Example of Scheduling and Allocation

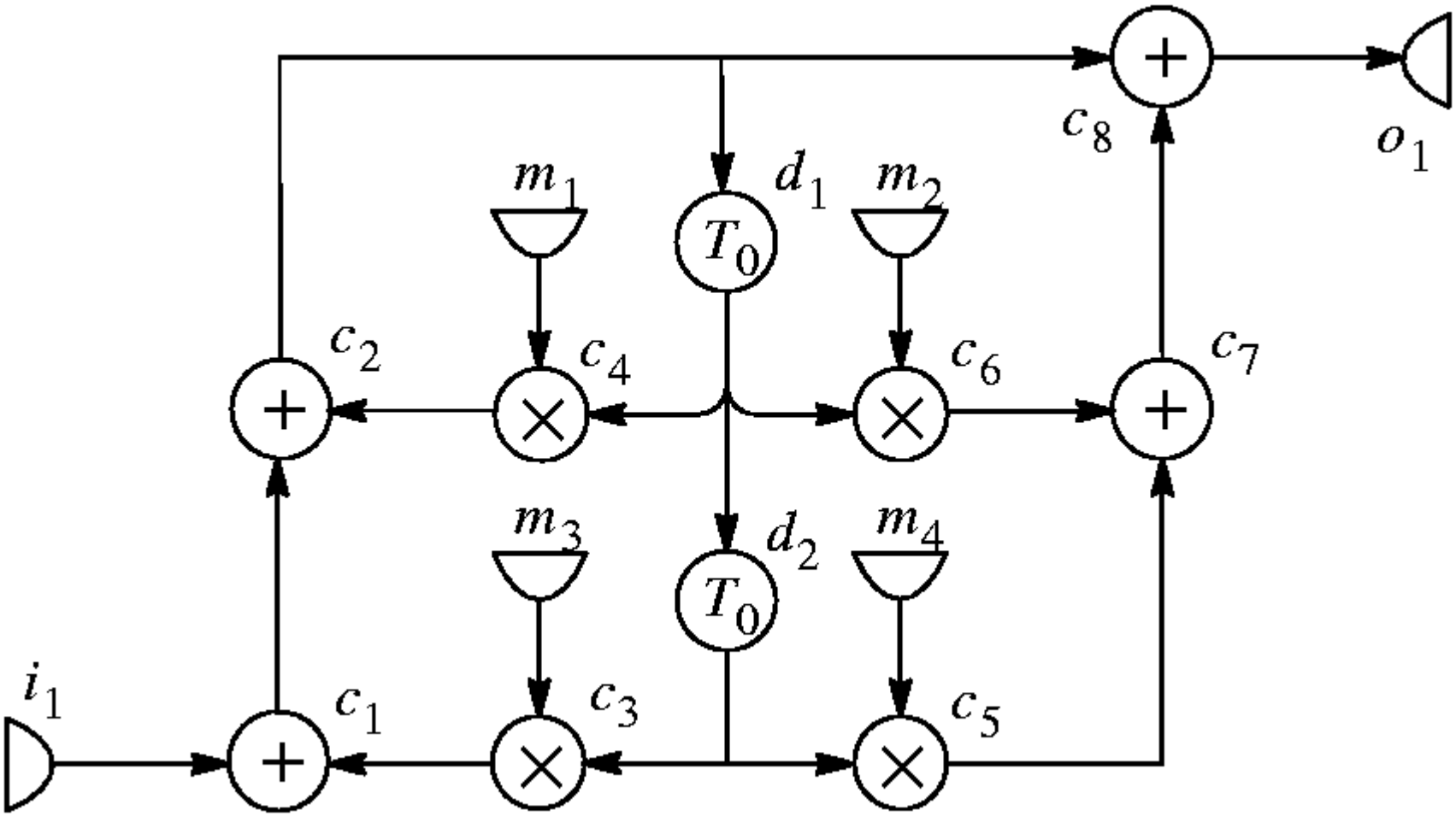
Version 2, September 2000

Jaap Hofstede

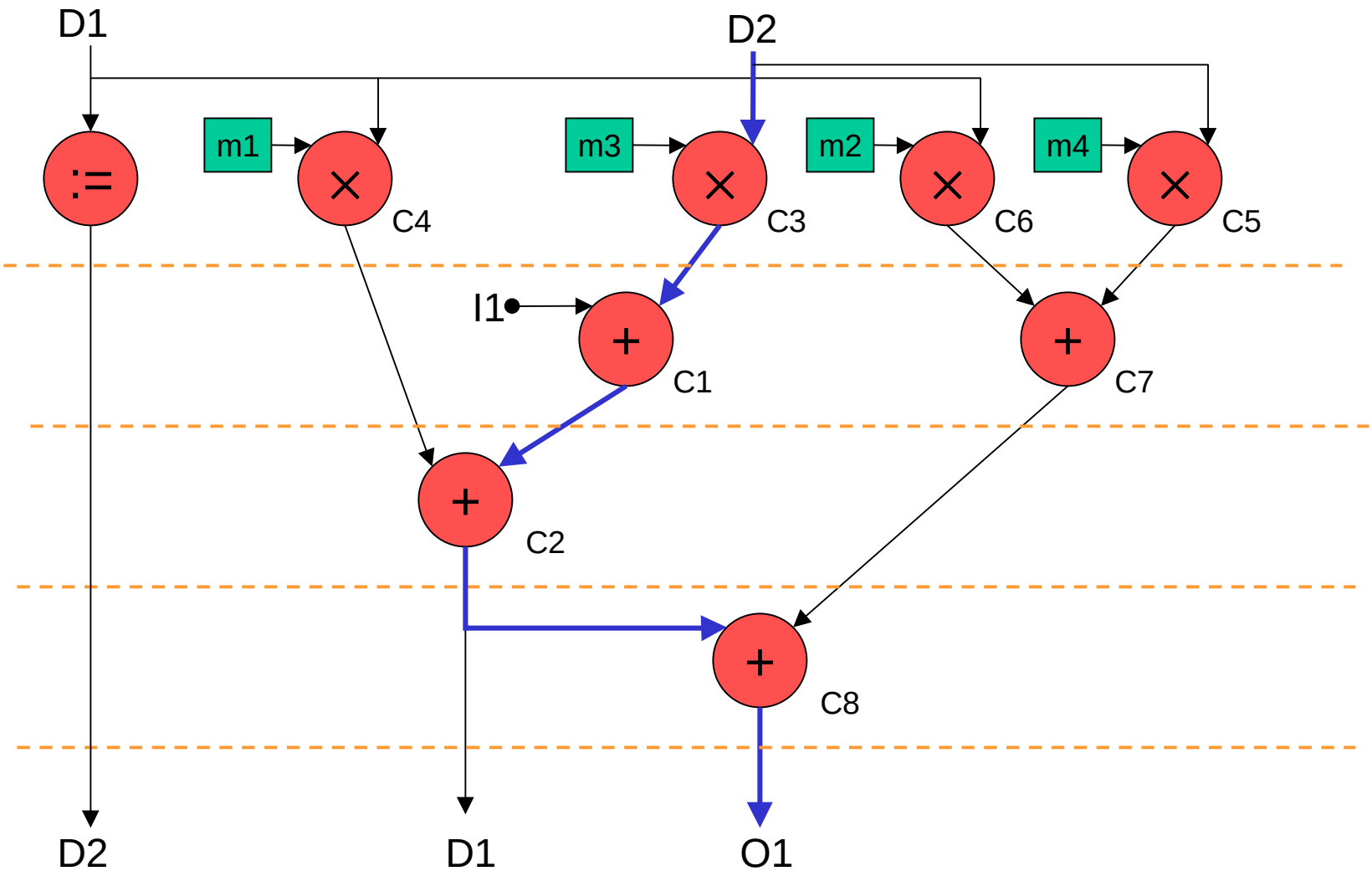
```
#define m1 …
#define m2 …
#define m3 …
#define m4 …

main()
{ float t, i1, o1, d1=0.0, d2=0.0;
  while (1) {
      in(i1);
      t = i1 + m3*d2 + m1*d1;
      o1 = t + m4*d2 + m2*d1;
      d2 = d1; d1 = t;
      out(o1);
  }
}
```

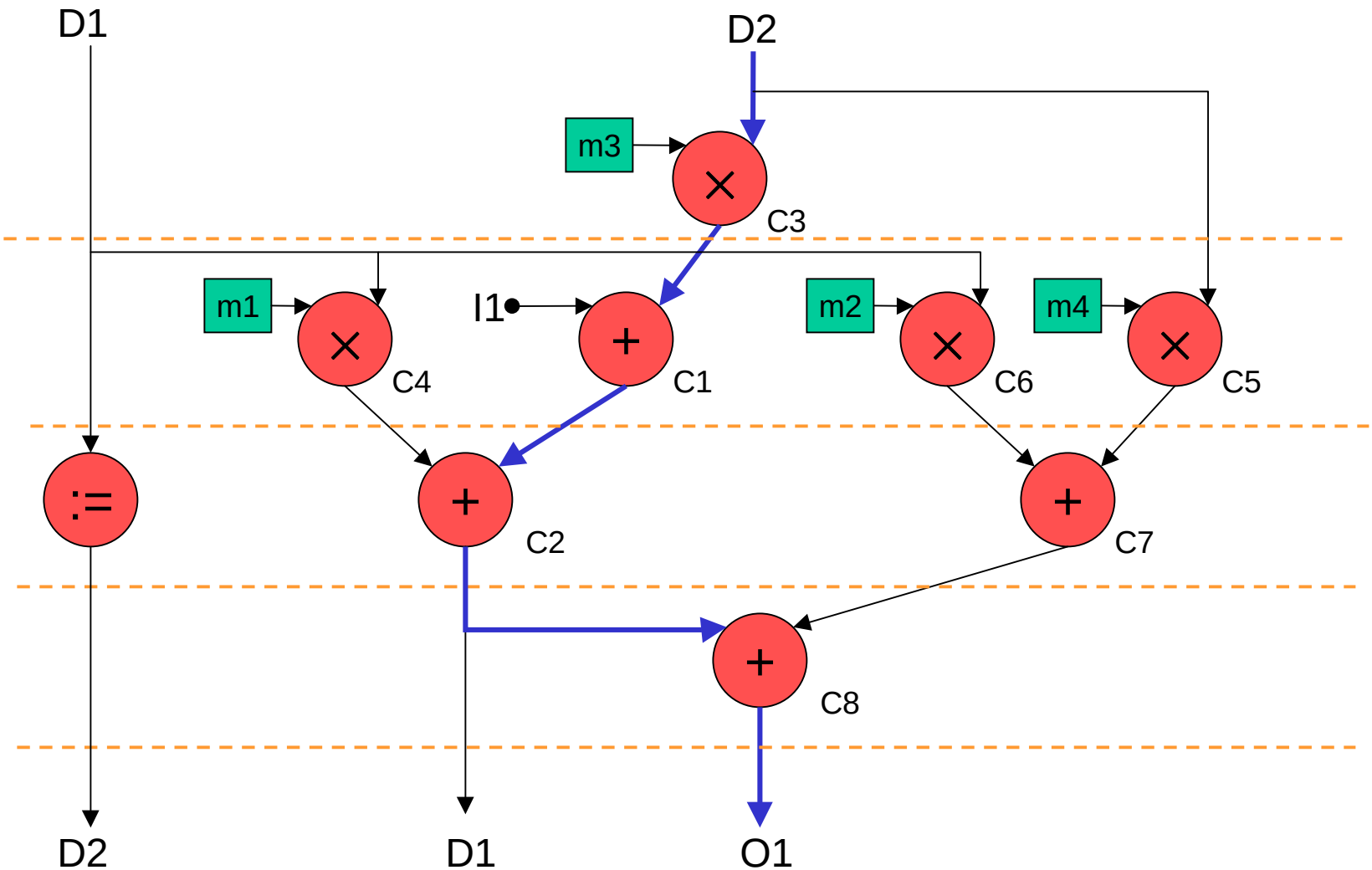
Specification in C



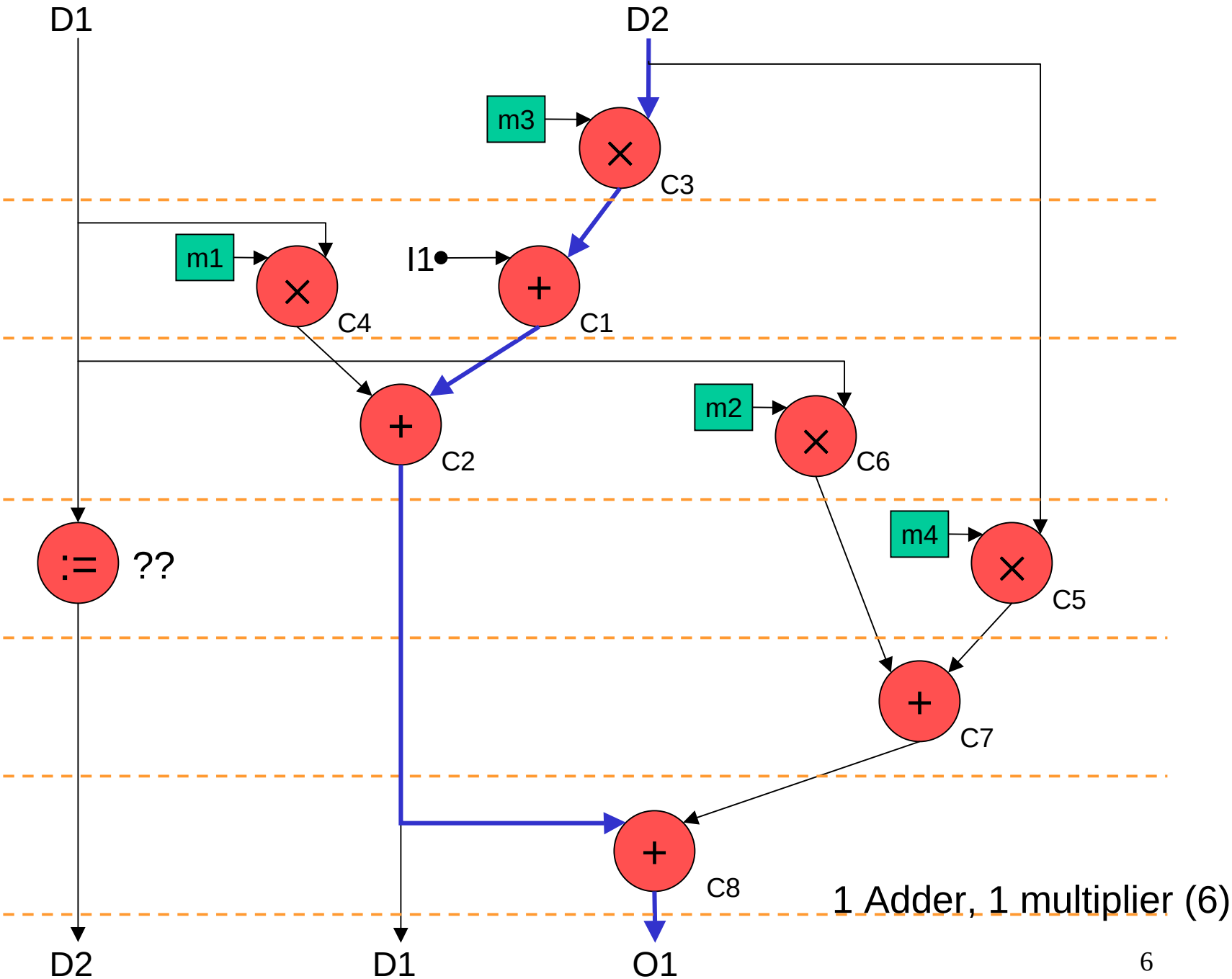
Original dataflow graph



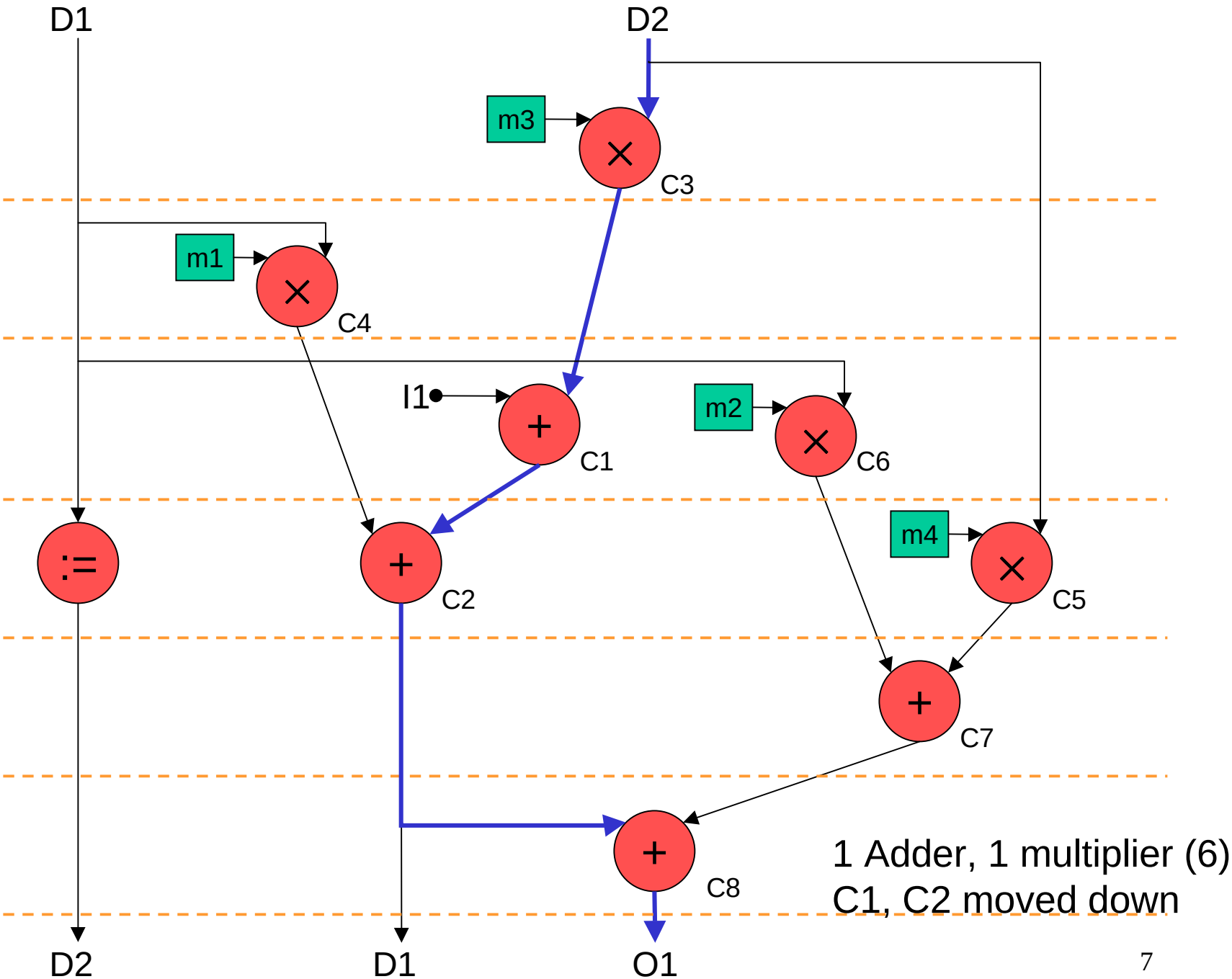
ASAP (4)

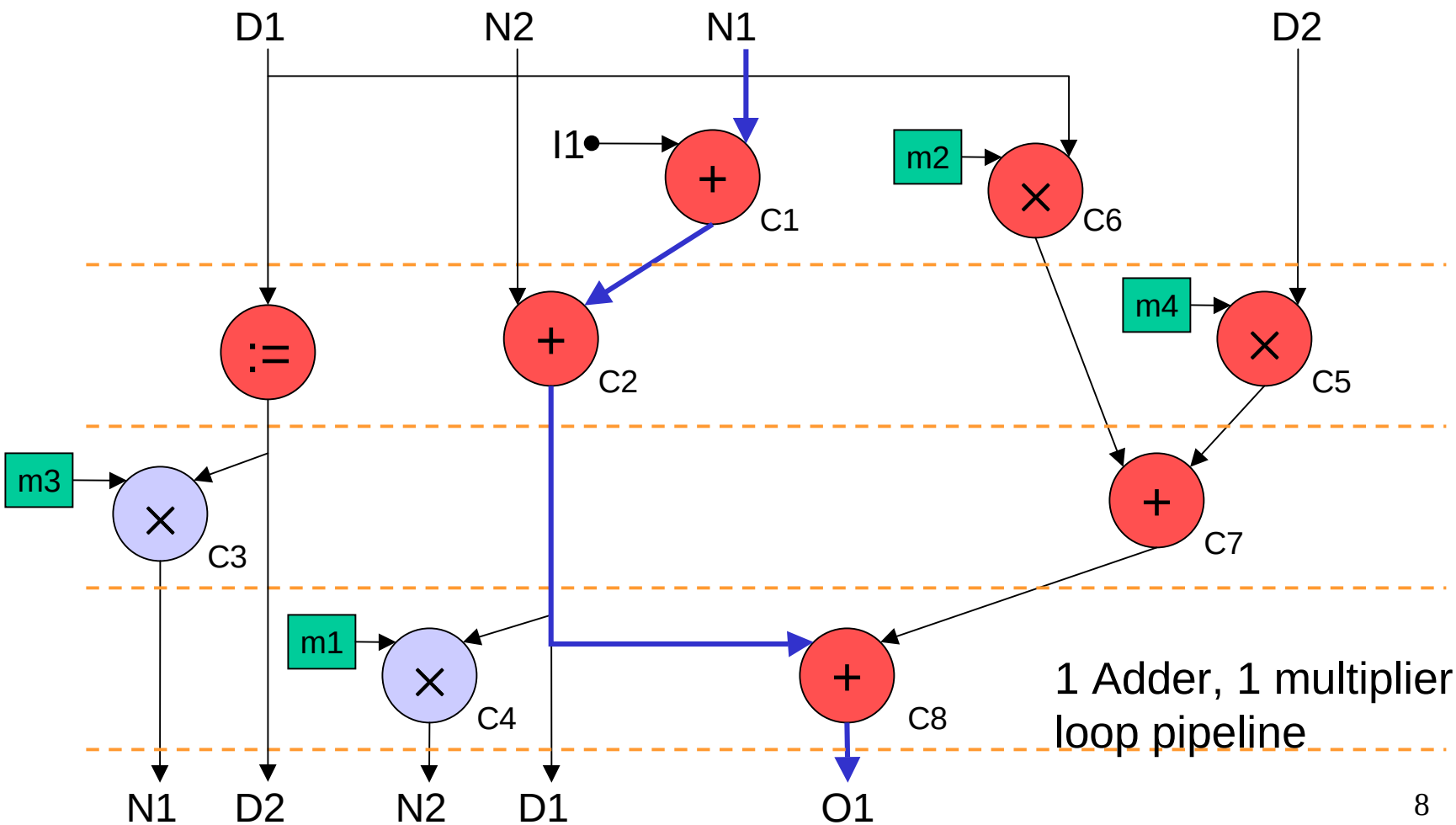


ALAP (4)

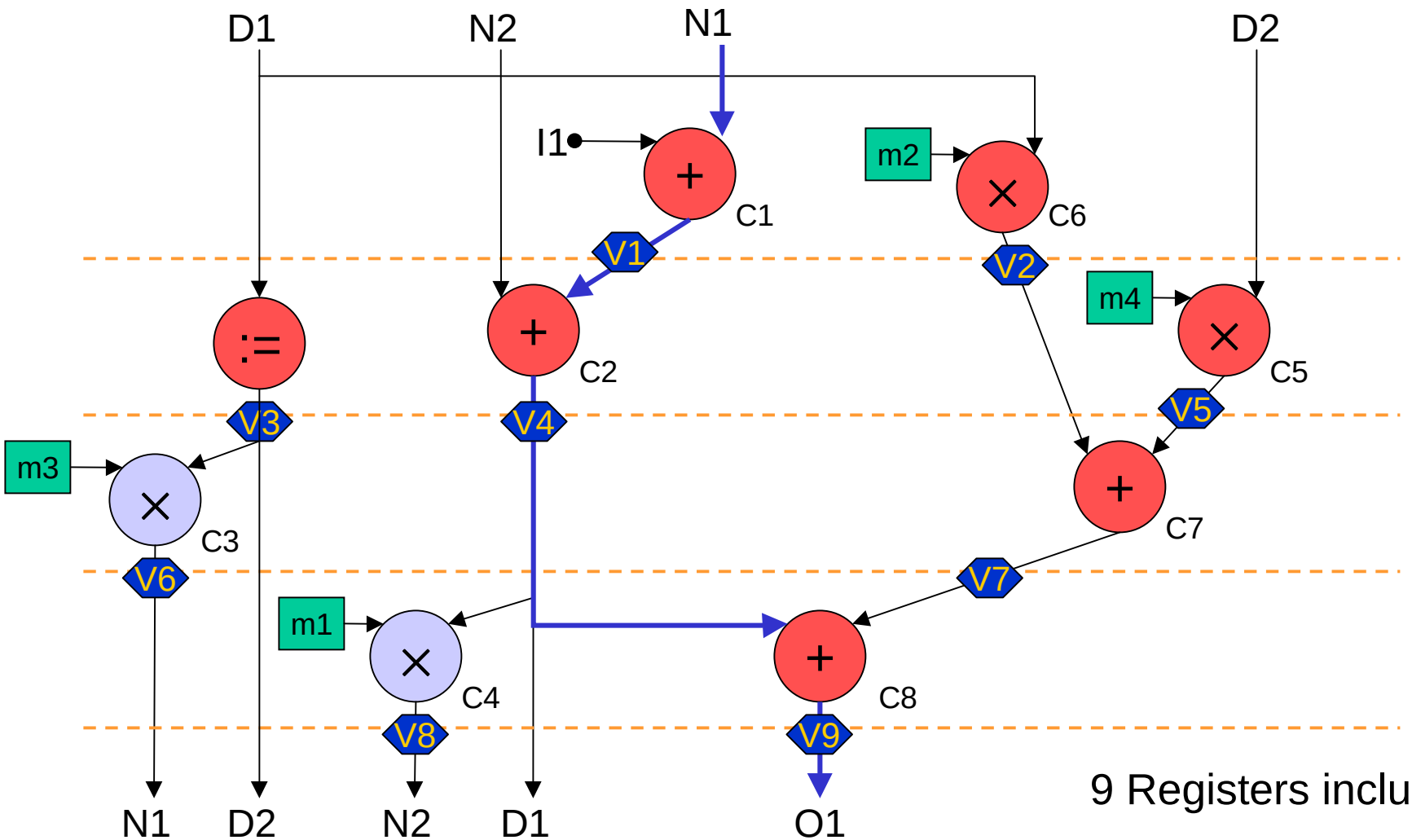


1 Adder, 1 multiplier (6)

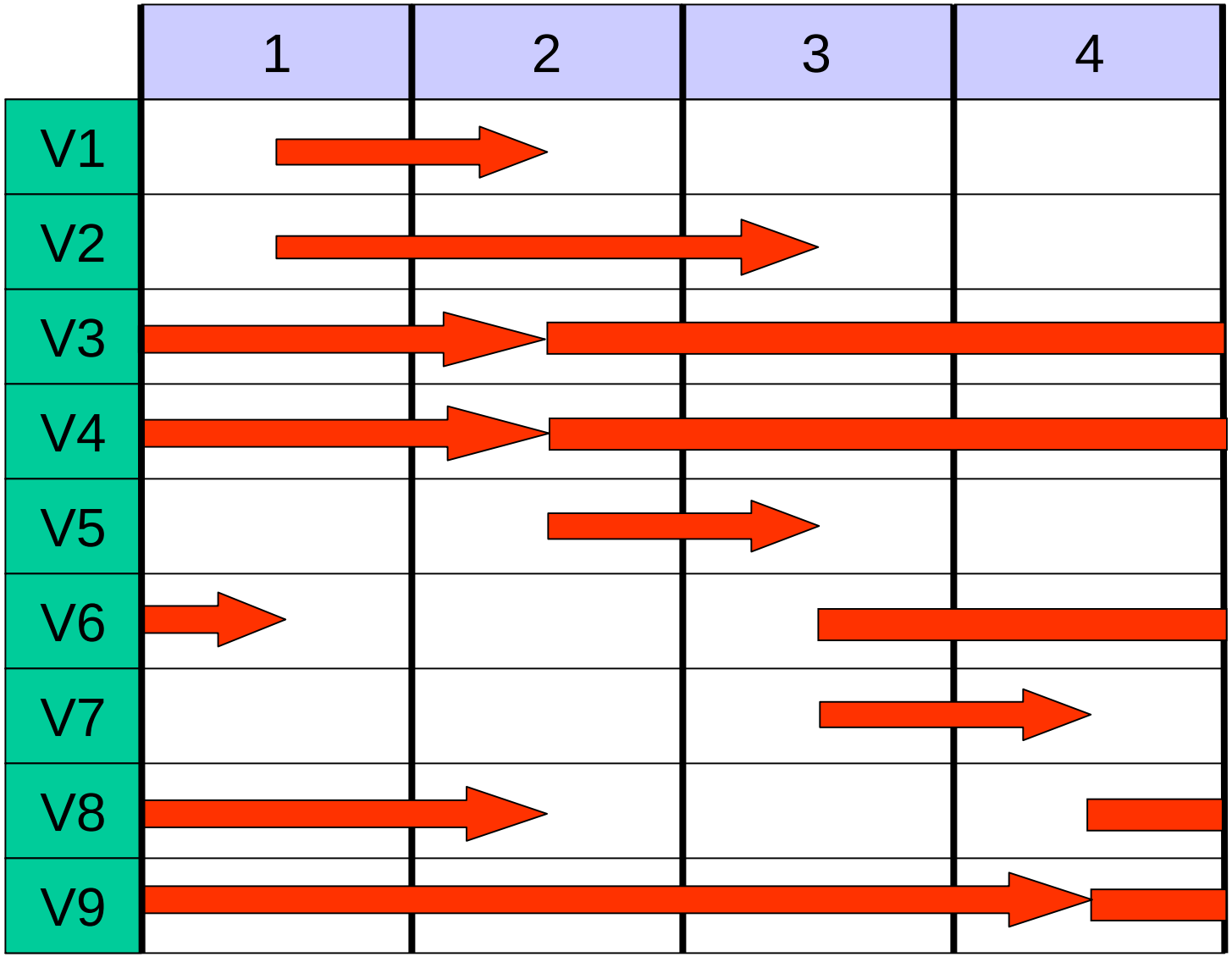

D1
D2
m3
C3
m1
C4
I1
C1
m2
C6
:=
C2
m4
C5
C7
C8
D2
D1
O1
1 Adder, 1 multiplier (6)
C1, C2 moved down



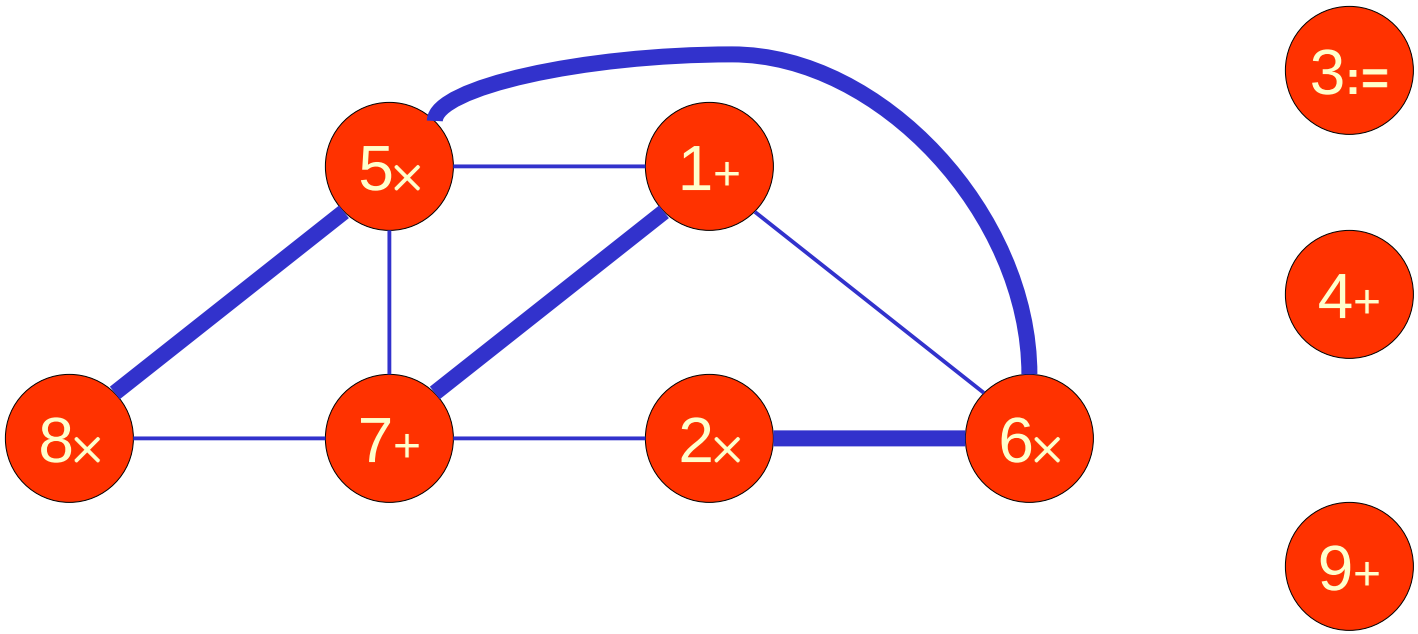
1 Adder, 1 multiplier (4) loop pipeline



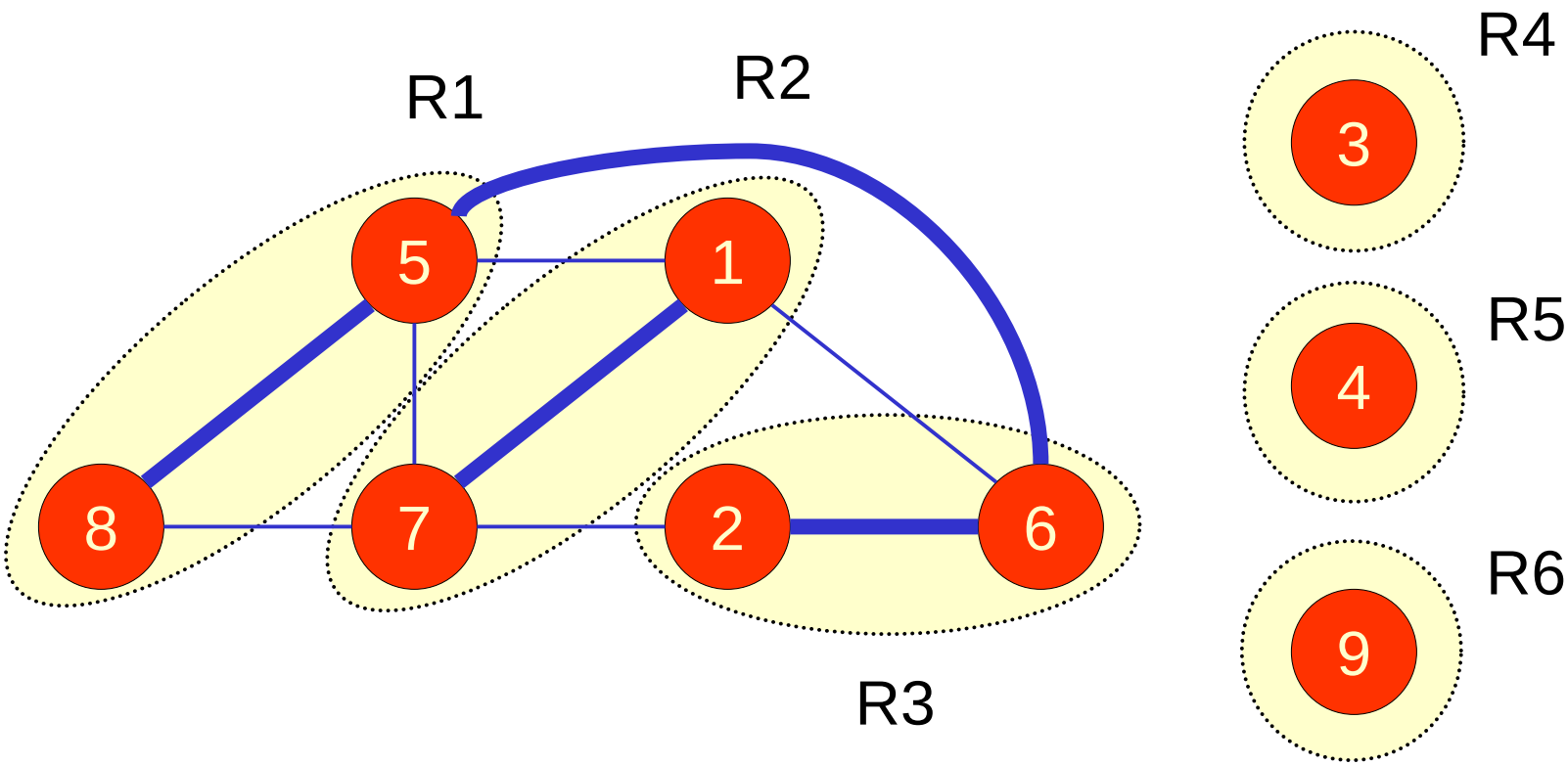
9 Registers included



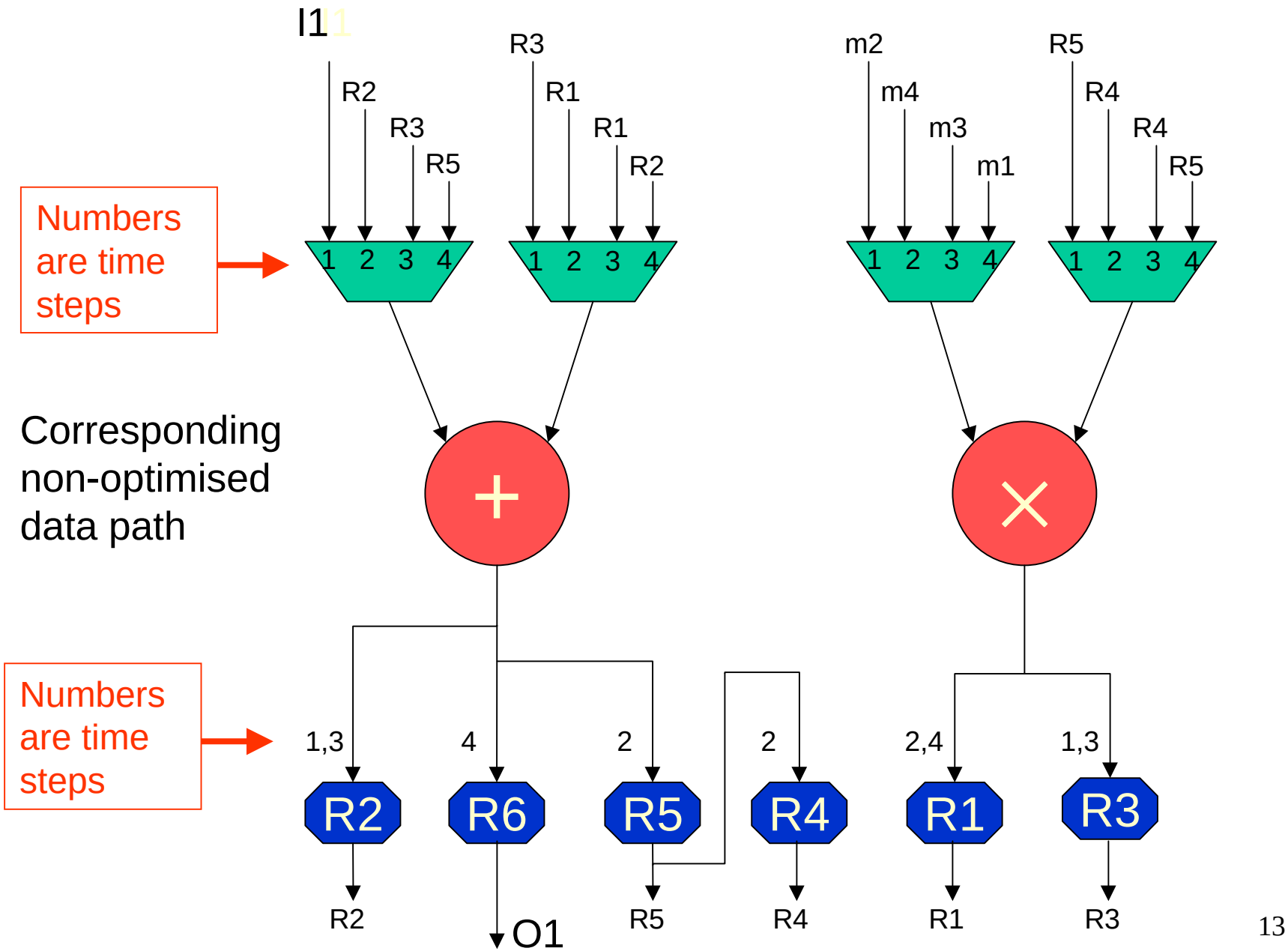
Lifetimes of registers



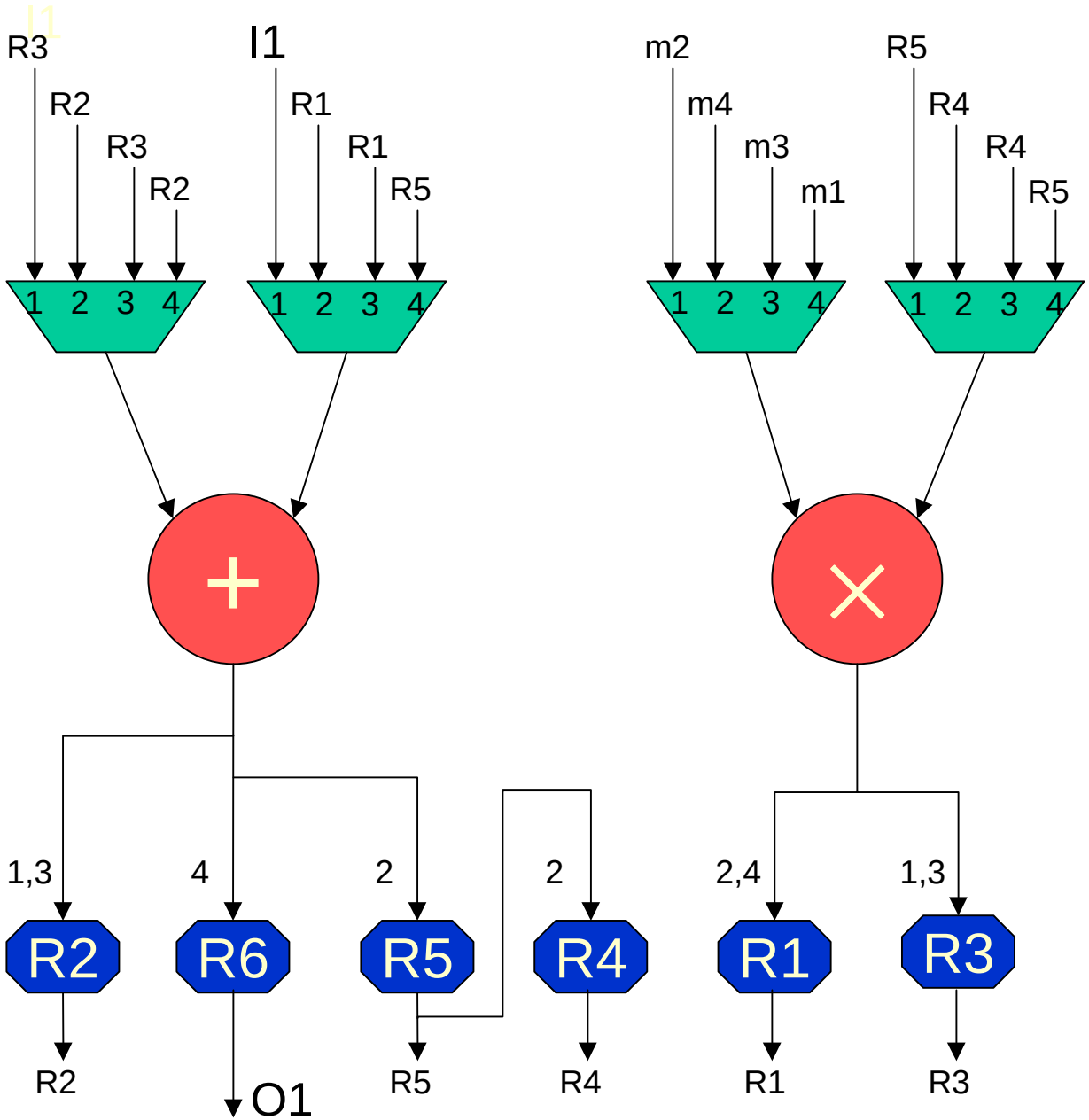
Lifetime compatibility graph
(+ × := is source of data)



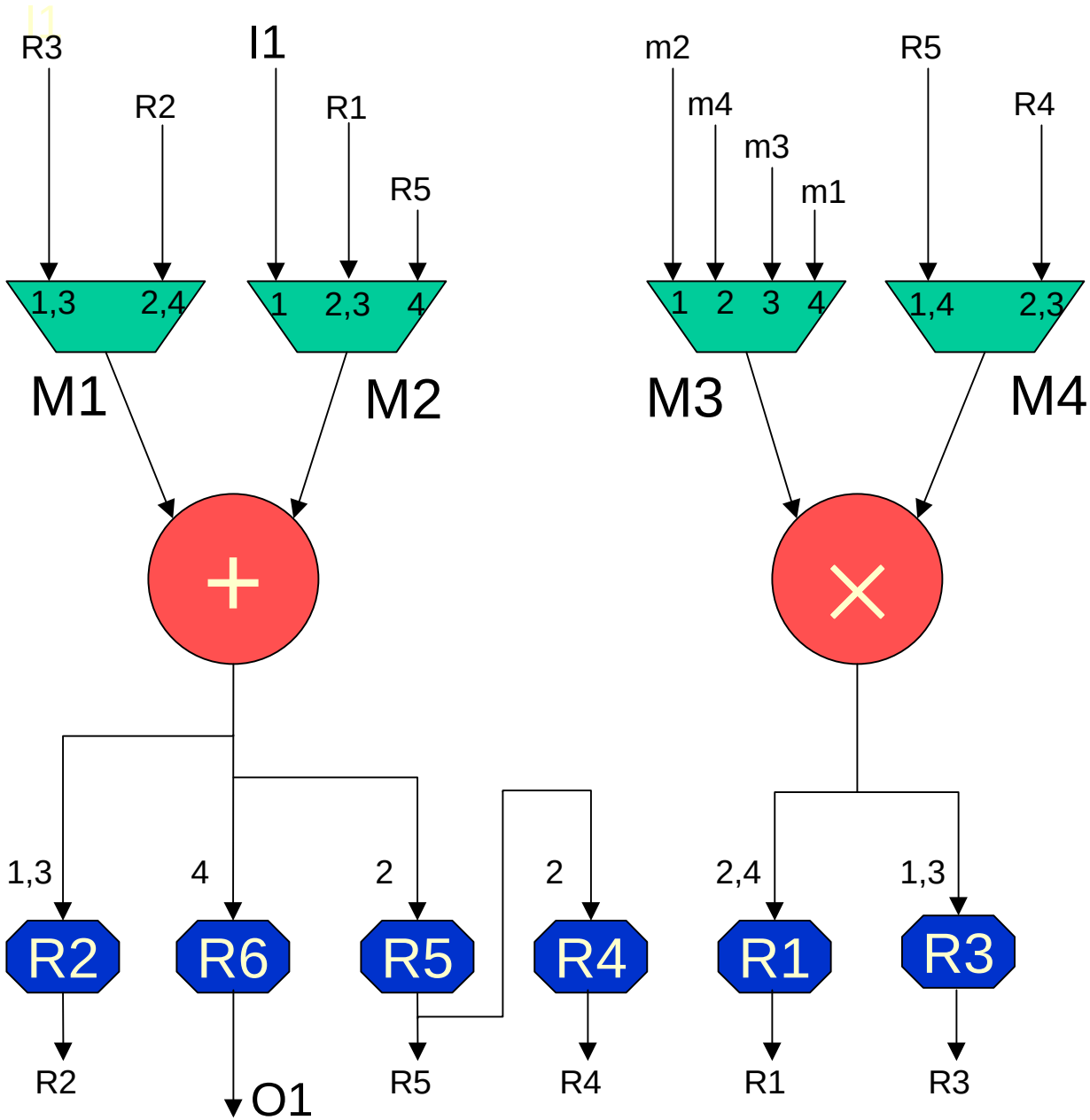
Clique partitioning


I1
R2
R3
R5
R3
R1
R1
R2
m2
m4
m3
m1
R5
R4
R4
R5
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
Numbers are time steps
Corresponding non-optimised data path
+
×
Numbers are time steps
1,3
4
2
2
2,4
1,3
R2
R6
R5
R4
R1
R3
R2
O1
R5
R4
R1
R3

Multiplexers optimised by commutative property



Size of multiplexers reduced



Controller-1

- All six registers have an enable input, *ena*
  R*x*.ena
- M1 and M4 have 1 control input, *s:*
  M*x*.s
- M2 and M3 have 2 control inputs, *s1* and *s0*:
  M*x*.s1 and M*x*.s0
- Controller has four states:
  State1, State2, State3, State4

Controller-2

State	enable						M1	M2		M3		M4
	R1	R2	R3	R4	R5	R6	s	s1	s0	s1	s0	s
1	0	1	1	0	0	0	0	0	0	0	0	0
2	1	0	0	1	1	0	1	0	1	0	1	1
3	0	1	1	0	0	0	0	0	1	1	0	1
4	1	0	0	0	0	1	1	1	0	1	1	0

R1.ena = M1.s = M3.s0 = State2 + State4

R2.ena = R3.ena = State1 + State3

R4.ena = R5.ena = State2

R6.ena = M2.s1 = State4

M2.s0 = M4.s = State2 + State3

M3.s1 = State3 + State4