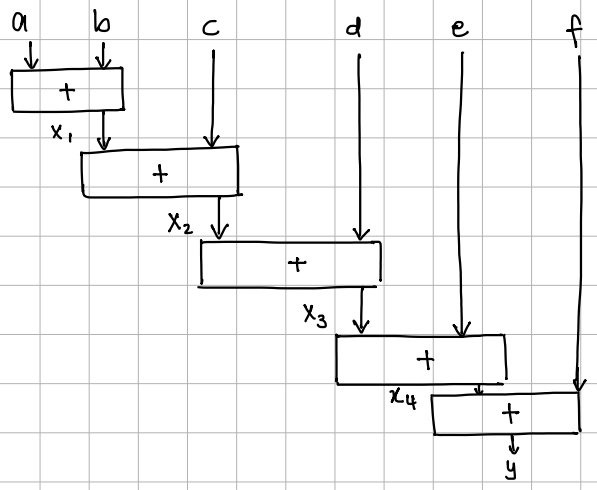
dataflow diagrams : data dependency graphs that illustrate a computation

It is useful when

1. exploring disadvantages or advantages in design
2. estimate area and performance of design
3. design datapath-centric systems, systematically

$y = a + b + c + d + e$



If a and b are a 8-bit input value,

$a + b = x_1$

The max. value of $a = b = 255$.

So, max value of $x_1 = 255 + 255$

$= 510$

size of x_1 is 9 bits.

In general, adding n bits with itself gives $n+1$

So, $x_2 = x_1 + c$

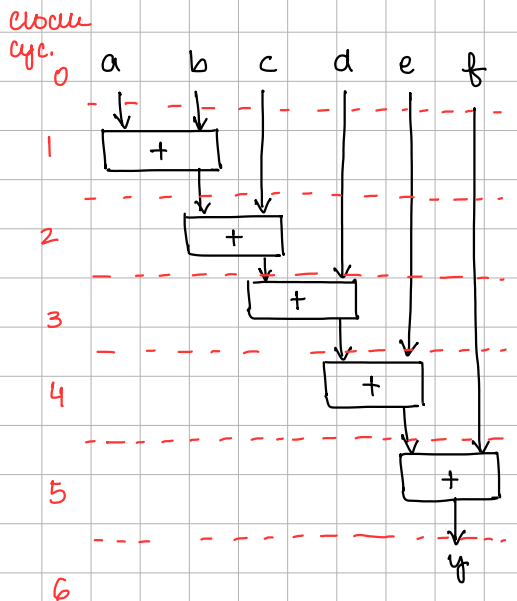
x_2	x_1	c
10	9	8
765	510	255

For $x_3 = x_2 + d$, x_3 can be 11 based on general rule

Based on max values, $X_3 \leq 1020$ so 10 bits still works!

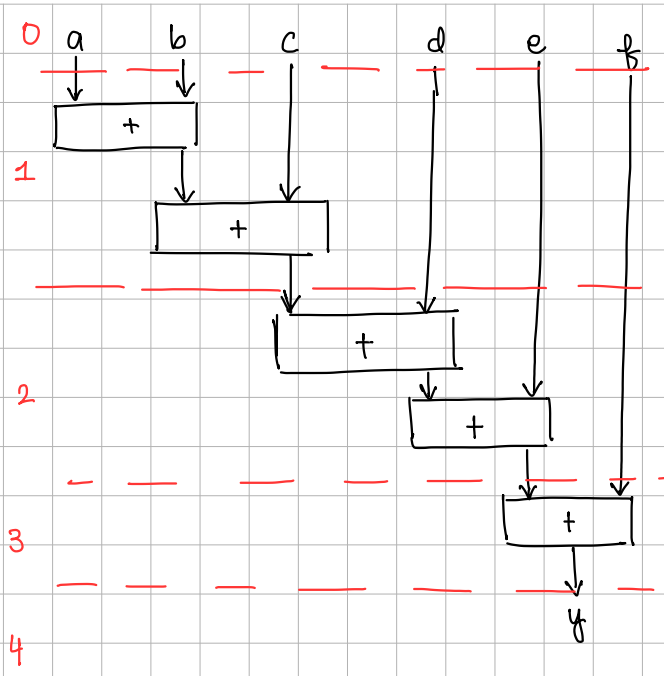
For $x_4 = x_3 + e$ = 11 bits are needed. For y, max (y) = 1530 so 11 bits work.

To find the latency,



→ 6 clock cycles latency

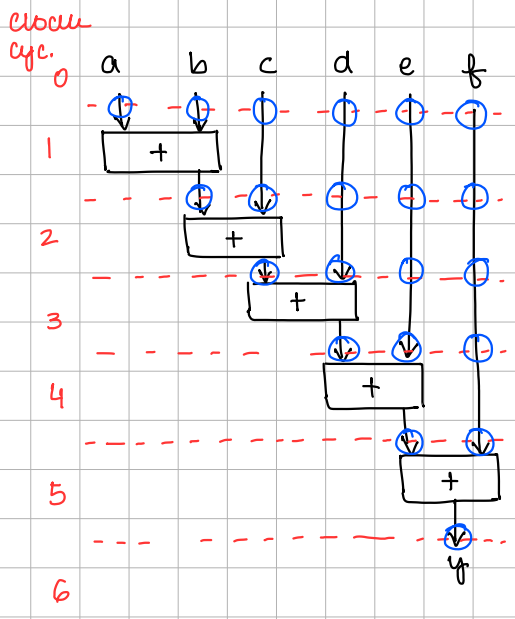
With appropriate control logic, this can be done with one adder, multiplexer to choose operand & FSM to control mux.



If we use more clock adder.
we can reduce latency per clock cycle.

Note that output takes it's own clock cycle

Every time we cross a clock cycle boundary, data value must be stored.
single binary val → D-ff
multiple binary vals → DR



Each O → register
Total: 21 registers

Performance Estimation

performance of a computation is

$$\text{Performance} \propto \frac{1}{t_{\text{execution}}}$$

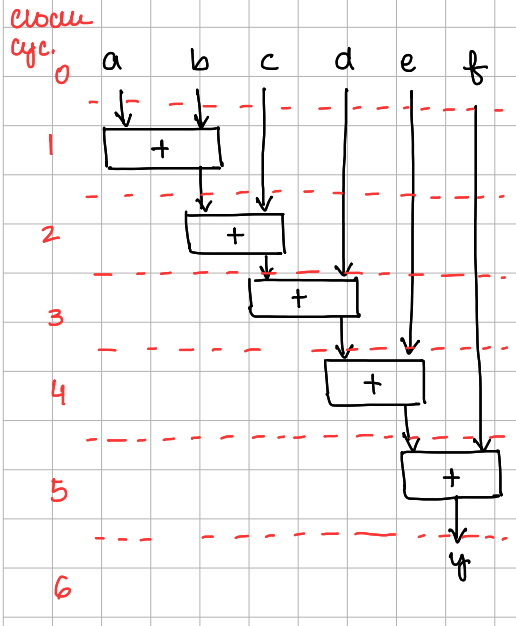
Longer processing time
worse performance

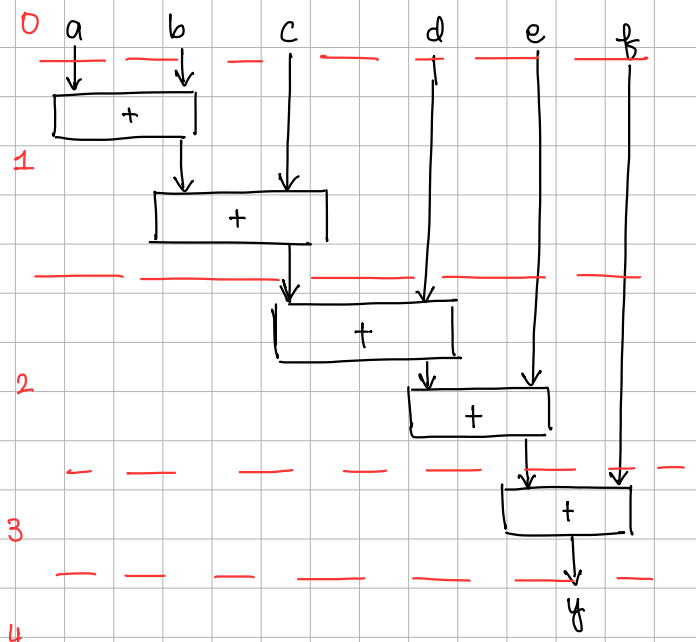
where

$t_{\text{execution}}$ = Latency × Clock period.

# of clock cycles required to produce output.

combinational circuit = 0 latency





Design for slowest step in the pipeline

Assume:

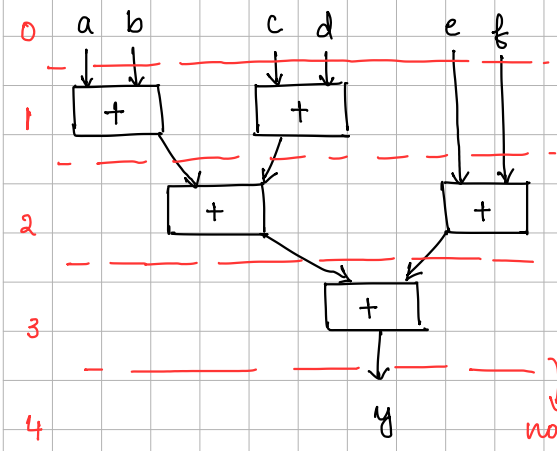
single adder = 8ns, register = 2ns

In first design, $t_{execution}$ = 10ns x 6
= 60ns

In this design, $t_{execution}$ = 18ns x 4
= 72ns

Area estimation

max # of blocks in clock cycle = total # of components.
max # of signals in clock cycle = total # of registers
max # of unconnected signal tails = # inputs
max # of unconnected signal heads = # outputs



not two adders cause simultaneous calculation

Inputs	6
Outputs	1
Registers	6
Adders	2
Clk Period	1 adders + 1 register.
Latency.	4

Designing with Dataflow Diagrams

Example:

functional requirement

output = (a x d) + c + (d x b) + b

performance requirement

max clk period: 1 reg delay + (2 addition delays + 1 multiply delay)

max latency: 4 clock periods.

resource requirement

max of three inputs & one output

max 2 adders

max 2 multipliers

unlimited regs.

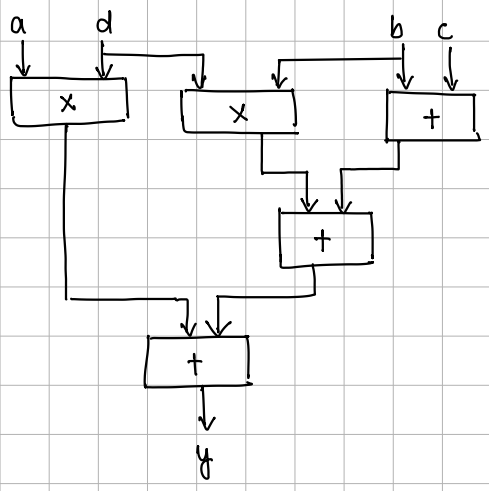
Design Process

1. Scheduling
2. I/O allocation
3. Initial High-level model
4. Register allocation
5. Datapath allocation
6. Multiplexer insertion to connect datapath comp.
7. FSM
8. Optimization

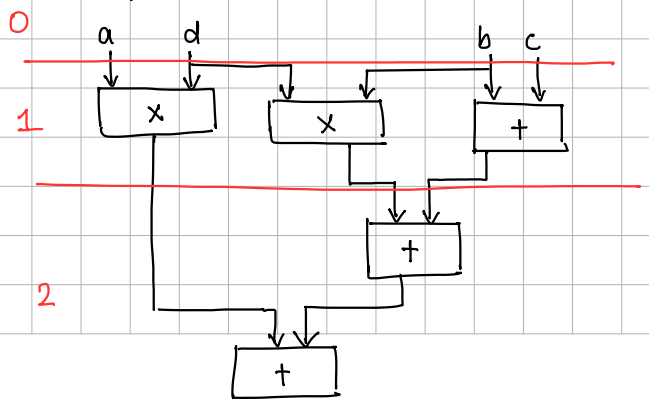
Scheduling

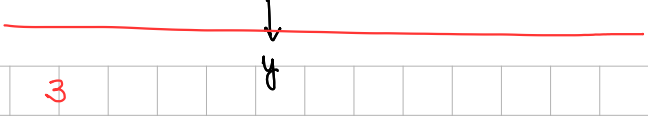
1. Create a data depenancy graph.

y = (a x d) + c + (d x b) + b

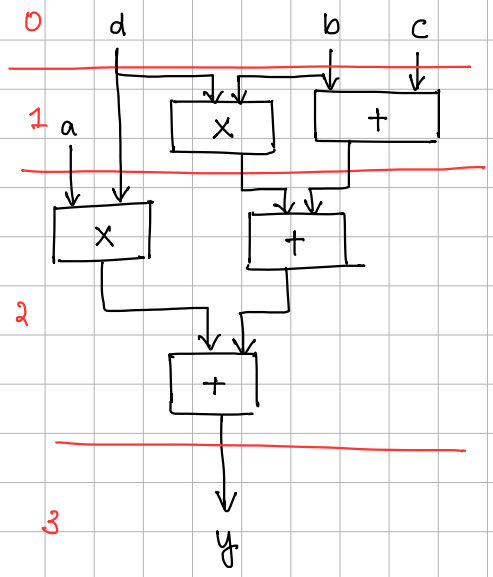


2. schedule operations into clock periods





3. reschedule to meet signal requirements.
   We start with 4 inputs ✗ violates 3 inputs rule.

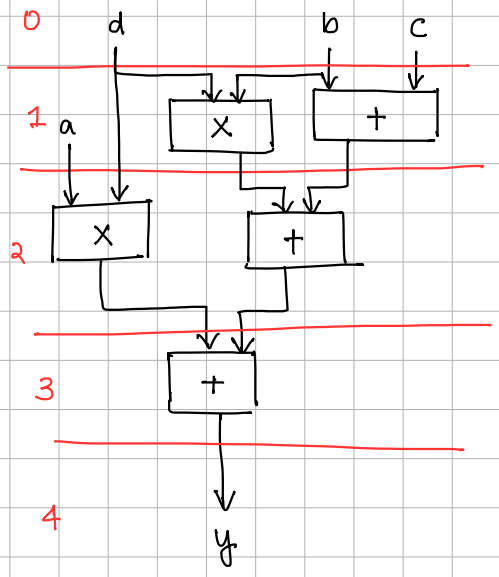


delaying b/d is bad cause they are involved in critical computations

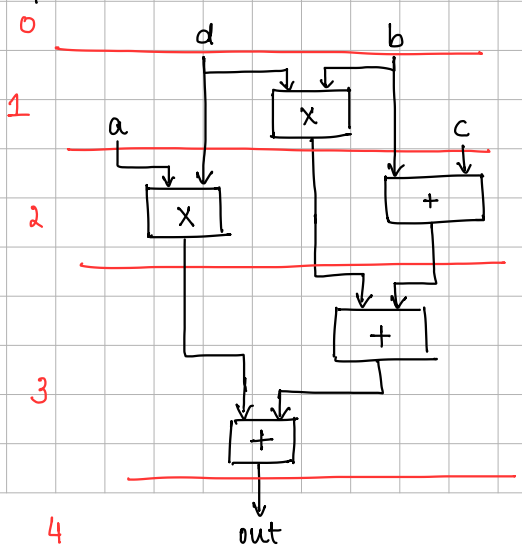
c is part of a longer chain

a is optimal.

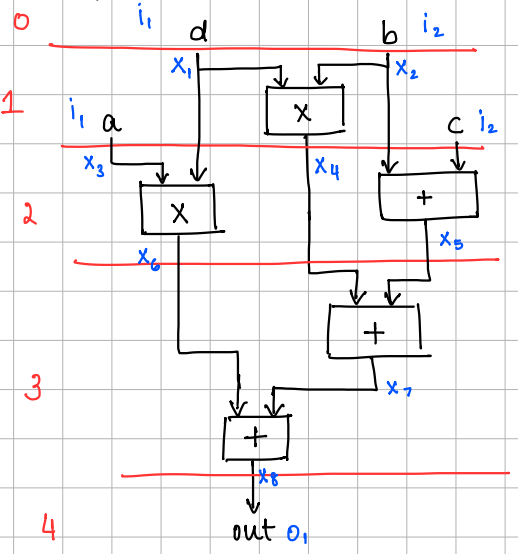
4. Fix clock period violation (clock period 2)



5. optimize.



Input / Output allocation

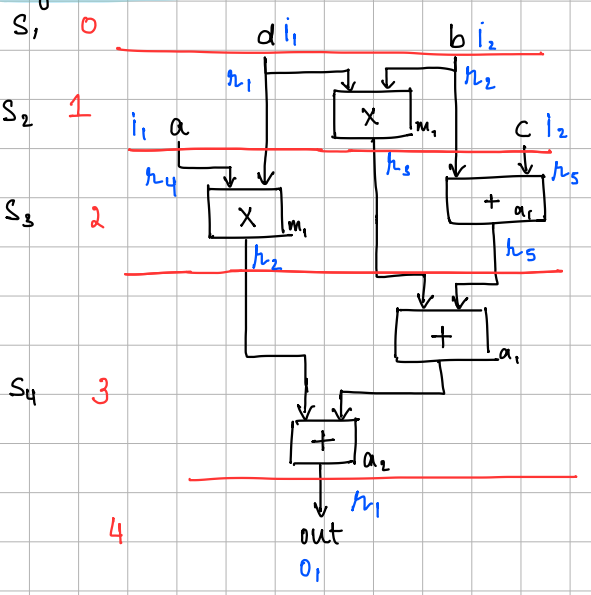


Initial High-level model → Build a FSM where

- every state repr. a clock boundary
- add a reset signal to ensure in first state.

See code, for FSM and TB.

Register allocation.



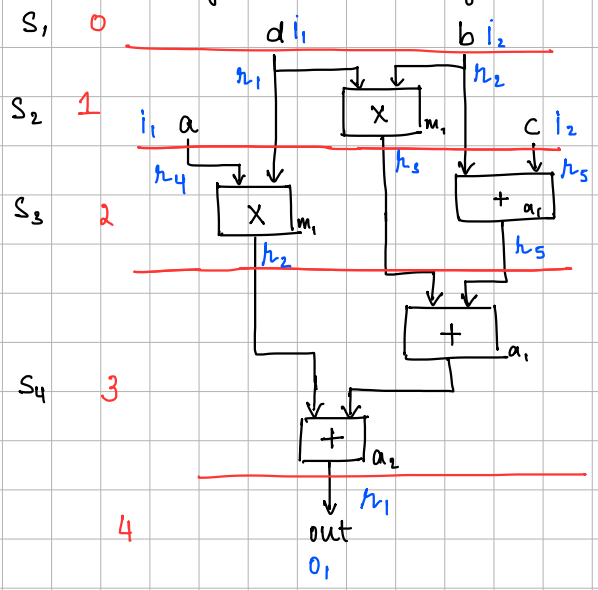
state	i_1	i_2	o_1
1	d	b	—
2	a	c	—
3	—	—	—
4	—	—	o_1

state	m_1	a_1	a_2
1	—	—	—
2	h_1 h_2	—	—
3	h_1 h_4	h_2 h_5	—
4	—	h_3 h_5	h_2 a_1
const.	h_1 No	No h_5	h_2 a_1

→ Based on this, we need 5 num and 3 clk-enabled regs

state	h_1	h_2	h_3	h_4	h_5
1	1 i_1	1 i_2	X	X	X
2	0 —	0 —	1 m_1	1 i_1	1 i_2
3	X	1 m_1	0 —	X	1 a_1
4	1 a_2			X	
constant?	Still uses input so continue clk: No	No	m_1	i_1	No: new input so 1

Final note: register size analysis.



For i_1 and i_2,

$r_1 = 16b$ $r_2 = 16b$

a c

For i_1 and i_2

$r_4 = 16b$ $r_5 = 16b$

For $i_1 \times i_2$, $r_2 + i_2$

$r_3 = 32$ $r_5 = 17$ bits

$r_2 = 32$

Then,

$r_1 = 34$ bits. { 2 more additions }

regs.

r_1	34b
r_2	32b
r_3	32b
r_4	16b
r_5	17b

wires.

m1a	16	16 in
m1b	16	
m1o	32	

a1a	32	32 inputs adder		a2a	32	33 inputs adder
a1b	17			a2b	33	
a1o	33			a2o	34	