FÖRELÄSNING 13

Sekvenskretsar - vippor och latchar

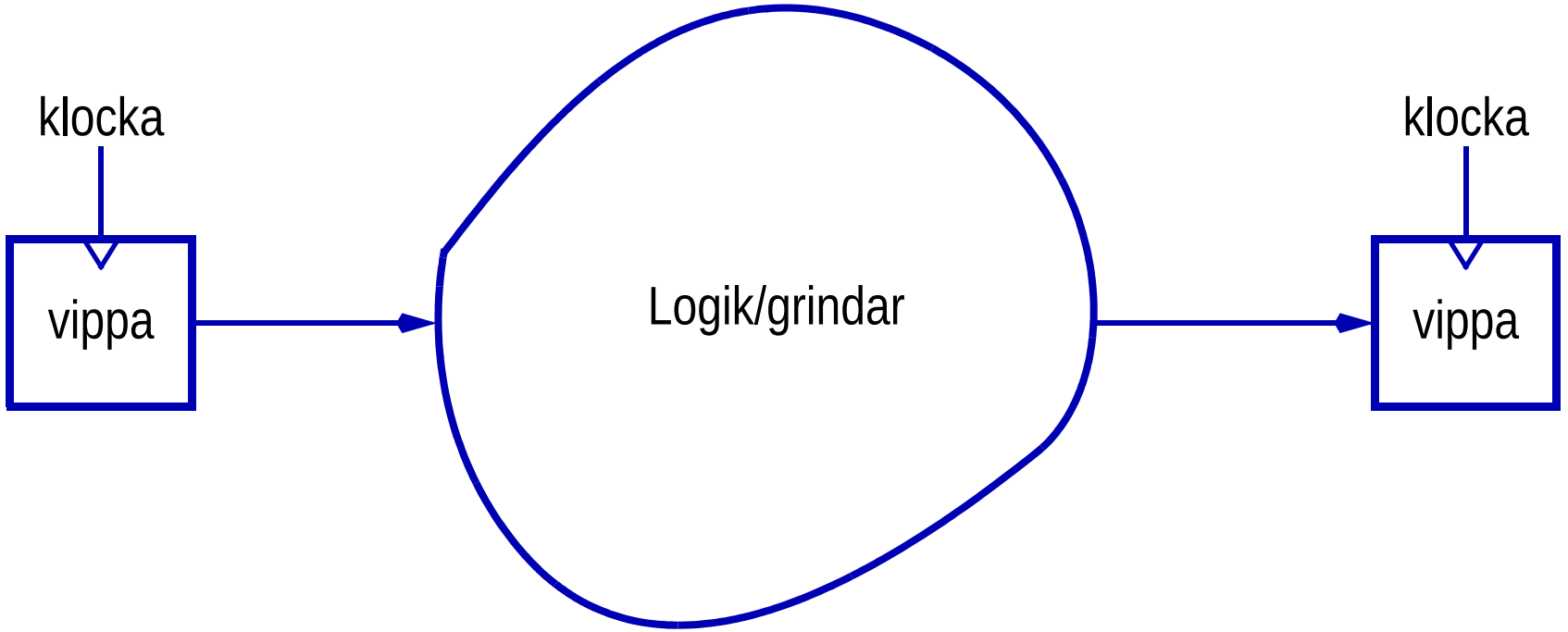
Digitalt konstruktionsexempel

Sekvenskretsar - vippor och latchar

(S&S4 13.7/S&S5 11.1)

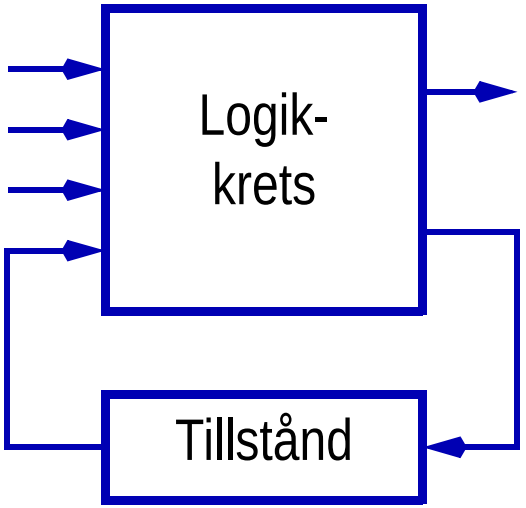
Logik och Vippor

- Vi har maximalt en klockperiod på oss att utvärdera CMOS-logiken som finns mellan vippor/register.



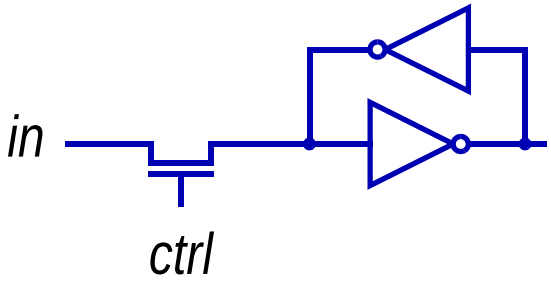
KOMBINATORISKA OCH SEKVENTIELLA KRETSAR

Kombinatorisk



Sekventiell - FSM

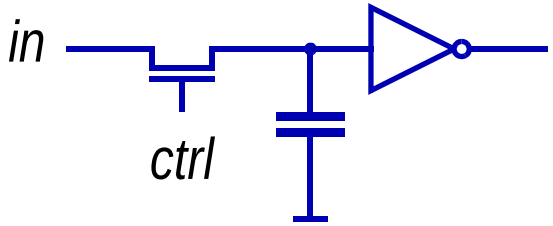
Statiska och Dynamiska Kretsar



En statisk krets

-

noden med det logiska värdet (efter NMOS:en) ansluts till V_{DD} eller jord



En dynamisk krets

-

noden med det logiska värdet (efter NMOS:en) håller sin spänning dynamiskt när *ctrl* = 0

TIMINGKRETSAR

- För det mesta kretsar analog teknik kring kontinuerlig signalamplitud och kontinuerlig tid, medan digital teknik använder sig av en begreppsvärld där signalamplitud och tid diskretiseras till 0 / 1 respektive klockcykler.
- De digitala kretsar vi hittills tittat på är förstärkande element som slår från jord till matningsspänning, och vi har alltså lärt känna hur digitala kretsar beter sig m.a.p. de digitala signalernas amplitud.
- Nu är det dags att studera de digitala kretsar som sköter synkroniseringen av signaler till de digitala tidsluckor som vi får använda: Dessa kallas timingkretsar för de tar hand om den klockstyrda tidspassningen (“timingen”) av de logiska signalerna.

Latchar och Vippor

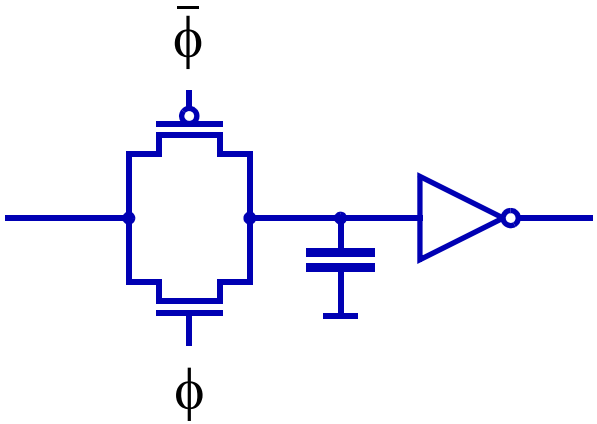
En latch ...

- är transparent under en halv klockperiod, och lagrar data på utgången under den andra halvan.
- fördröjer en signal en halv klockperiod.

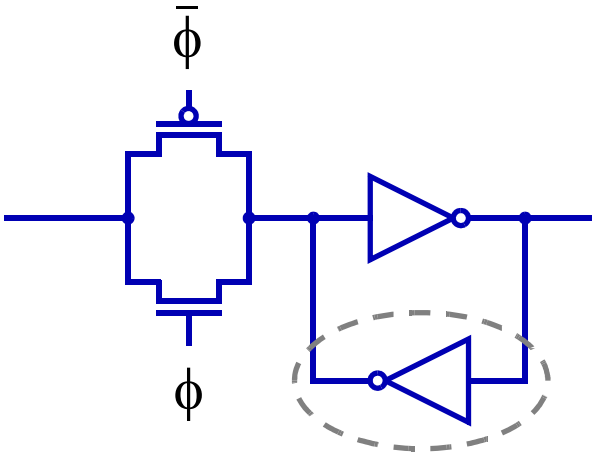
En D-vippa ...

- innehåller två olika latchar (en Mästare följd av en Slav).
- fördröjer en signal en klockperiod.

Dynamiska och Statiska Latchar

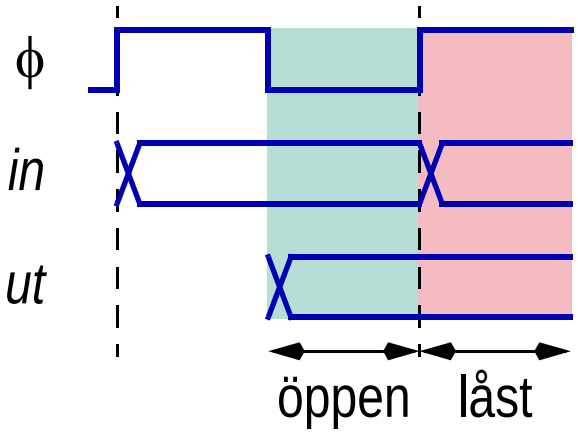


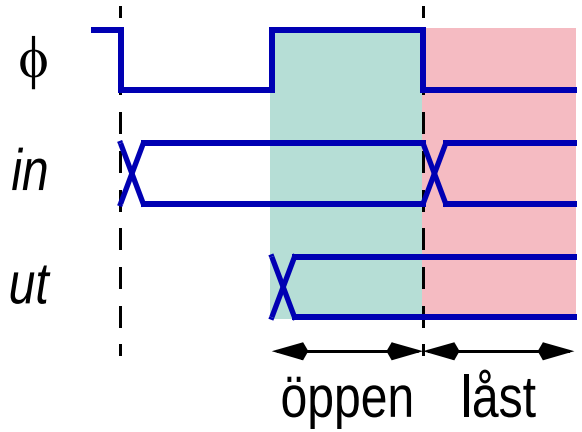
Dynamisk latch

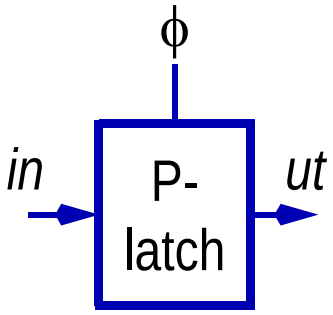


Statisk latch

LATCHAR OCH TIMING



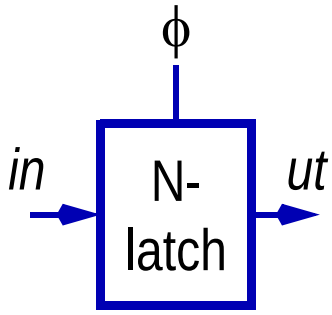




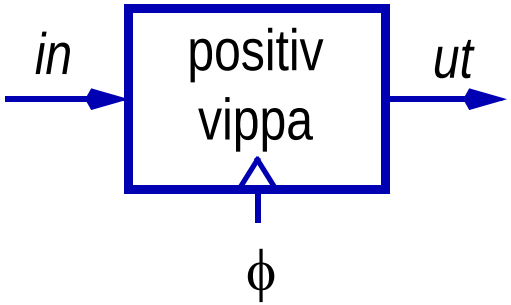
öppen $\Rightarrow$ *ut* = *in*

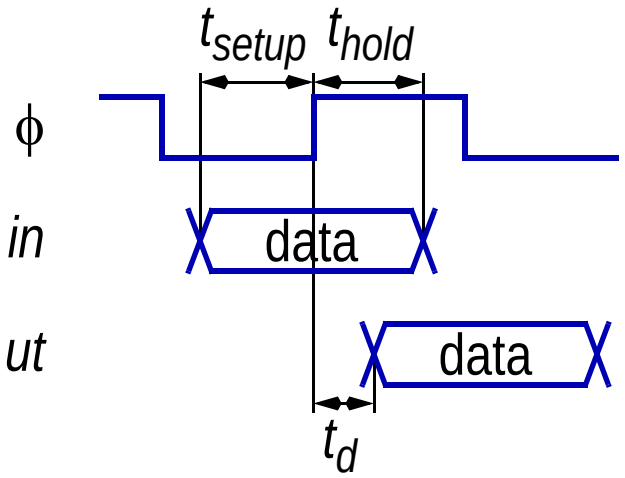
låst $\Rightarrow$ *ut* från öppna

tillståndet låses kvar



En Flanktriggad Vippas Egenskaper

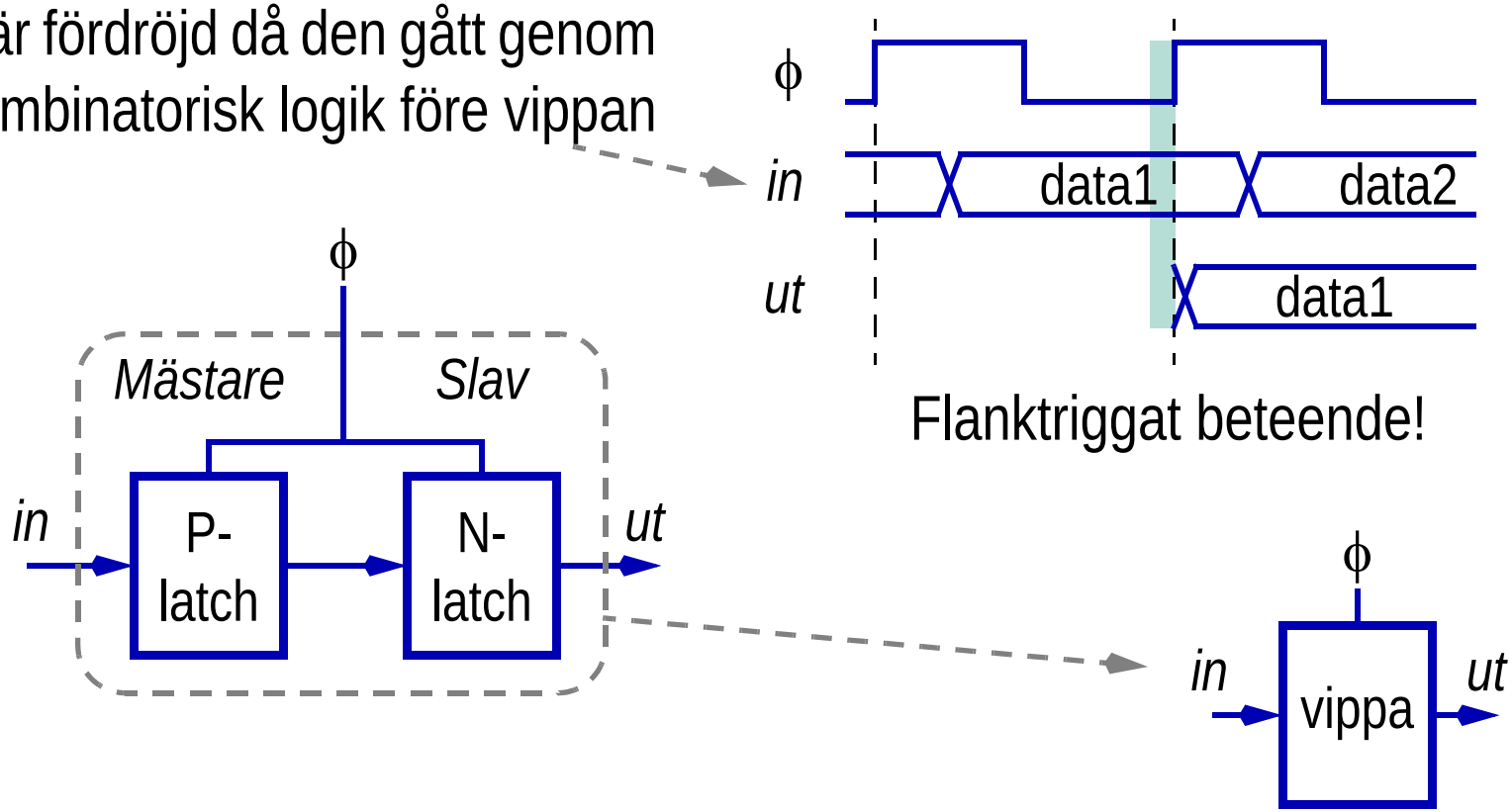




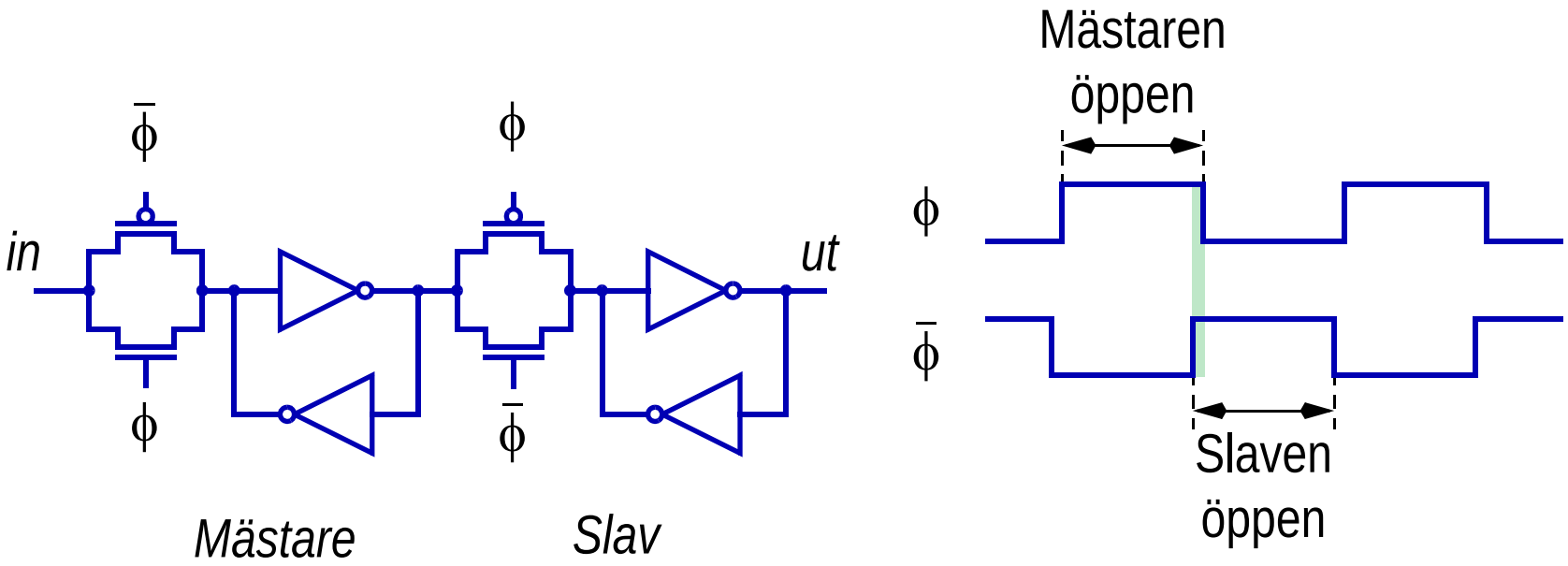
- Setuptiden/*holdtiden* är den minsta tidrymd före/*efter* klockflanken som data måste finnas tillgängligt vid vippans ingång för att korrekt funktion ska kunna garanteras.
- Klockperioden måste väljas så att $T > t_{setup} + t_d(\text{vippa}) + t_d(\text{logik})$.

En Master-Slave Vippa och Dess Timing

in är fördröjd då den gått genom kombinatorisk logik före vippan

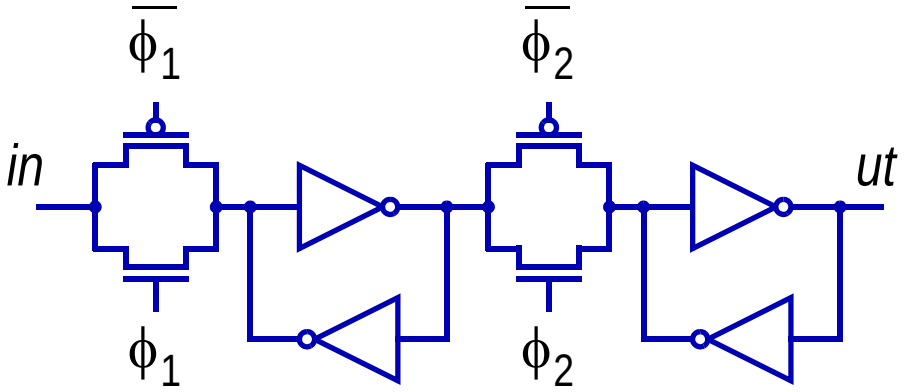


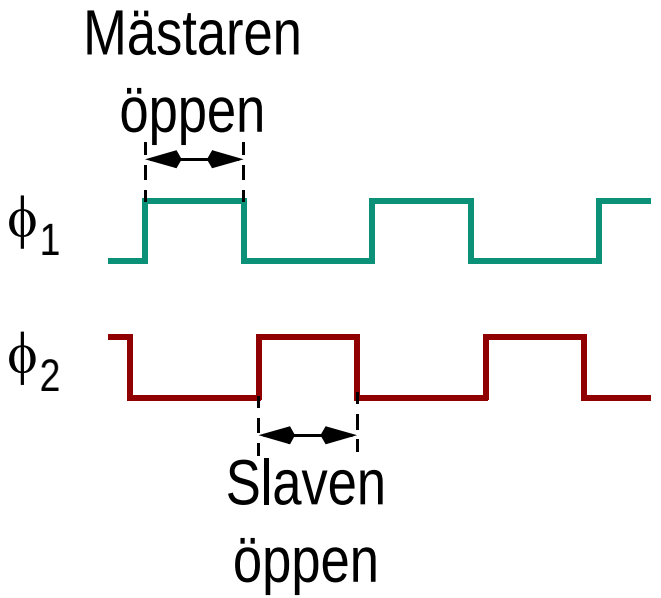
En Krets för en Master-Slave Vippa



- Om klockinvers $\overline{\phi}$ ligger före ϕ så kan vippan sluta att fungera, eftersom båda latcharna kommer vara öppna under kort tid:
  Vi får ett race (datarusning).

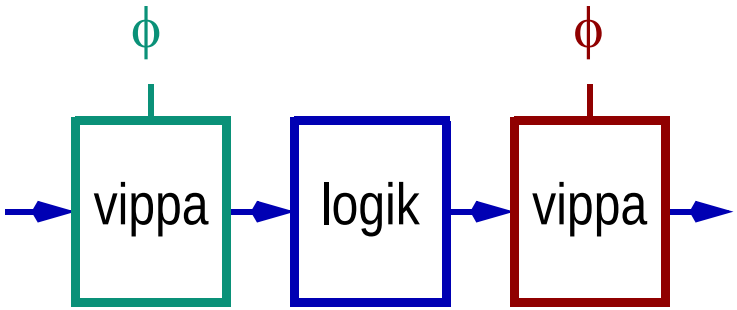
TVÅFASKLOCKNING

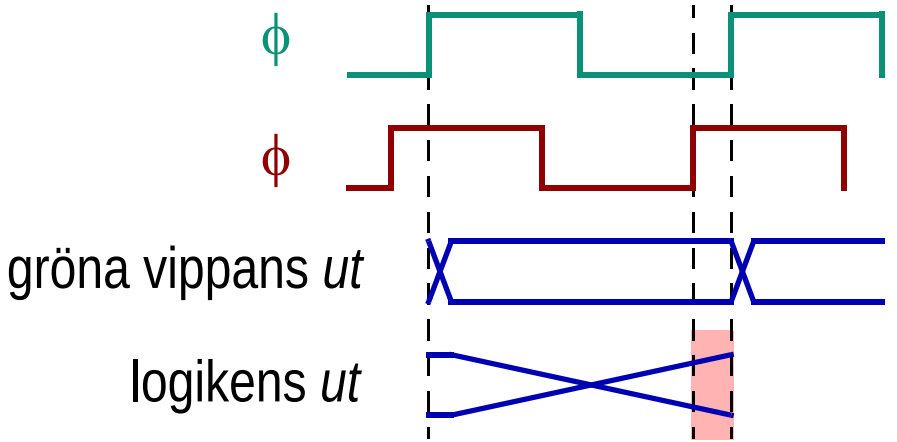




- Med icke-överlappande tvåfasklockning kan man reglera så att race inte inträffar.
- Man har normalt sett inte råd med denna konservativa klockstrategi, för den kastar bort vår dyrbara tid.

KLOCKSKEV OCH TIMINGBUDGET





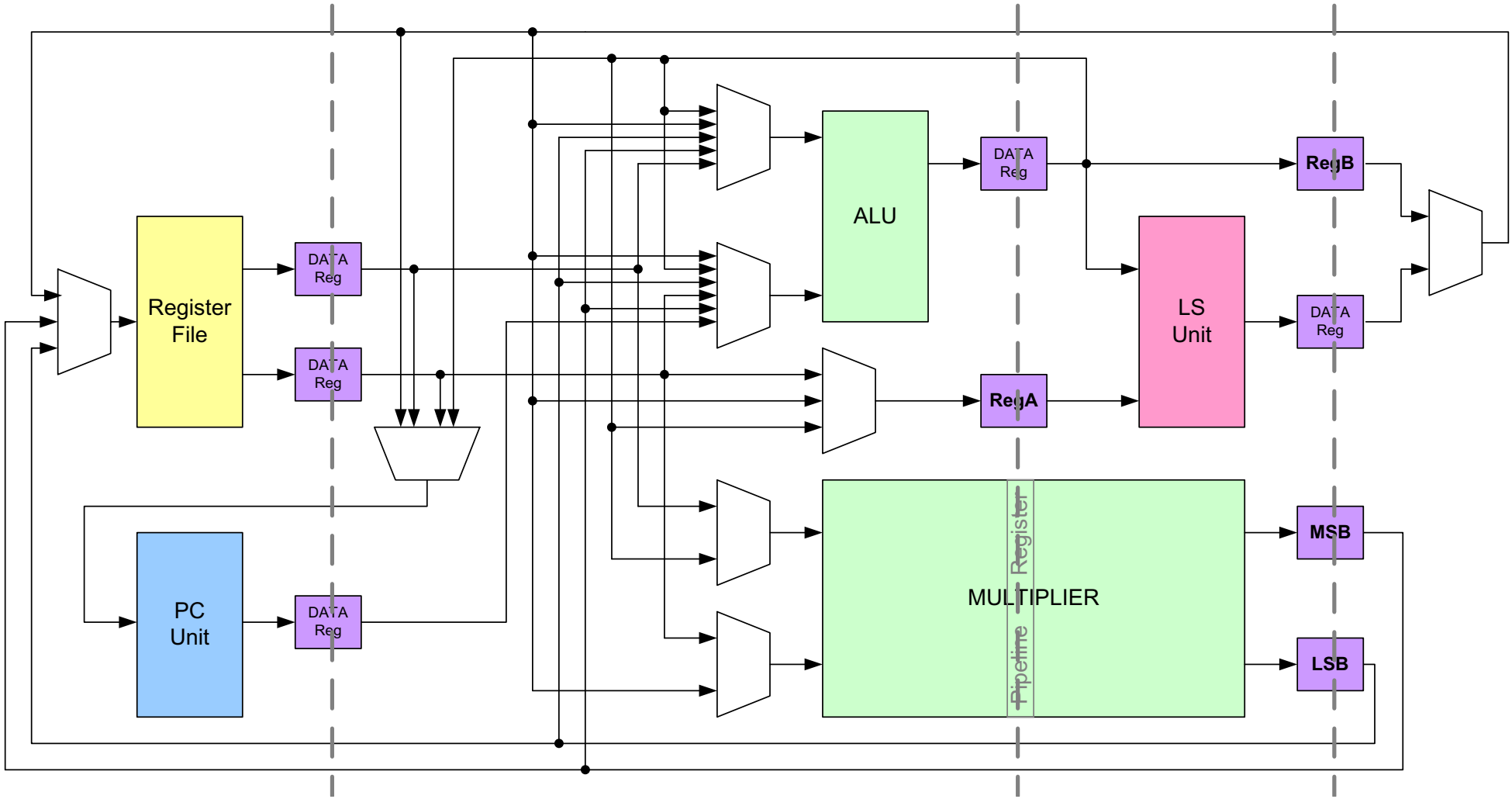
- Marginaler på logikens fördröjning krävs om det finns risk för klockskev.
- Logiken får alltså inte utnyttja hela periodtiden *T* p.g.a. klockskev.

Digitalt konstruktionsexempel

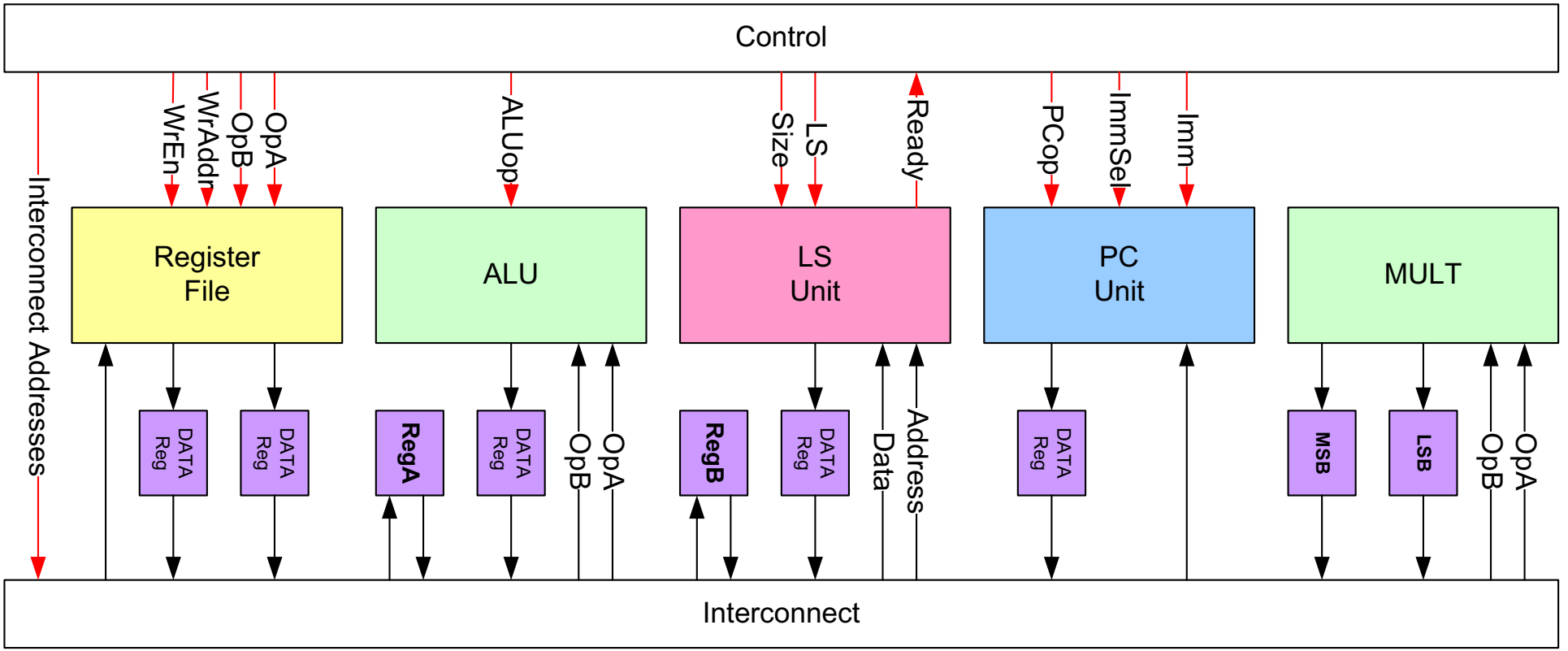
KONSTRUKTIONSFLÖDE

- Övergripande kravspecifikation: Till exempel en komplett processor.
- Nedbrytning i detaljer: Till exempel fokusering på en multiplikator, mellan registergränser.
- Algoritmval: Ska multiplikatorn använda Booth-omkodning?
- En första design baserad på tidigare erfarenhet (re-use) och tumregler (t.ex. tekniker från Kretselektronik).
- Utvärdering: Har vi konstruerat rätt funktionellt? Klarar vi kravspecens timing och power?
- Vid en lyckad utvärdering är man klar, annars omkonstruktion.

En MIPS-liknande processors dataväg



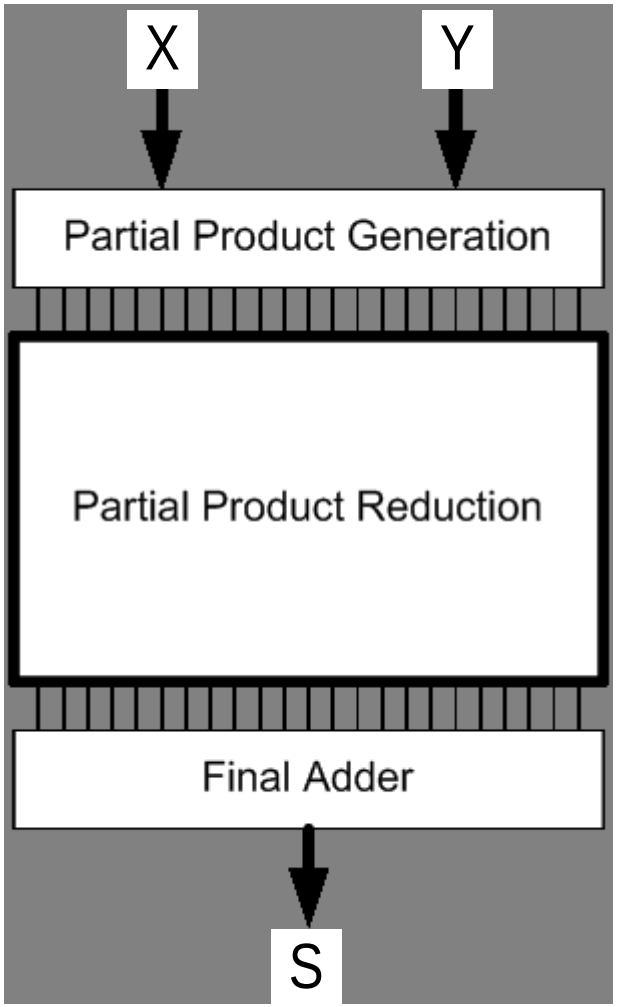
En FlexCore-Processors Dataväg



NEDBRYTNING: MULTIPLIKATION

								y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
								x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
								p_{70}	p_{60}	p_{50}	p_{40}	p_{30}	p_{20}	p_{10}	p_{00}
							p_{71}	p_{61}	p_{51}	p_{41}	p_{31}	p_{21}	p_{11}	p_{01}	
						p_{72}	p_{62}	p_{52}	p_{42}	p_{32}	p_{22}	p_{12}	p_{02}		
					p_{73}	p_{63}	p_{53}	p_{43}	p_{33}	p_{23}	p_{13}	p_{03}			
				p_{74}	p_{64}	p_{54}	p_{44}	p_{34}	p_{24}	p_{14}	p_{04}				
			p_{75}	p_{65}	p_{55}	p_{45}	p_{35}	p_{25}	p_{15}	p_{05}					
		p_{76}	p_{66}	p_{56}	p_{46}	p_{36}	p_{26}	p_{16}	p_{06}						
	p_{77}	p_{67}	p_{57}	p_{47}	p_{37}	p_{27}	p_{17}	p_{07}							
s_{15}	s_{14}	s_{13}	s_{12}	s_{11}	s_{10}	s_9	s_8	s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0

Nedbrytning: Multiplikatorns Blockschema



Algoritmval för Multiplikation

Teckensättning:

- Modifierad Booth-omkodning, Baugh-Wooley, ...

Reduktion av partiella produktbitar:

- Carry-save, HPM-träd, Wallace-träd, Left-Right, Leapfrog, ...

Vad ska man välja?

Erfarenhet och/eller kunnande från digitalkurser behövs.

ALGORITMVAL: BOOTH-OMKODNING 1(2)

- Vi har $x = -x_0 2^0 + x_1 2^{-1} + x_2 2^{-2} + x_3 2^{-3}$. Analysera $x = 2x - x$.
- $2(-x_0 2^0 + x_1 2^{-1} + x_2 2^{-2} + x_3 2^{-3}) - (-x_0 2^0 + x_1 2^{-1} + x_2 2^{-2} + x_3 2^{-3})$, eller

  $-x_0 2^1 + x_1 2^0 + x_2 2^{-1} + x_3 2^{-2} + x_0 2^0 - x_1 2^{-1} - x_2 2^{-2} - x_3 2^{-3}$, eller

  $-x_0 2^1 + (x_1 - (-x_0)) 2^0 + (x_2 - x_1) 2^{-1} + (x_3 - x_2) 2^{-2} - x_3 2^{-3}$, eller

  $(x_2 + x_1 - 2x_0) 2^{-1} + (x_3 - 2x_2) 2^{-3}$.
- Bara två signifikansnivåer används till sist: 2^{-1} and 2^{-3}.

Algoritmval: Booth-Omkodning 2(2)

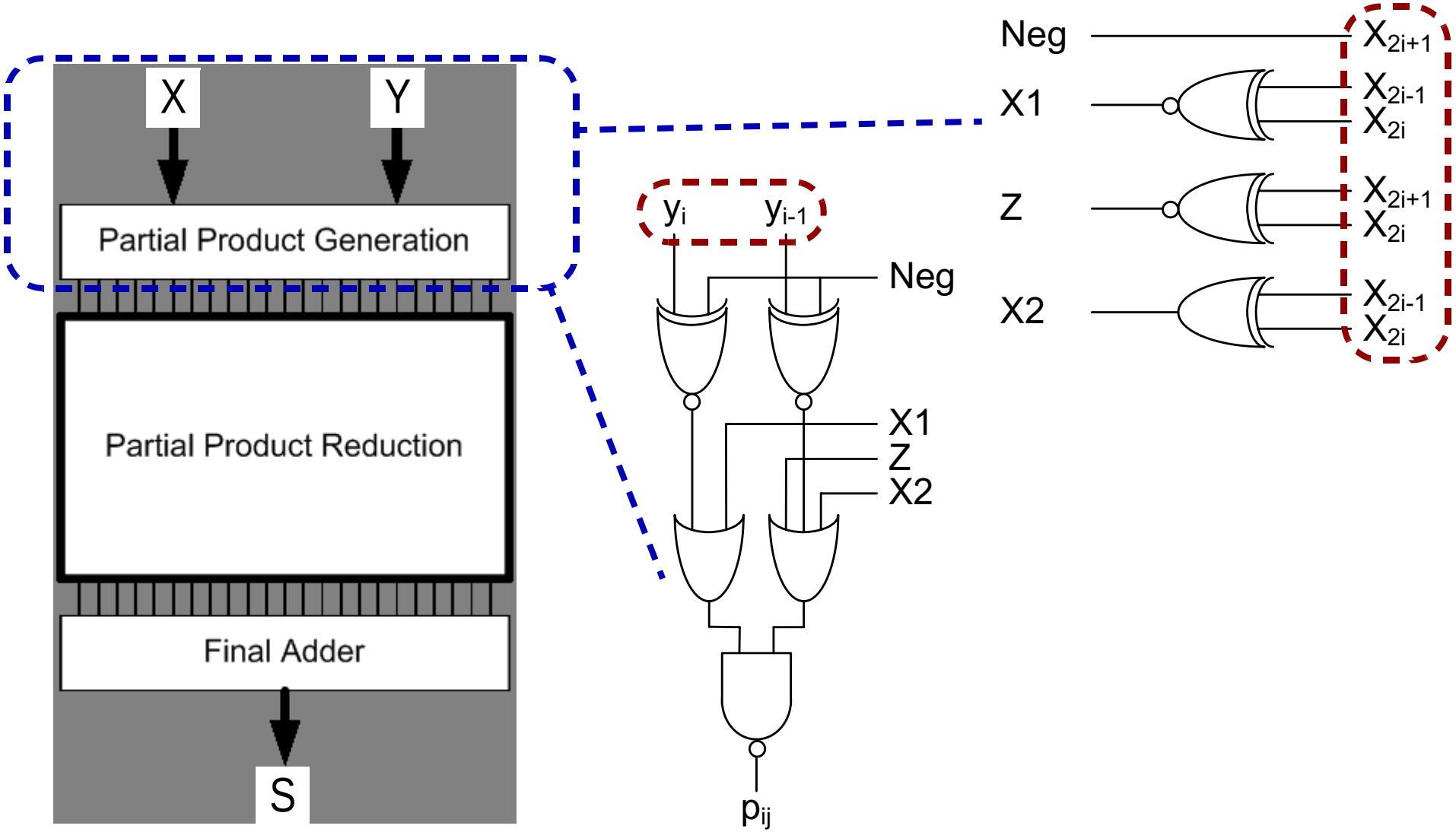
x_{i-2}	x_{i-1}	x_i	*Inc*
0	0	0	0
0	0	1	+1
0	1	0	+1
0	1	1	+2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

$s = x \cdot y$ där $y = 011001$, $x = 101110$

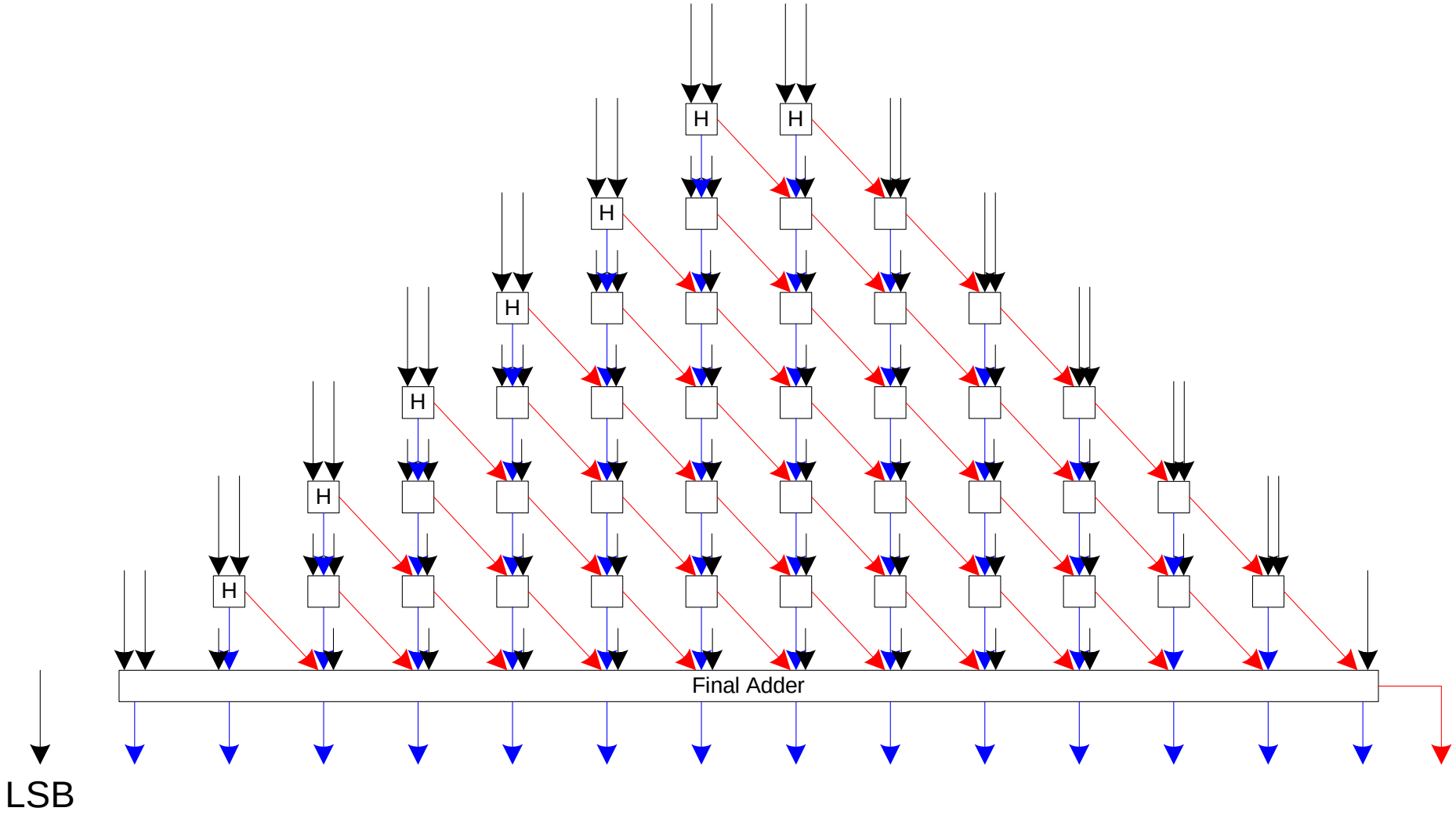
1. $x_4\, x_5\, (x_6) = 100 \Rightarrow$ Addera $(-2 \cdot y)2^0$
2. $x_2\, x_3\, x_4 = 111 \Rightarrow$ Addera $(0 \cdot y)2^2$
3. $x_0\, x_1\, x_2 = 101 \Rightarrow$ Addera $(-1 \cdot y)2^4$

```
      1 1 1 1 1 0 0 1 1 1 0
      0 0 0 0 0 0 0 0 0
+     1 1 0 0 1 1 1
-------------------------
      1 1 0 0 0 1 1 1 1 1 0
```

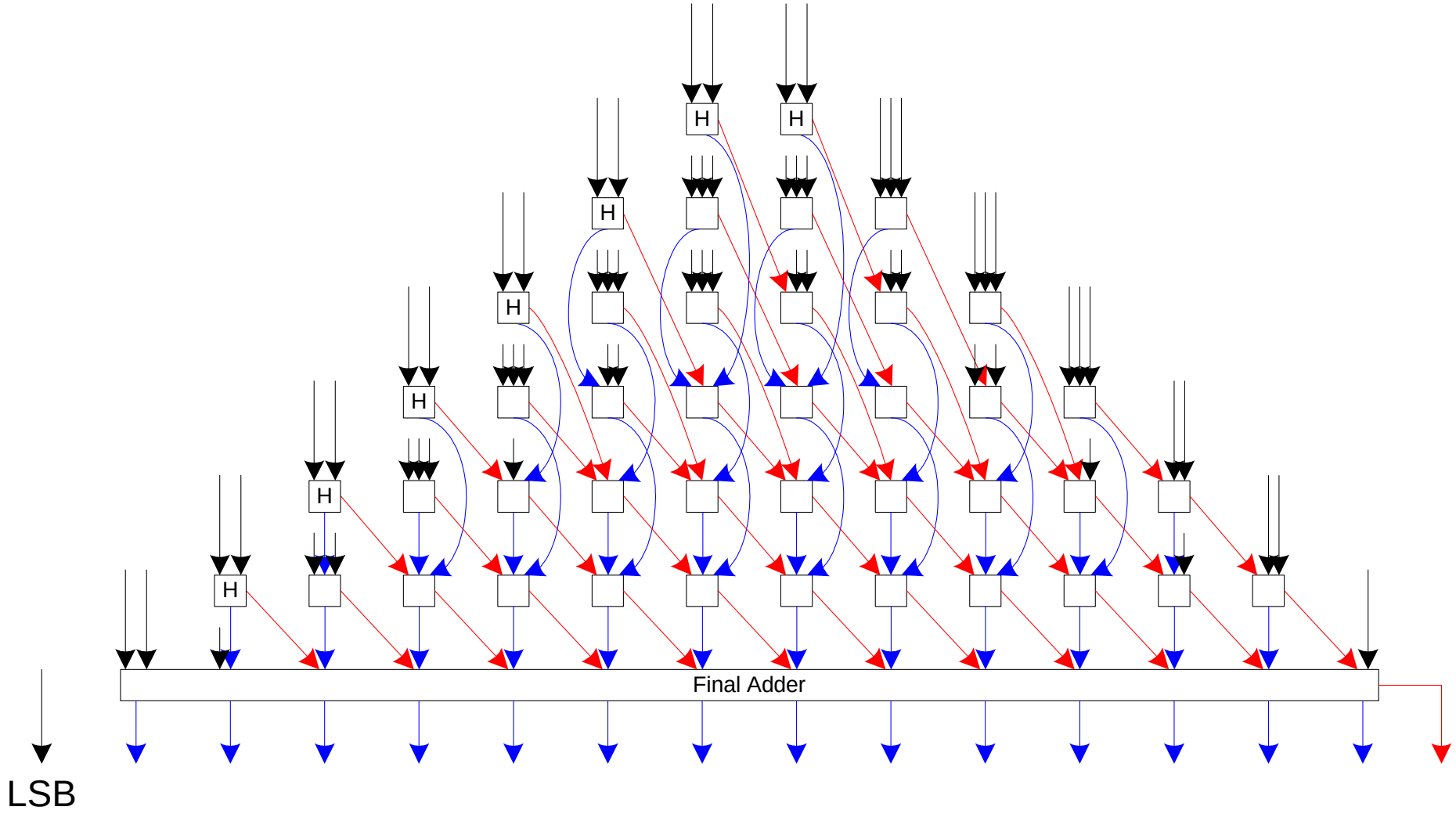
INITIAL DESIGN: BOOTH-OMKODNINGSKRETSAR



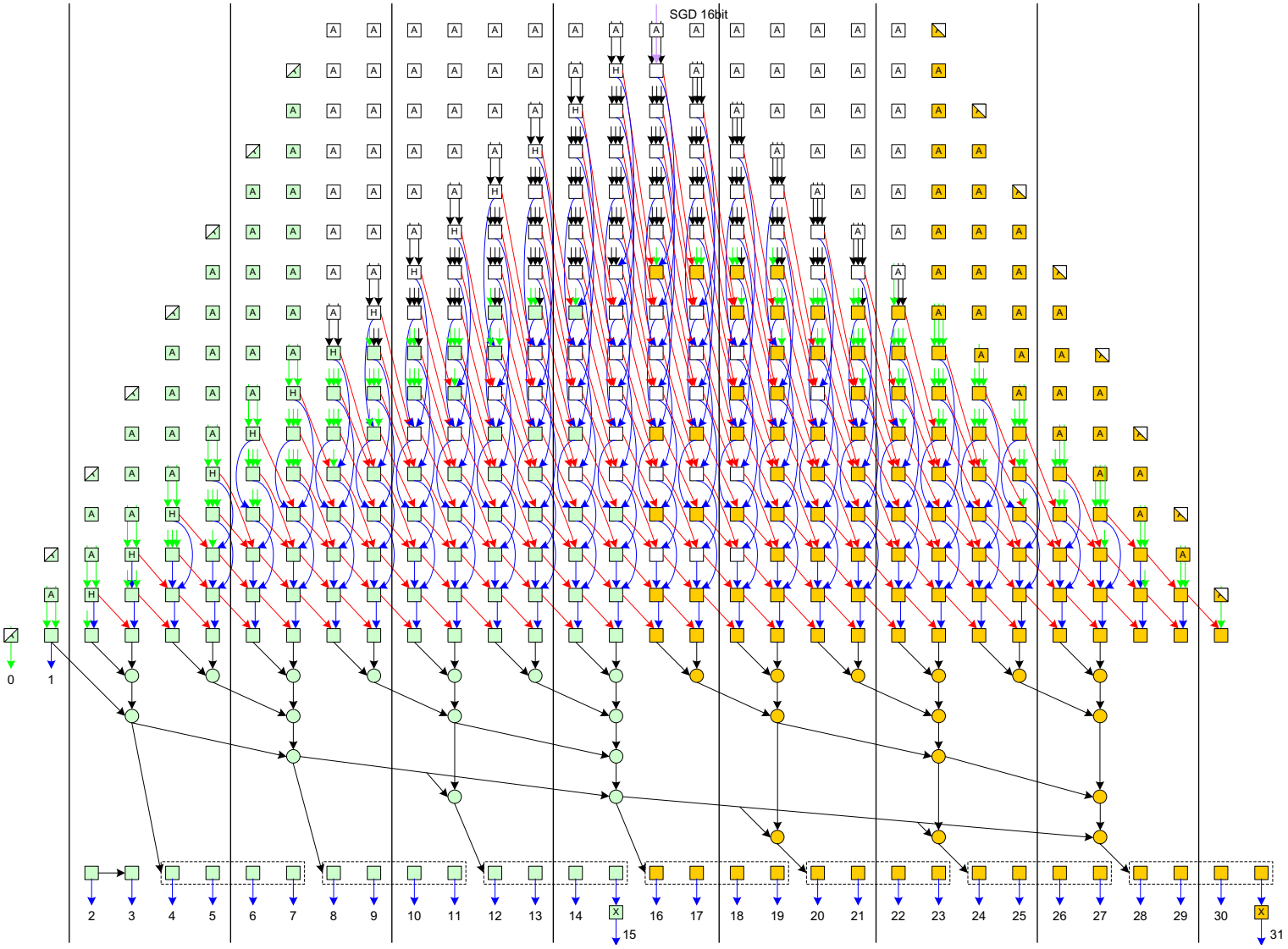
Initial Design: Carry-Save för PP-Reduktion



Initial Design: Logaritmisk Struktur => Låg Delay

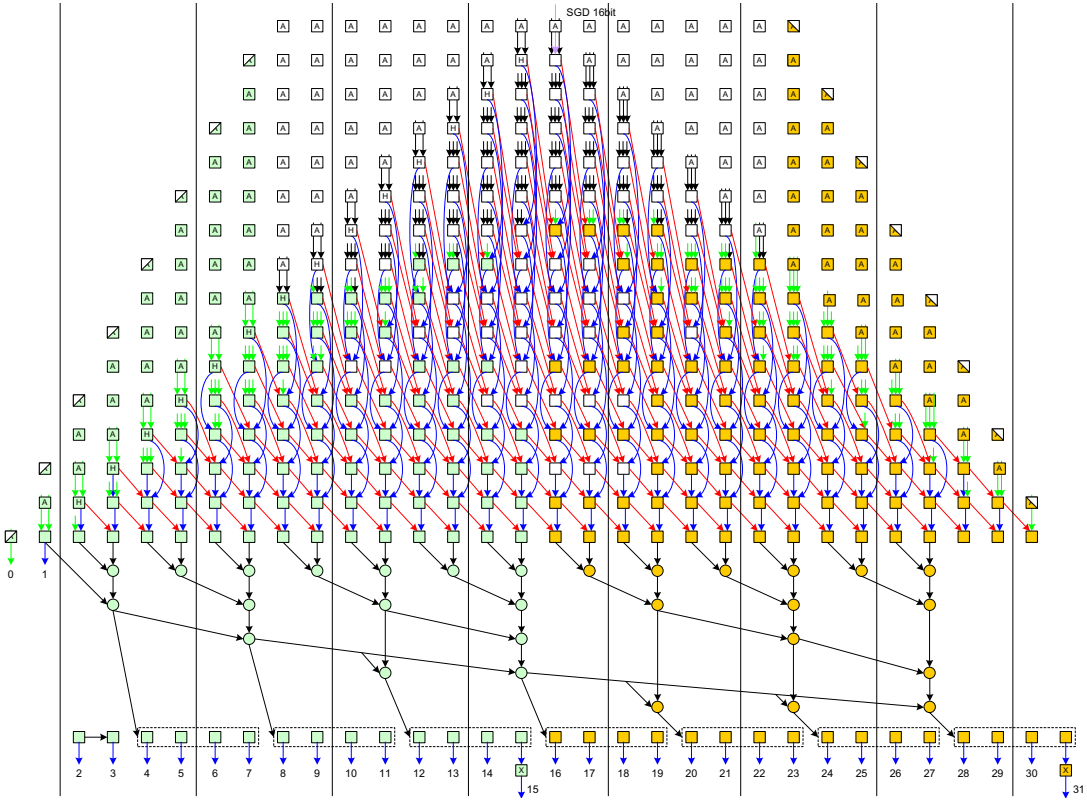


INITIAL DESIGN: 16-BITARS MULTIPLIKATOR



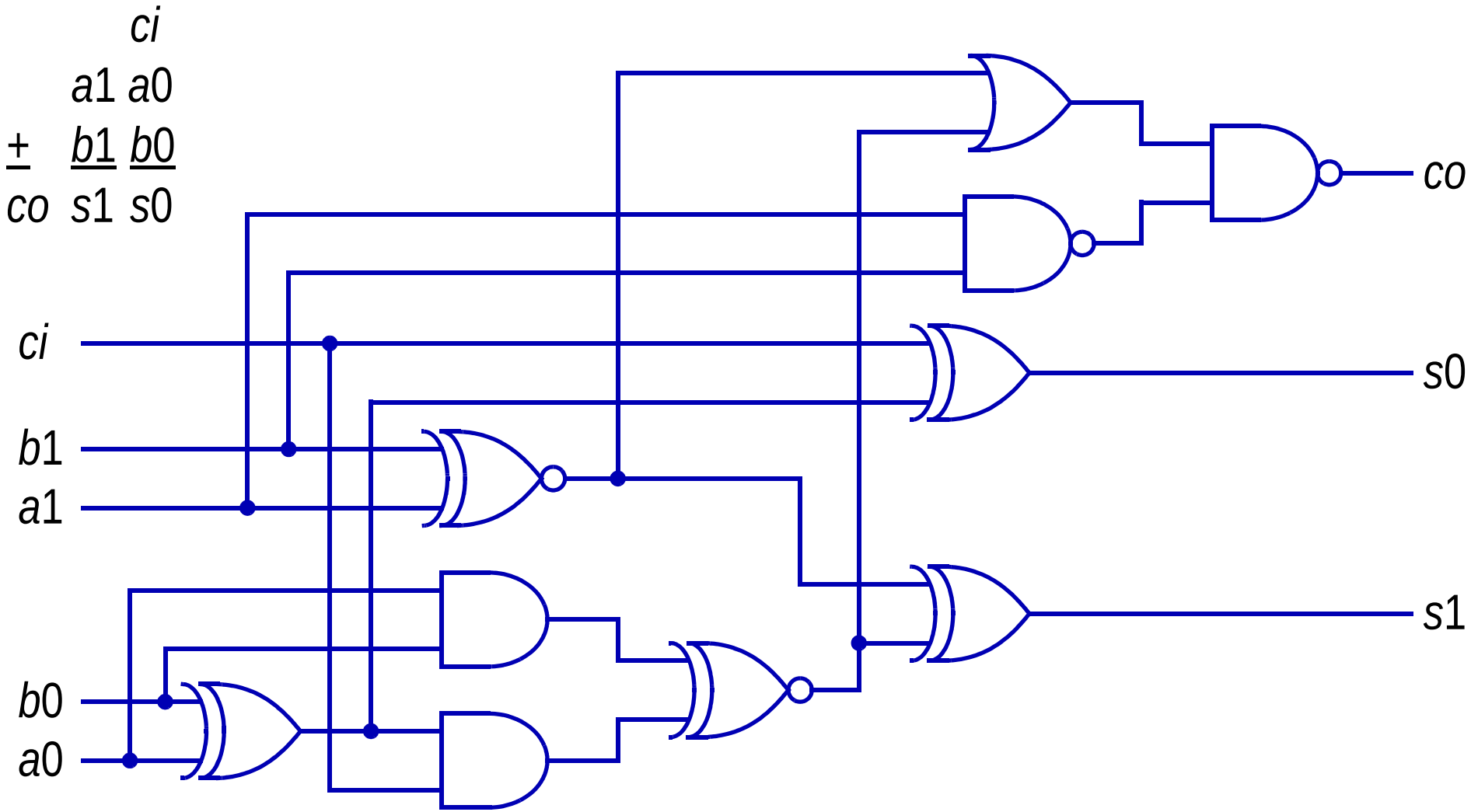
Utvärdering av Timing

Hur ska man kunna hitta fördröjningen i en komplicerad krets?



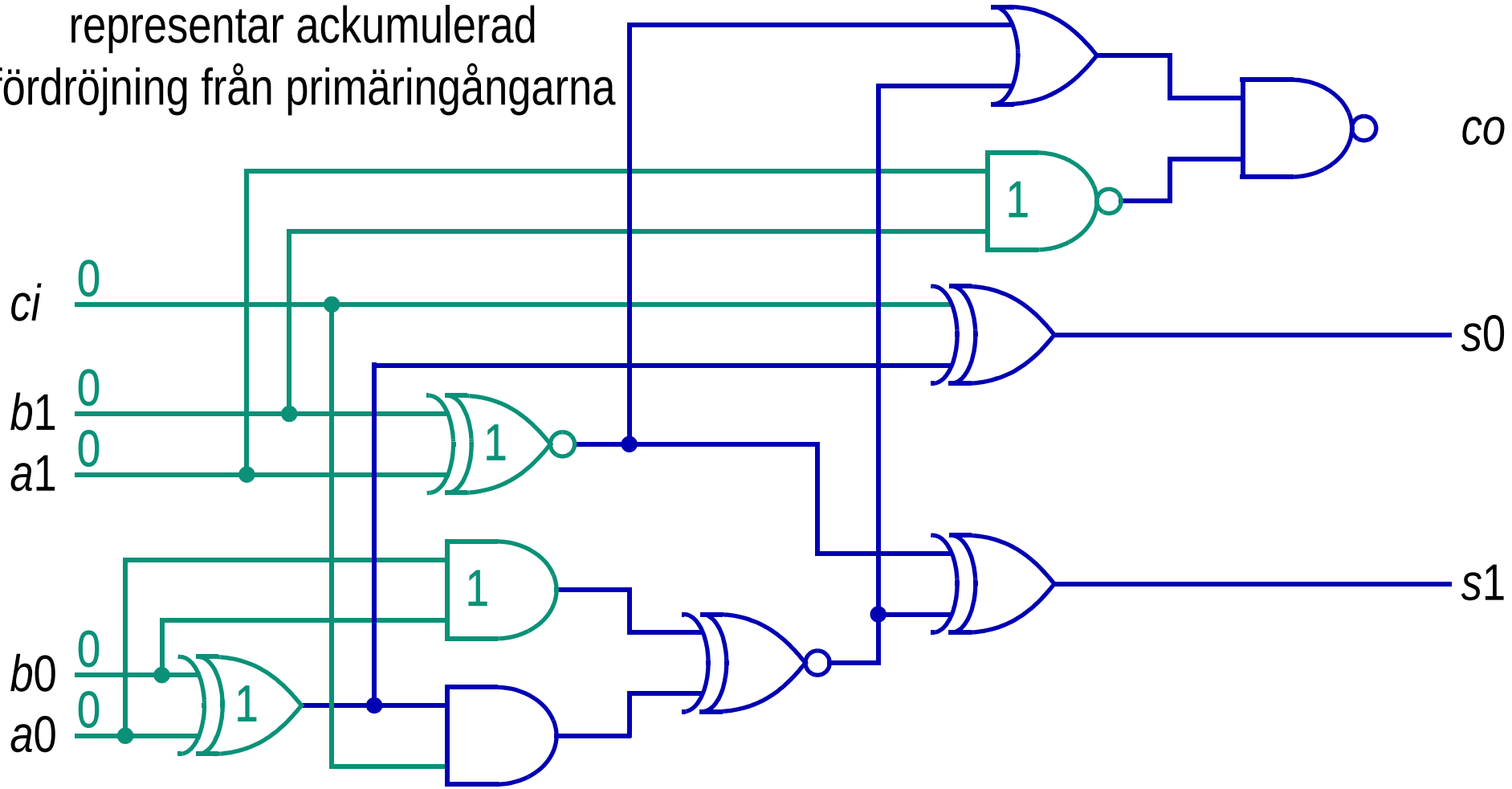
Statisk timinganalys (STA)!

UTVÄRDERING: STA 1(11)

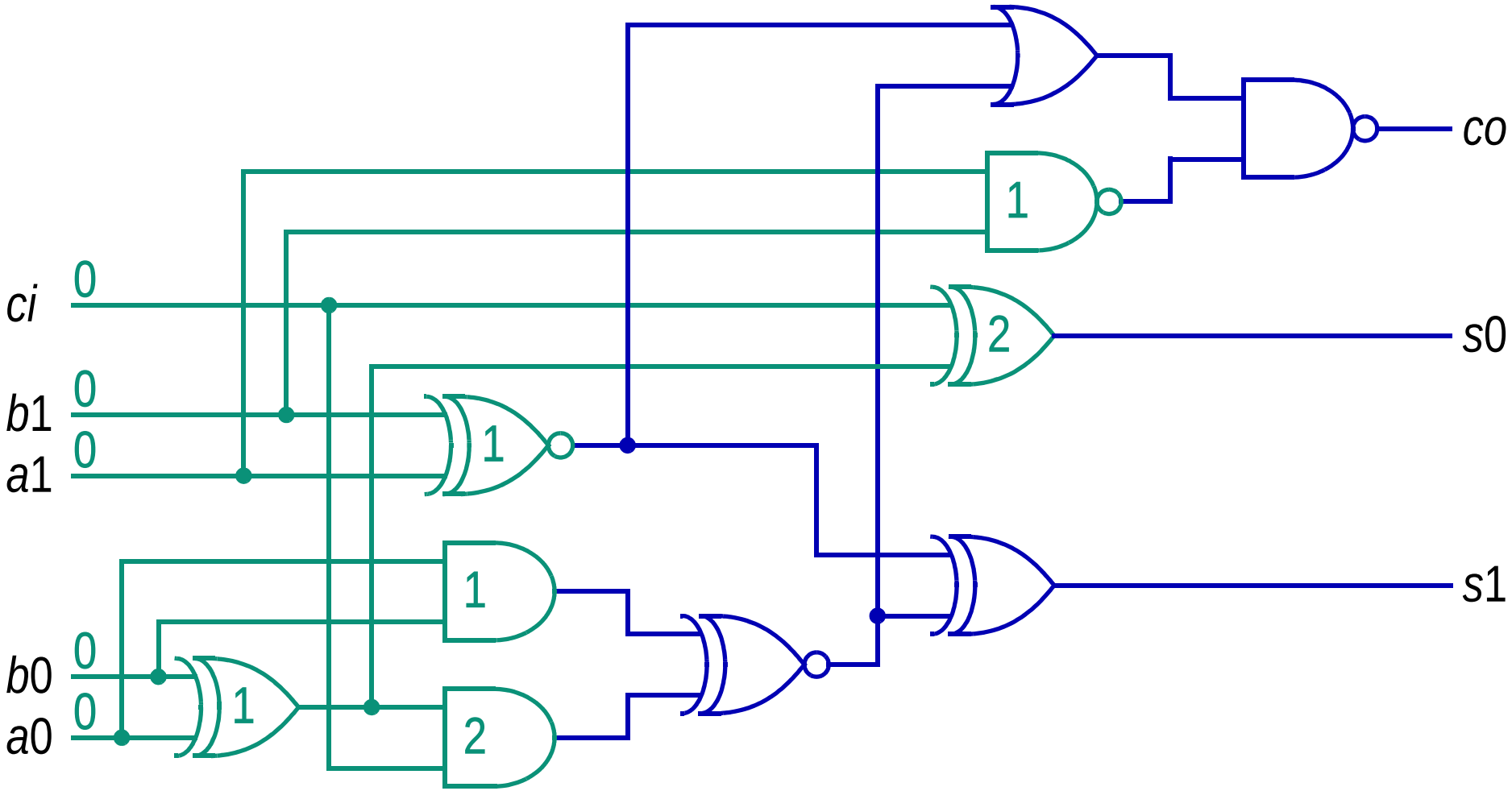


UTVÄRDERING: STA 2(11)

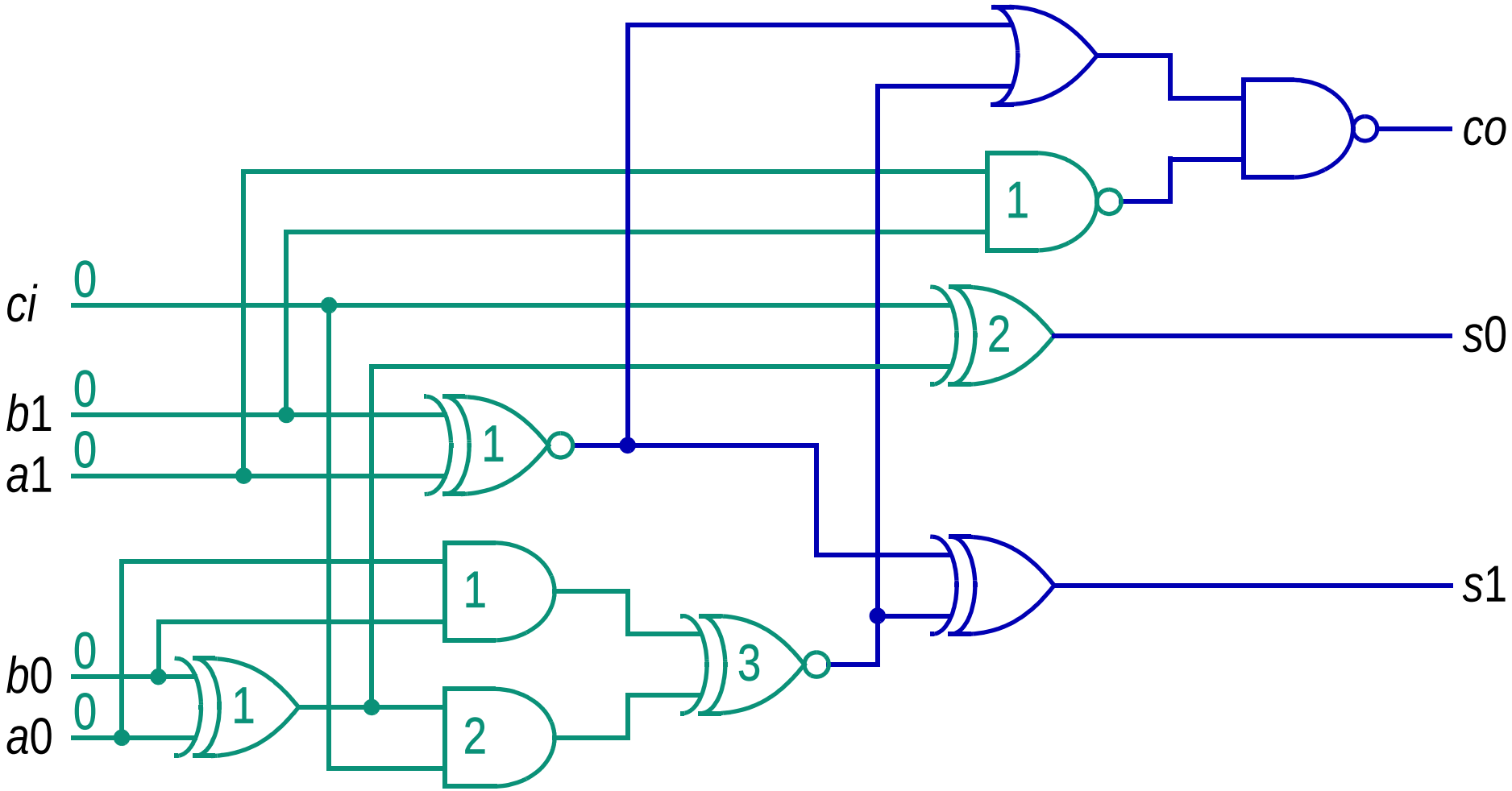
Siffran inom grindsymbolen representar ackumulerad fördröjning från primäringångarna



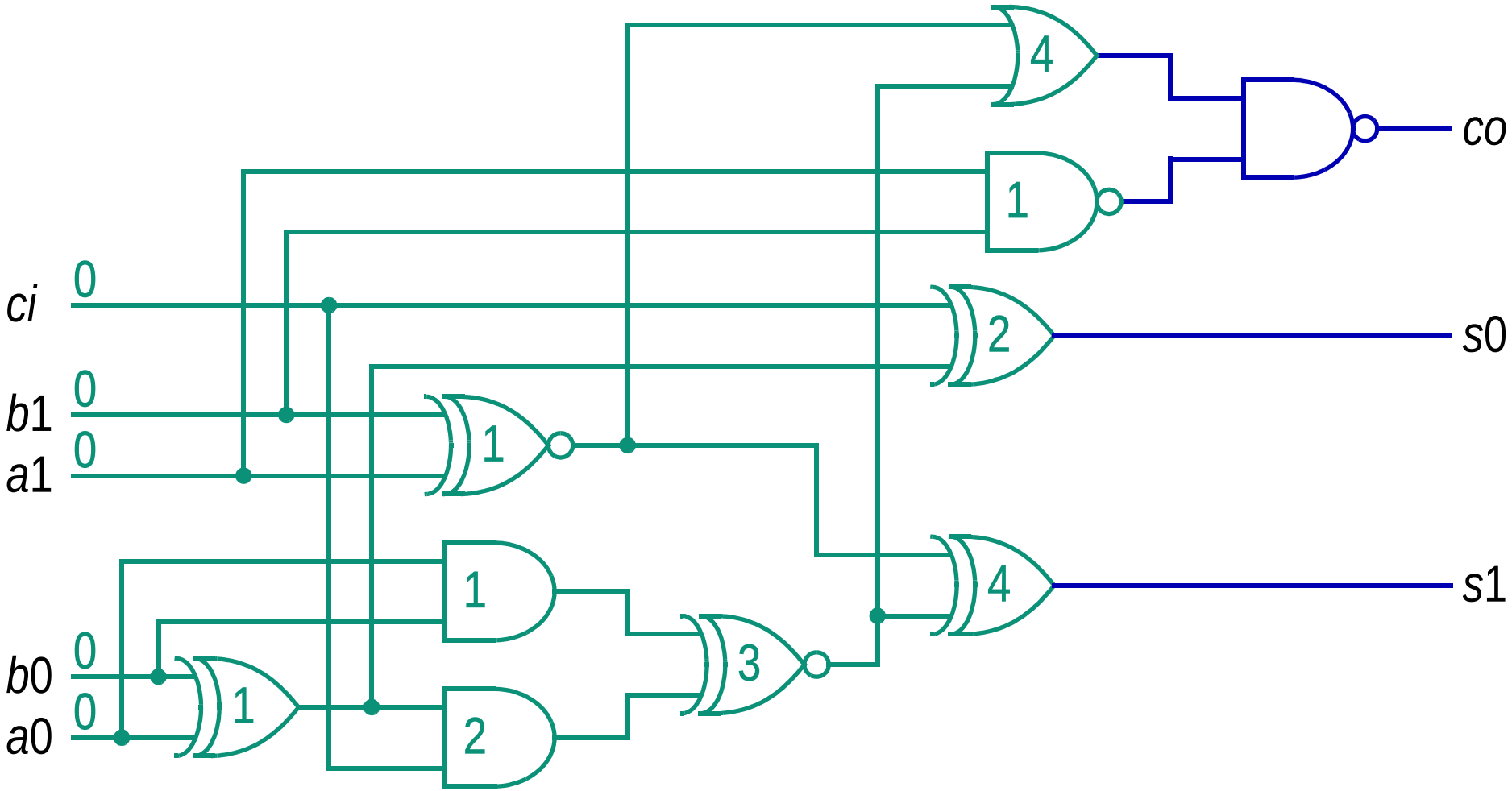
UTVÄRDERING: STA 3(11)



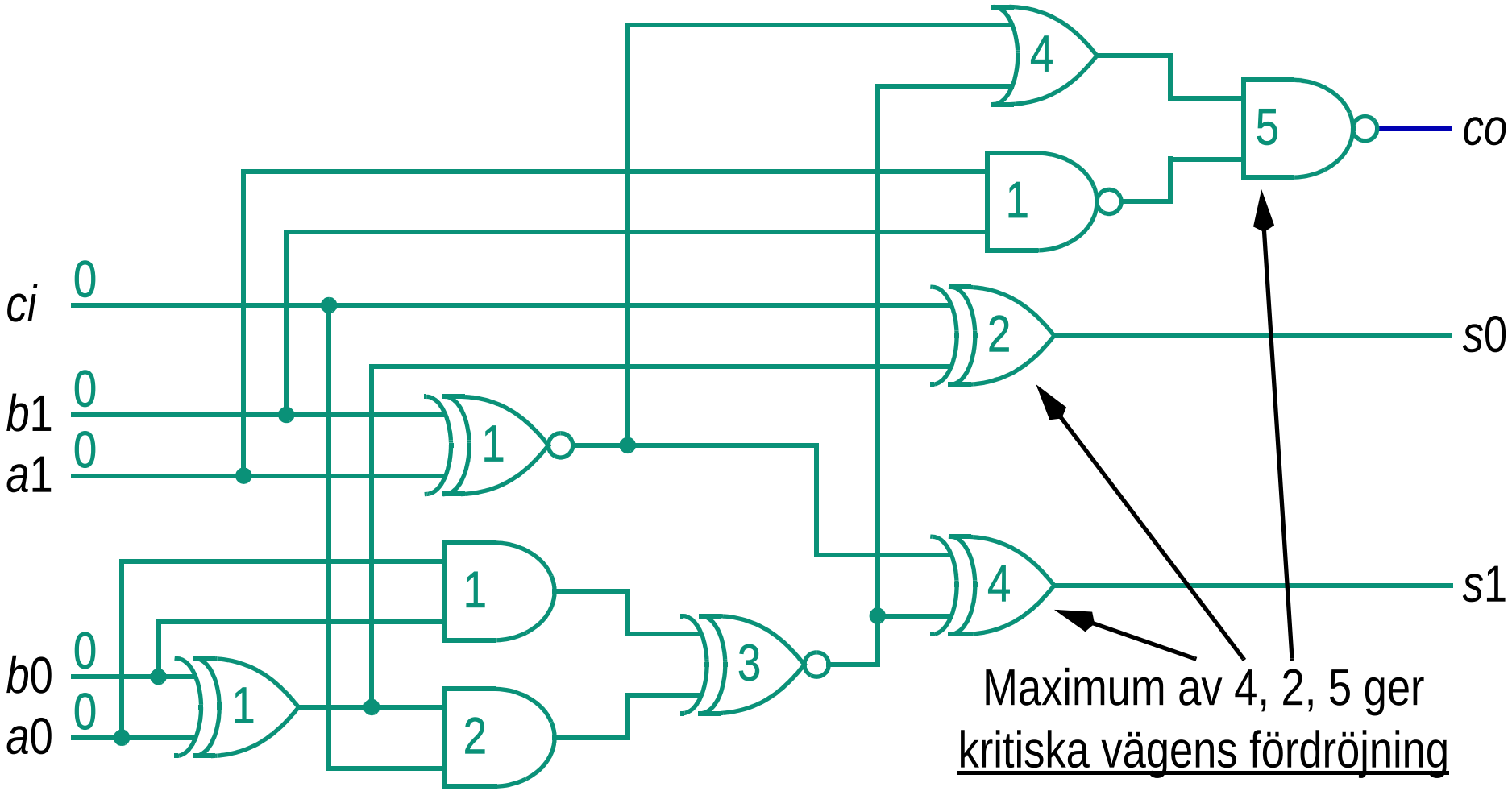
UTVÄRDERING: STA 4(11)



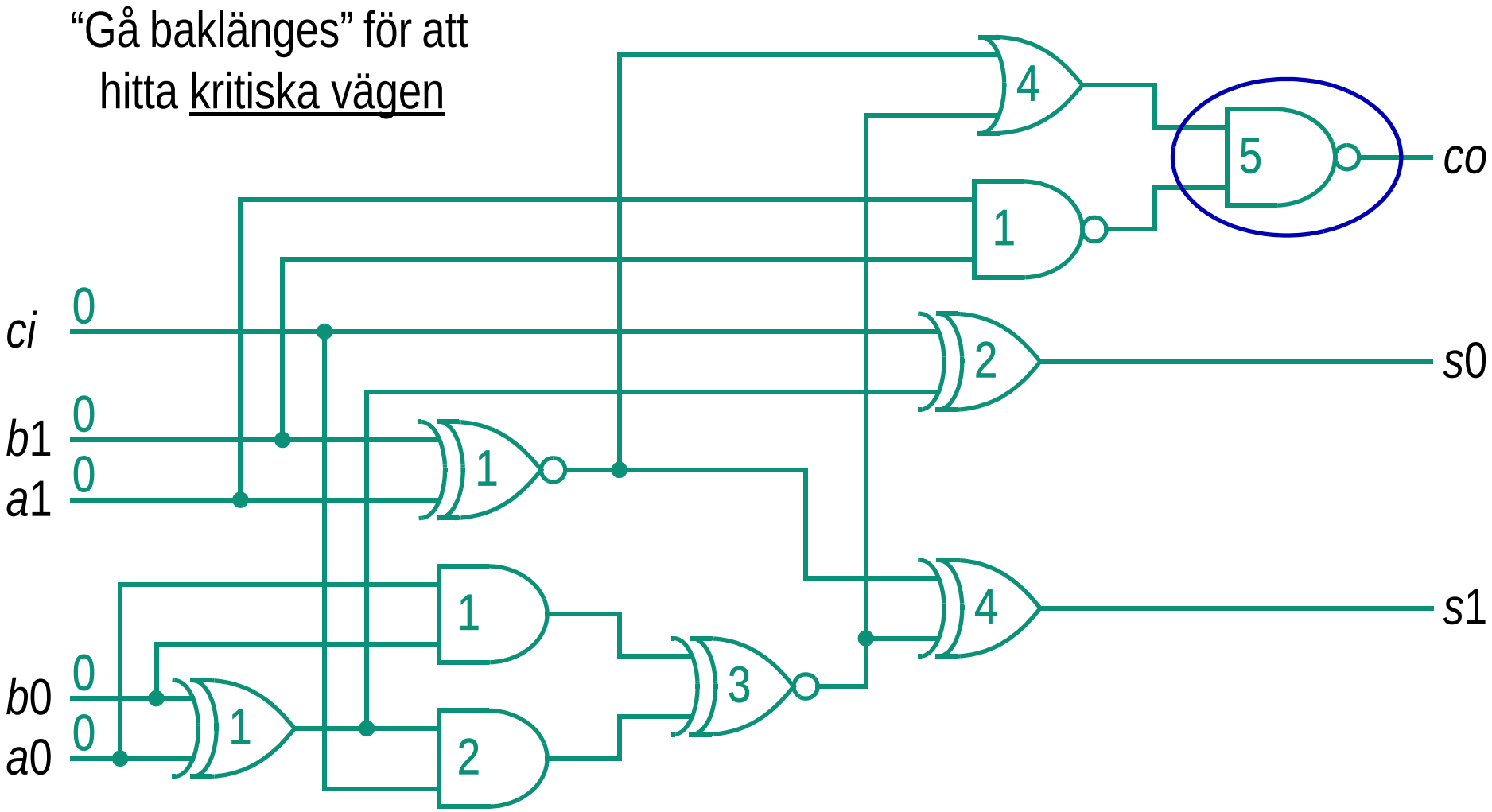
UTVÄRDERING: STA 5(11)



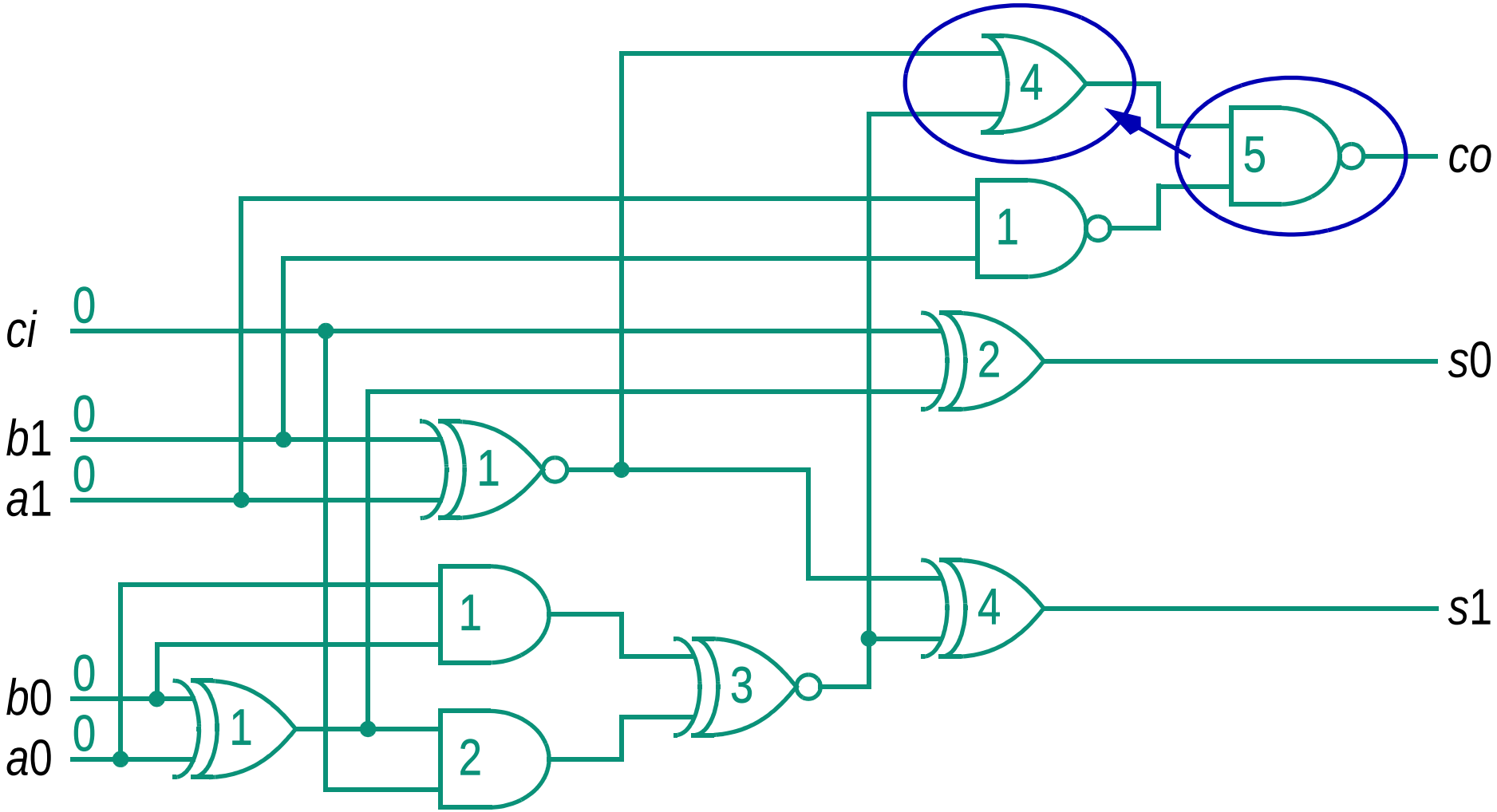
UTVÄRDERING: STA 6(11)



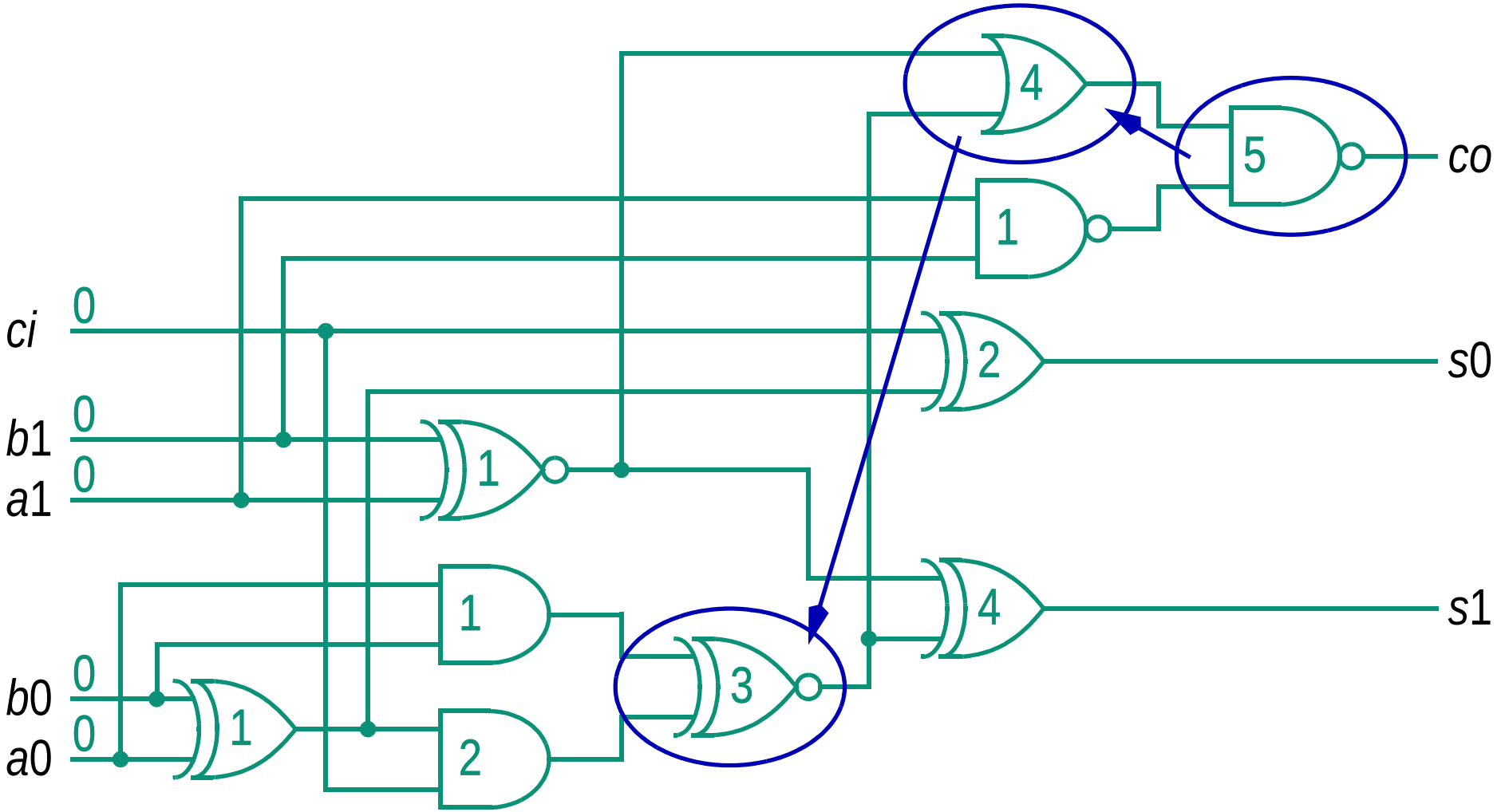
UTVÄRDERING: STA 7(11)



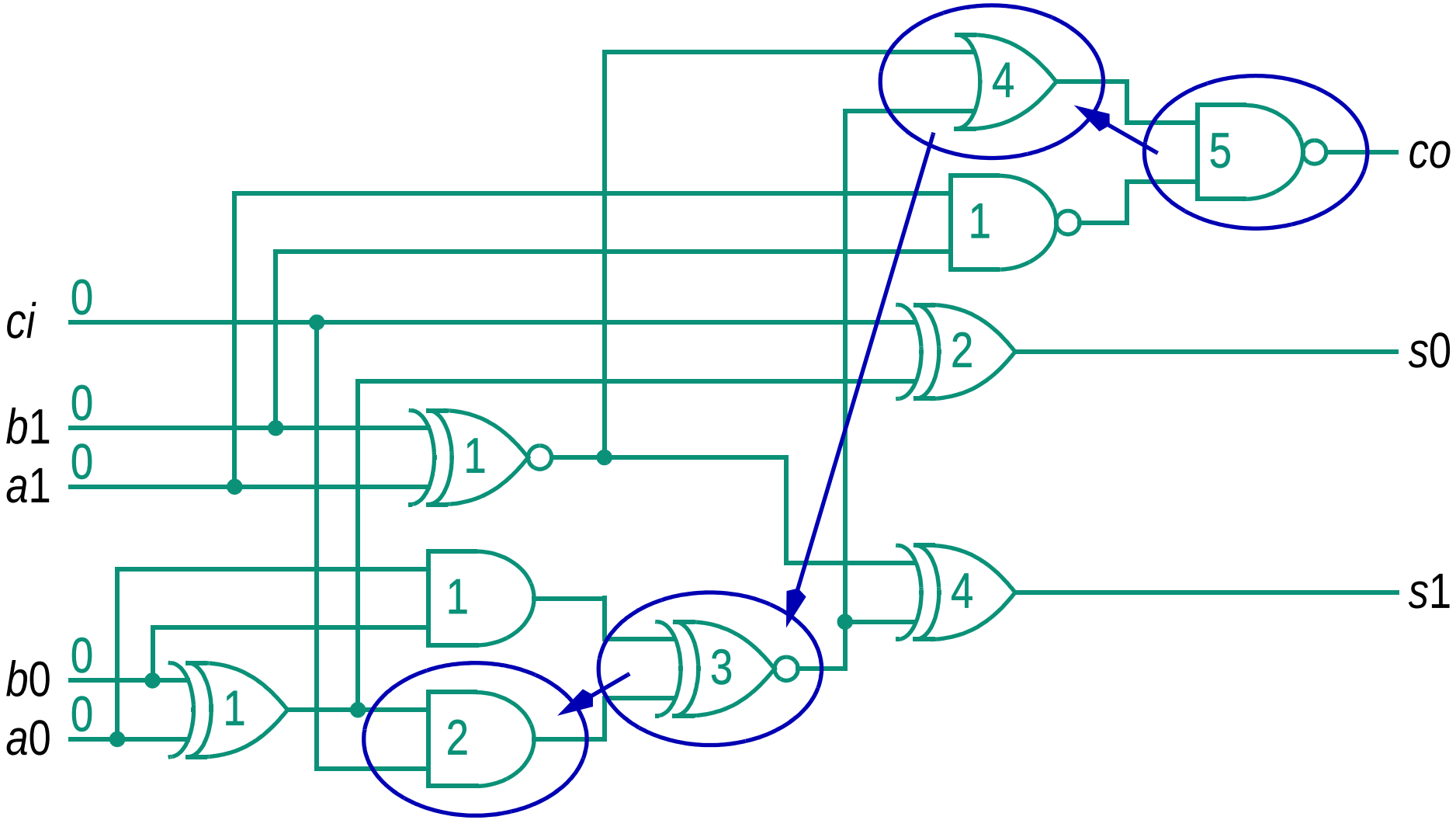
UTVÄRDERING: STA 8(11)



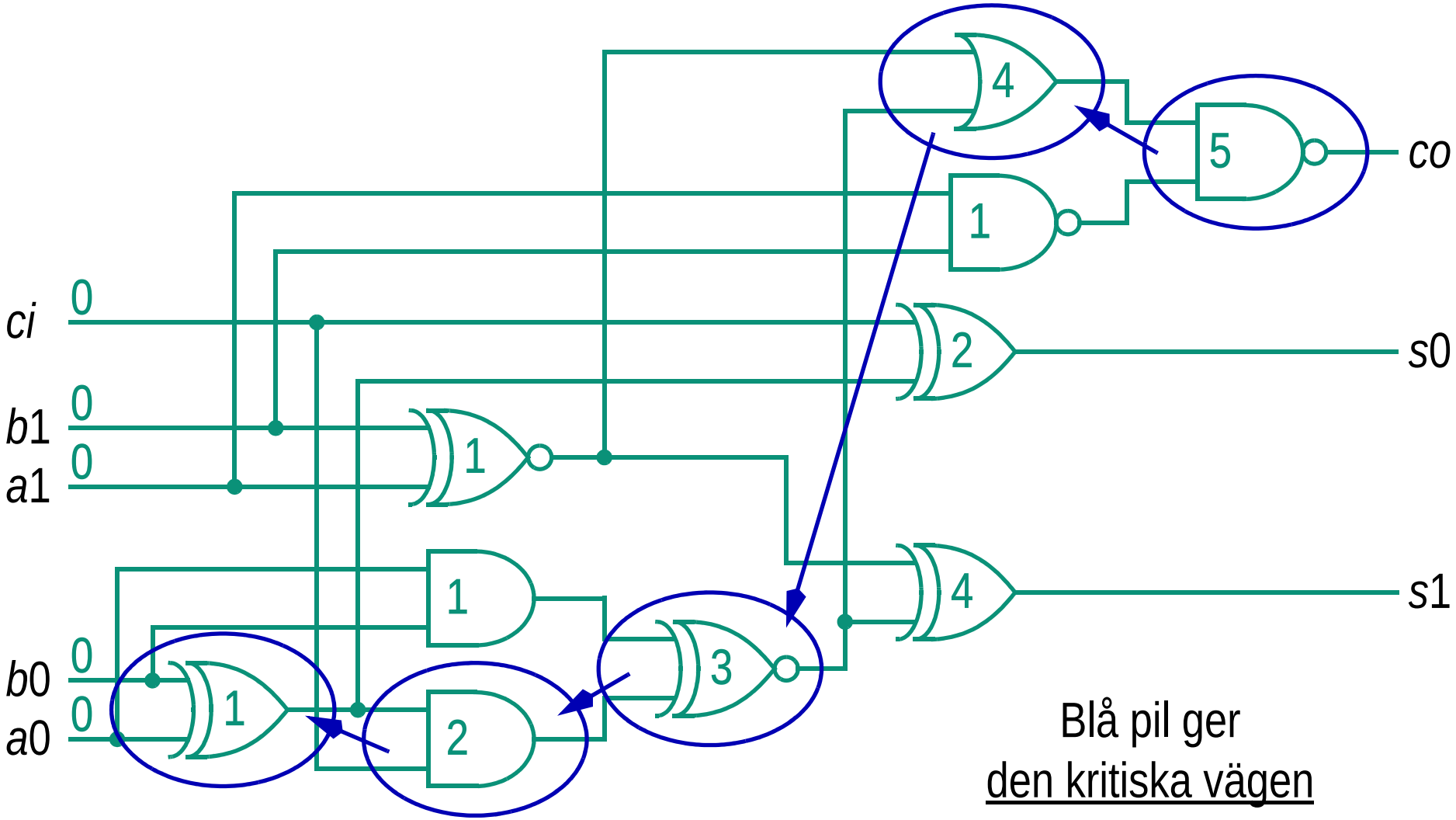
UTVÄRDERING: STA 9(11)



UTVÄRDERING: STA 10(11)

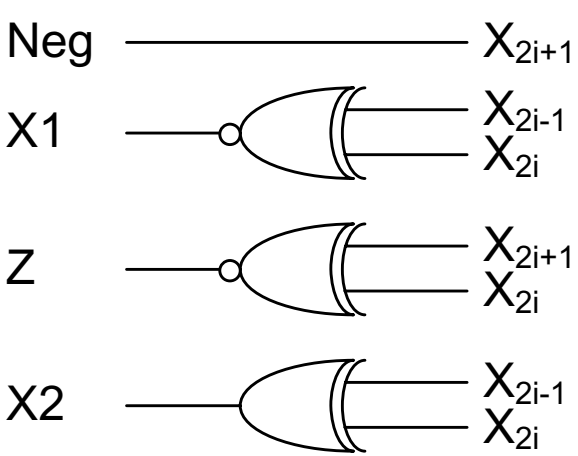


UTVÄRDERING: STA 11(11)

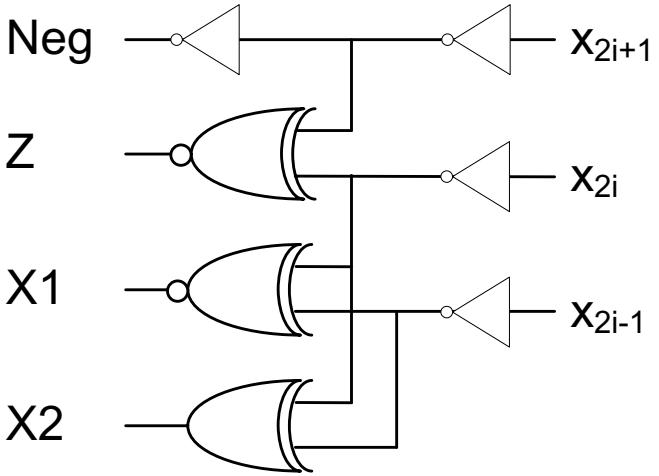


Utvärdering: Vi upptäcker ett Timingproblem

Ursprungskretsen klarar inte större fan-outs!

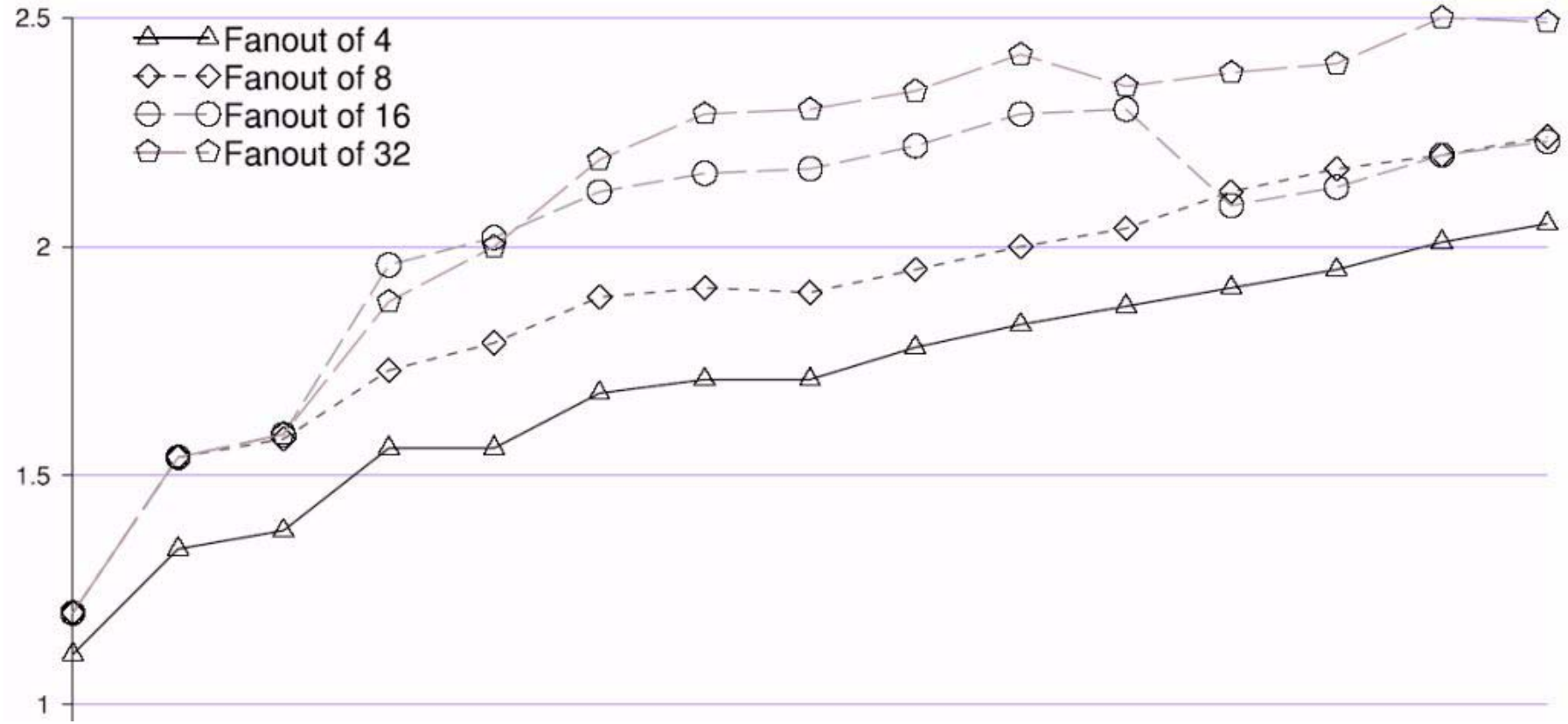


Ny, buffrad krets



UTVÄRDERING: OLIKA KODAR-FAN-OUT

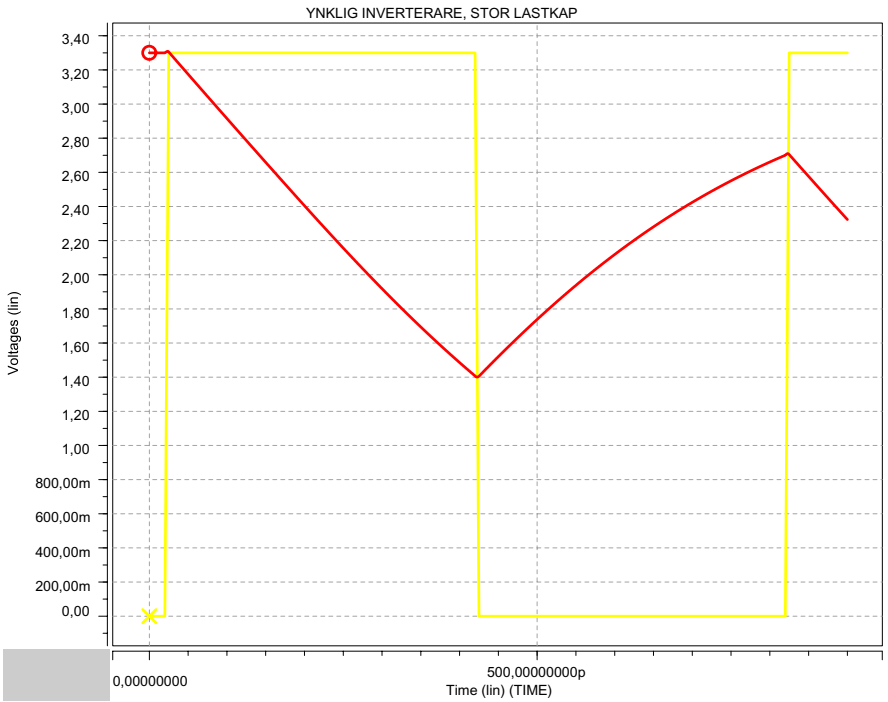
Fördröjning [ns]

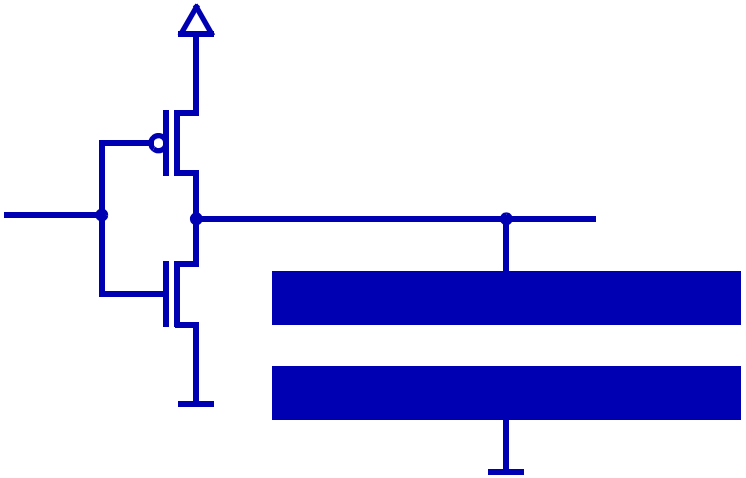


Multiplikatorbredd [4 -> 64 bitar]

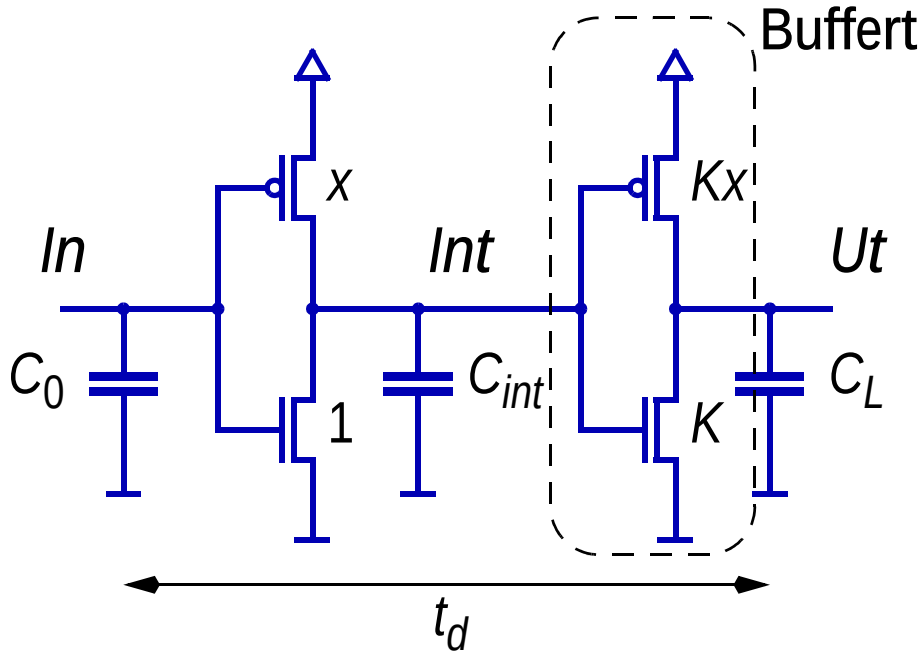
KASKADKOPPLING - BUFFRING 1(3)

- Att försöka driva en stor kapacitans med en ynklig grind är ineffektivt.





KASKADKOPPLING - BUFFRING 2(3)



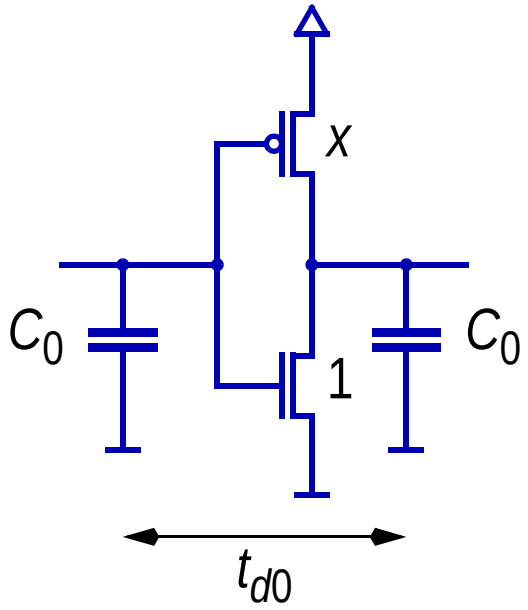
- För inverteraren som driver bufferten, är belastningskapacitansen $C_{int} \propto K$ (under förutsättning att buffertens två gateterminaler helt dominerar C_{int}).
- För bufferten gäller att transistorbredderna $W \propto K$.

KASKADKOPPLING - BUFFRING 3(3)

- Vi vet att $t_d \propto C_L / W$. Därför gäller $t_{d0} \propto C_0 / 1$.

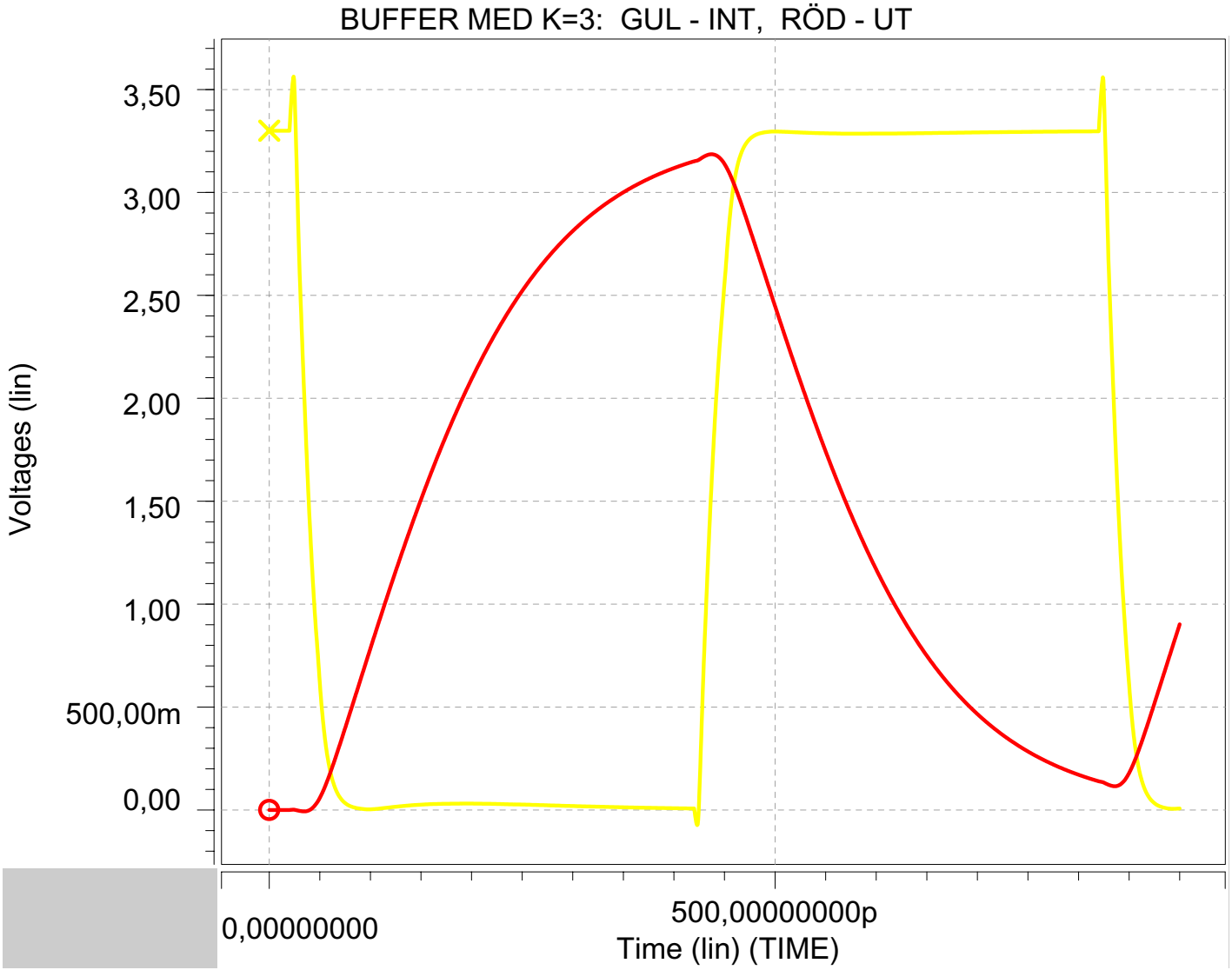
1. För den första inverteraren: $t_{dINV} = K\, t_{d0}$
   eftersom C_{int} ökar K gånger.

2. För bufferten: $t_{dBUF} = \dfrac{C_L / C_0}{K}\, t_{d0}$ eftersom
   dess C_{ut} är C_L / C_0 gånger större och
   dess W är K gånger större (än för kretsen till höger).

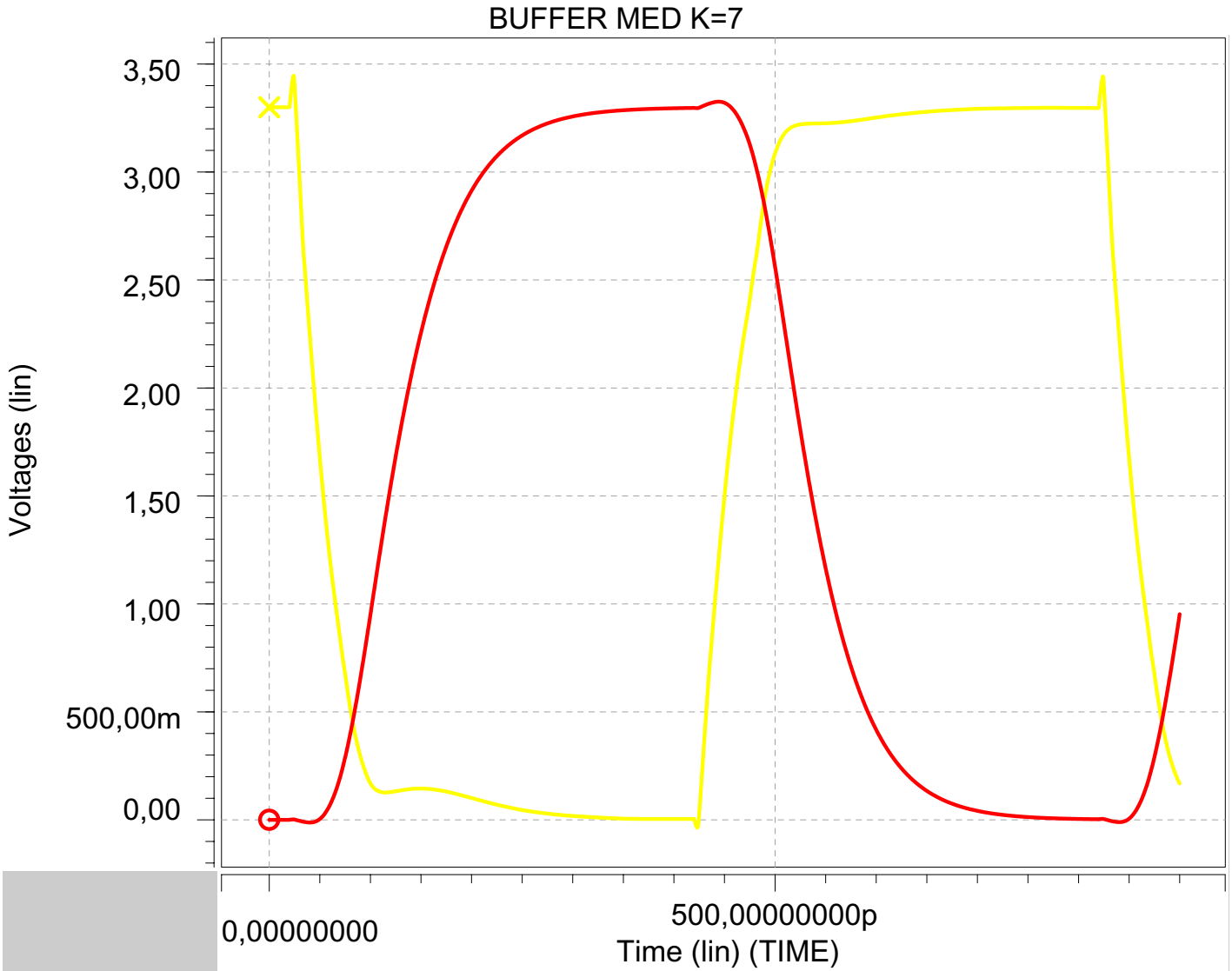


- Leta efter minimal delay: $\dfrac{\partial t_d}{\partial K} = t_{d0} \cdot \left(1 - \dfrac{C_L / C_0}{K^2}\right) = 0 \Rightarrow K = \sqrt{\dfrac{C_L}{C_0}}$.

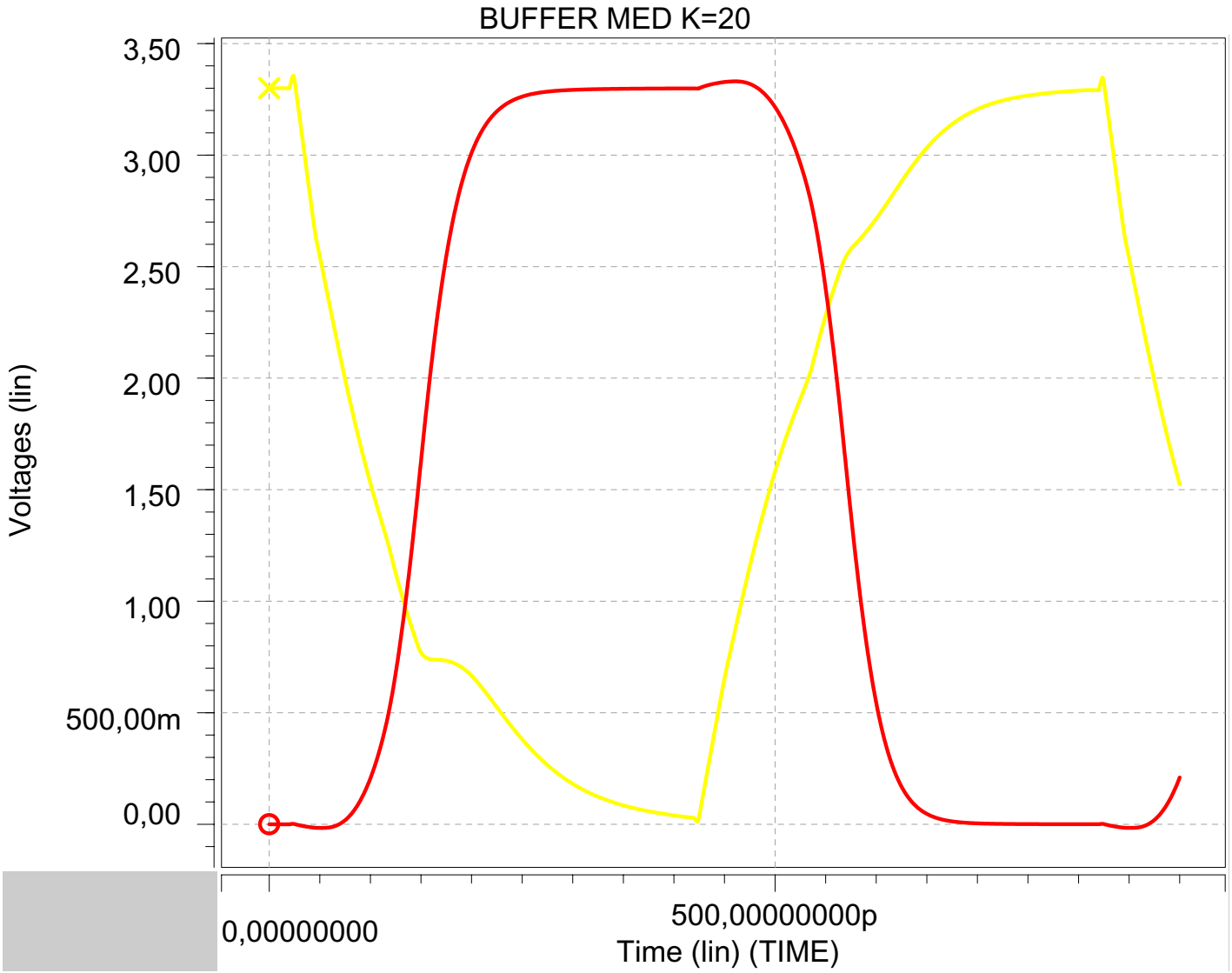
BUFFERT MED FÖR LITET K - DÅLIG *UT*



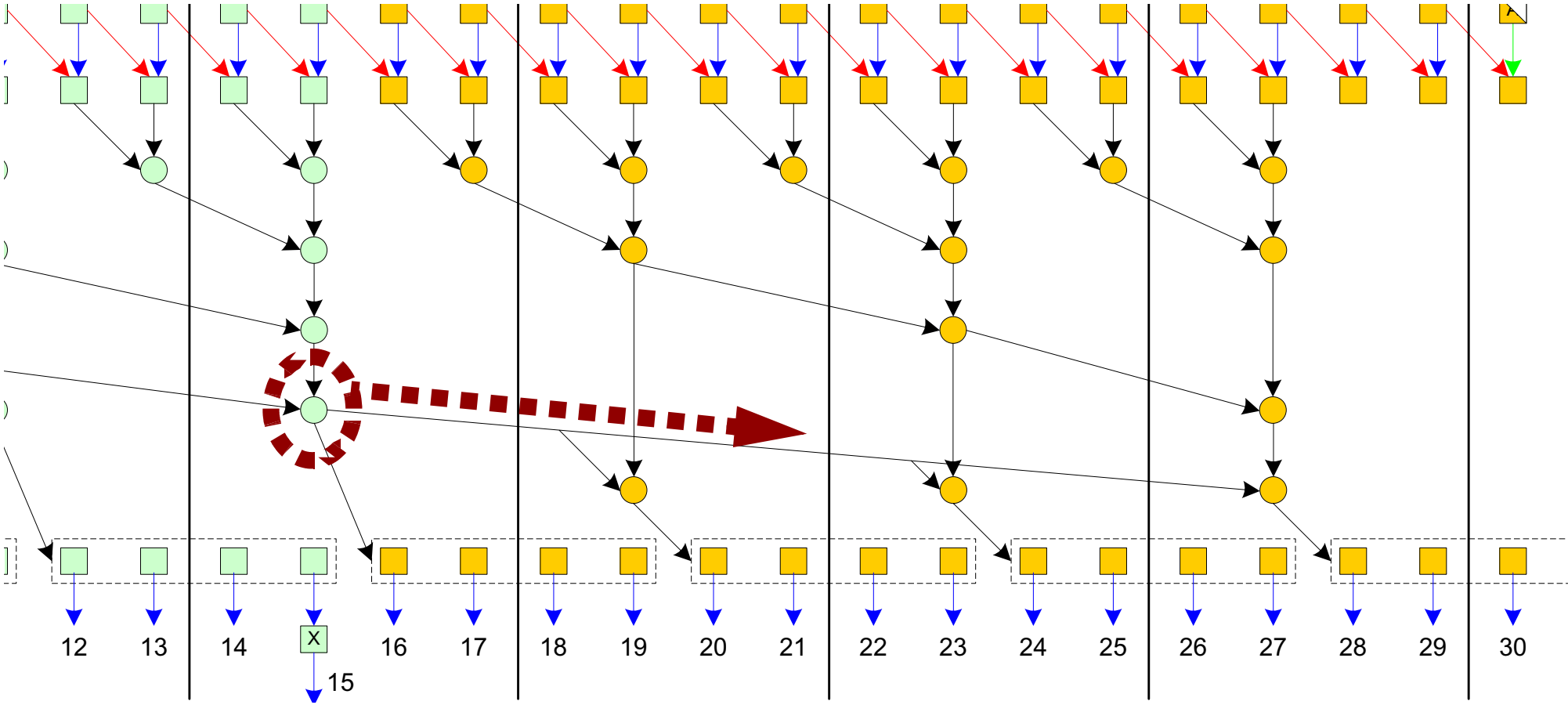
Buffert med Lagom K



BUFFERT MED FÖR STORT K - DÅLIG *INT*



Buffring i Multiplikatorns Slutadderare



Inte enbart hänger man på ett stort antal grindar,
utan ledningen som ansluter dem är i regel ganska lång.

64-b Adder från Intel