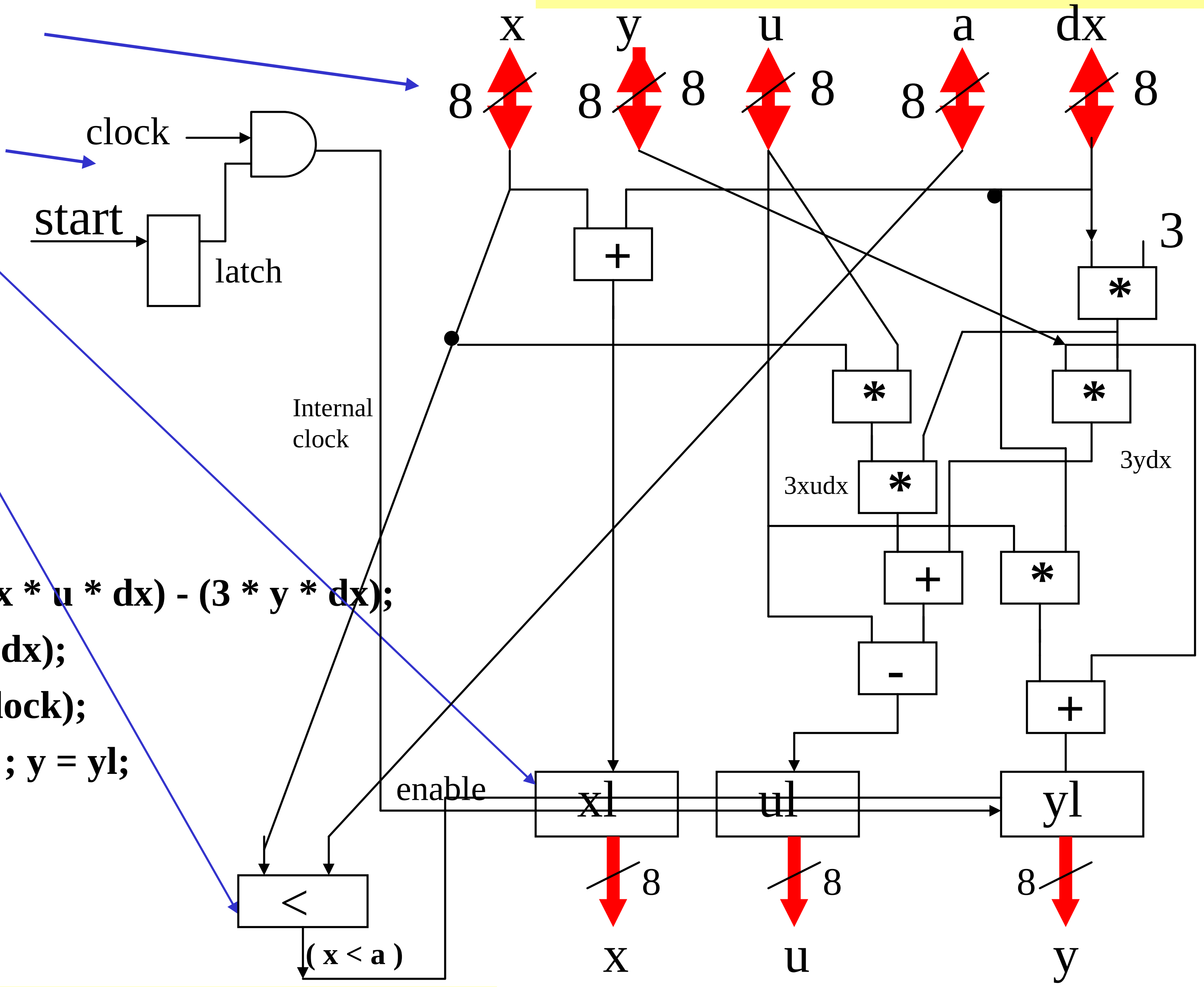
(a) Design the complete Data Path and Controller for this example. Any method from the class is applicable for any part of the complete design procedure.

(b) Explain your selections of methods and design decisions.

(c)Verify your solution.

Solution to Problem 15.

```
module DIFFEQ (x, y, u , dx, a, clock, start);
input [7:0] a, dx;
inout [7:0] x, y, u;
input clock, start;
reg [7:0] xl, ul, yl;
always
  begin
  wait ( start);
  while ( x < a )
     begin
      xl = x + dx;
      ul = u - (3 * x * u * dx) - (3 * y * dx);
      yl = y + (u * dx);
     @(posedge clock);
     x = xl; u = ul ; y = yl;
  end
endmodule
```



Circuit optimized for speed. Some standard transformations of data path used. I selected this method to avoid time-consuming design of a control unit. Standard register with enable is used.

x,u and y are the same io signals as on top