En seriell adderare
2B1430 Konstruktion av digitala integrerade kretsar – LSI

Magnus Jansson 740909-0276
Fredrik Neiderud 781227-9318
Rickard Norström 770428-0432

2002-06-23

1 Sammanfattning

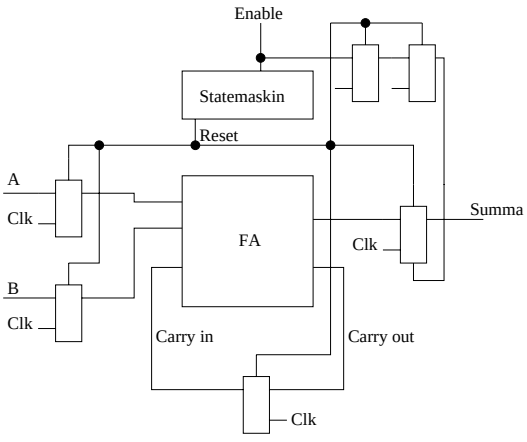
Detta är en rapport för projektdelen av kurs 2B1430 Konstruktion av digitala integrerade kretsar – LSI. Den behandlar vår konstruktion av en seriell adderare och dess delar.

2 Introduktion

När vi skulle välja projekt tyckte vi att en adderare verkade roligt att prova på. Då vi började titta på problemet tyckte vi att det behövdes en statemaskin för att kunna styra adderaren på det sättet som vi ville att den skulle bete sig. För att kunna få stabila insignaler, lade vi därför in register på ingångarna. Detta gjordes även på utgången. Med statemaskinen styr vi inte bara adderaren utan även in- och utregistren.

3 Teori

Vi har byggt en seriell adderare, som styrs av en enkel statemaskin. En seriell adderare lägger ihop två sekvenser av enbitars tal och skickar ut summan på en utgång. Adderaren fungerar så att eventuella carrybitar adderas tillsammans med insignalerna i nästa tidsenhet. Exempelvis blir $0011 + 0101 = 01101$, detta eftersom $1 + 1 = 10$. Adderaren är uppbyggd av en fulladder, en statemaskin, ett antal register och ett tristateregister.



Figur 1: Vår seriella adderare

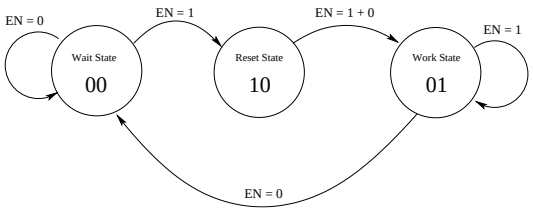
Alla de olika byggblocken är beskrivna i detalj nedan.

4 Vår implementation

4.1 Statemaskinmaskin

Som styrmekanism till den seriella adderaren har vi valt en enkel statemaskin. Statemaskinen ska starta med att göra en reset av adderarens minne. Till detta behövs tre tillstånd.

- Ett väntetillstånd som väntar på enablesignalen.
- Ett reset-tillstånd som varar under en klockcykel och går sedan till det sista tillståndet.
- Work, som väntar på att enable ska sluta vilket medför att räknandet är över.



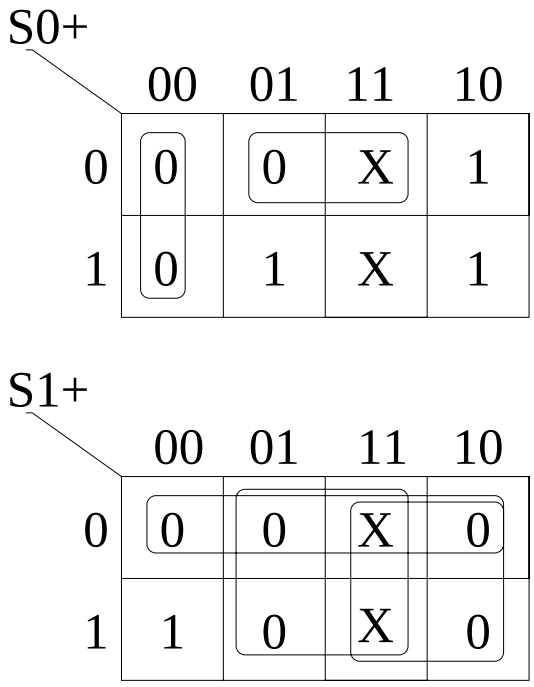
Figur 2: Statemaskinens lägen

Alla dessa tillstånd skapar en sanningstabell över tillståndsövergångarna. (se fig 3) Denna tabell be-

EN	S1	S0	S1+	S0+
0	0	0	0	0
0	0	1	0	0
0	1	0	0	1
0	1	1	–	–
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	–	–

Figur 3: Sanningstabellen

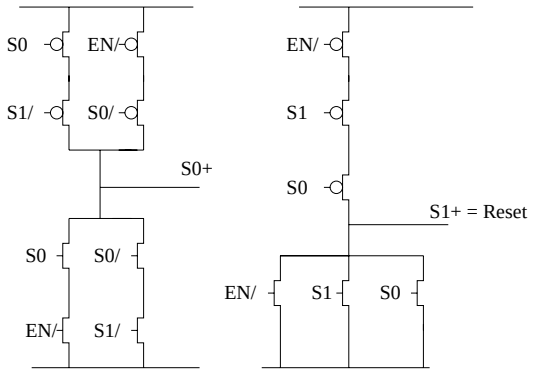
arbetas sedan till ett Karnaughdiagram över övergångarna. Ur diagrammet kan man härleda de boolska ekvationerna för PDN för de enskilda variablerna i statemaskinen. (se fig 4)



Figur 4: Karnaughdiagram

$\overline{S_{0+}} = (\overline{S_1} \times \overline{S_0}) + (\overline{EN} \times S_0)$
$\overline{S_{1+}} = \overline{EN} + S0 + S1$

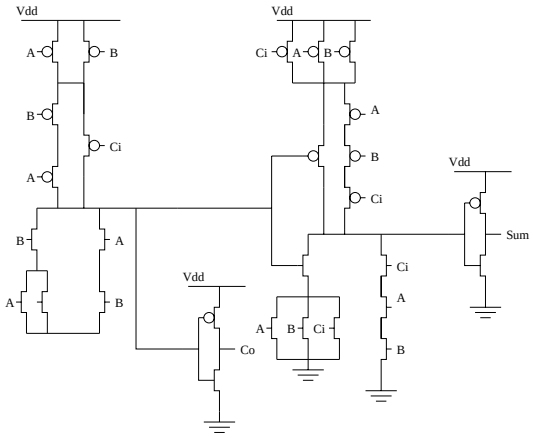
Och på transistornivå kan ett kretsschema se ut så här. (se fig 5) Utgångarna på kombinatoriken kopplas till var sitt register. Och eftersom S_{1+} dessutom används som resetsignal till resten av adderaren så har den en buffrad utgång.



Figur 5: Statemaskin på transistornivå

4.2 Adderare

Kärnan i vår seriella adderare består av en heladderare, en så kallad Full-adder. Konstruktionen kommer från kursboken, Rabaey (sid 389). Den har två ingångar (A och B) och två utgångar (Sum och Carry out). Adderaren är helt kombinatorisk och arbetar utan klocka. Det som kommer in på A och B kommer så småningom ge en utsignal på Sum och Carry.



Figur 6: Adderaren på transistornivå

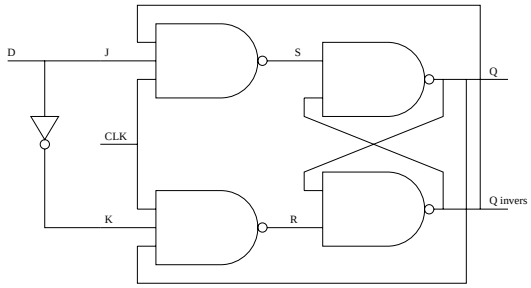
Det finns många olika tillvägagångssätt för att konstruera en adderare. Den konstruktion vi har valt är väldigt enkel att förstå sig på och dessutom alldeles utmärkt i vår tillämpning. I mer kritiska tillämpningar kanske man skulle vilja välja en snabbare eller ytmässigt mindre lösning. Dock är adderaren det minsta byggblocket i vårt projekt, och därför är inte ytan kritisk. Inte heller hastigheten är kritisk, eftersom de andra komponenterna är klockade. Adderaren hinner gott och väl räkna ut nästa värde innan de nya är redo att adderas.

4.3 Register

Vår seriella adderare innehåller också ett antal register. Dessa är uppbyggda av två stycken D-vippor som är kopplade i serie. Den första är aktiv hög och den andra aktiv låg. På så vis blir registret negativt flanktriggat och man slipper problemet att D-vipporna är "genomskinliga" på hög klocksignal. Registret har 3 ingångar och 2 utgångar. Ingångarna är D, klocka och reset.

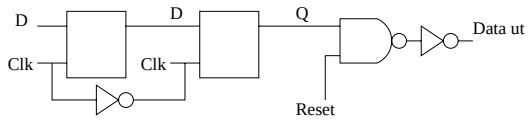
- D är data in
- Klockan förklarar sig själv
- Reset möjliggör nollställning av utsignalen.

Reset används under intitialiseringen av kretsen, för att vara säker på vad som finns på utgångarna. Utgångarna är Q och $\overline{Q}$, alltså data ut och dess inversa signal. Båda signalerna är tillgängliga utåt, eftersom vi då slipper invertera utsignalen flera gånger. Görs det i registret, behöver det bara göras en gång. D-vipporna som utgör registret är i



Figur 7: En D-vippa

sin tur uppbyggt av olika mindre funktionsblock. (se fig 7) Först konstruerade vi NAND-grindarna, som sen sattes ihop till en SR-vippa. Den kopplades i sin tur ihop till en JK-vippa. Ingångarna på JK-vippan kopplas samman, via en inverterare, och bygger på så vis upp en D-vippa, som är klockad. För att få till reset-funktionaliteten har vi kopplat in ytterligare en NAND-grind precis innan utgången på registret. (se fig 8) Registret kan

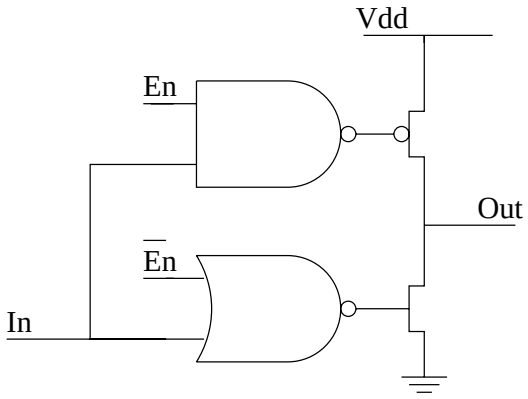


Figur 8: Register

säkert göras mindre och snabbare om man tillverkar det som ett enda funktionsblock direkt istället för att bygga det av små block. Däremot tar det mycket längre tid att bygga och det är mycket större risk att fel smyger sig in i konstruktionen. Mot denna bakgrund valde vi att göra som vi har gjort. Testkörningar visar att registret kan köras med klockfrekvenser upp över 1 GHz. Inte speciellt snabbt om man jämför med andra register, men gott och väl tillräckligt för vår applikation. Dessutom är inte ytan något problem. Hela vår seriella adderare får gott och väl plats i den pad-ring som behövs för att kunna ansluta den till omvärlden.

4.4 Tristate

Tristateregistret som sitter på adderarens utgång fungerar nästan precis som de register som sitter på ingångarna (se avd 4.3). Faktum är att det är baserat på de vanliga registrena och byggt utifrån dem. Det som skiljer dem åt är att Tristateregistret har en ingång mer och en utgång mindre. Den extra



Figur 9: Den del som skiljer Tristateregistret från ett vanligt register

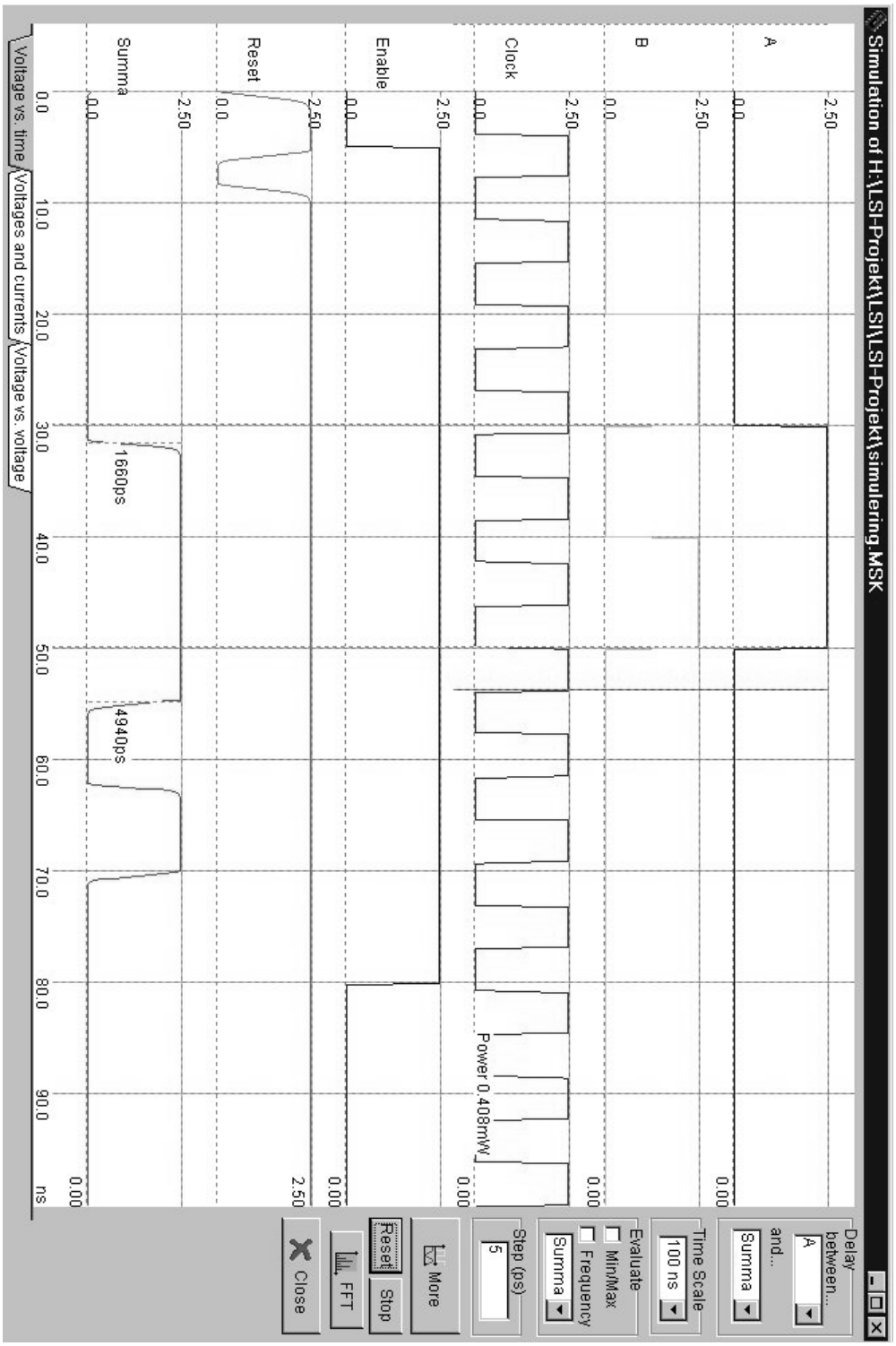
ingången kallas enable. Med $EN = 1$ arbetar registret som vanligt. Med $EN = 0$ däremot, stängs utgången av och utsignalen blir högimpediv. Den utgång som saknas är $\overline{Q}$, det vill säga den inverterade utsignalen. Anledningen till att den saknas är att $\overline{Q}$ endast behövs internt. Ut från vår seriella adderare är vi bara intresserade av $A + B$ och inget annat. Dessutom sparas en hel del yta genom att plocka bort den utgången. Om den hade varit med, behövs tristatestyrning på den med. Då blir det alltså dubbla tristatedelar i den lövcellen och det är det svårt att få plats med.

5 Slutsatser

Vi har alltså byggt en seriell adderare, som styrs av en statemaskin. Våra simuleringar visar att adderaren fungerar som tänkt. Förmodligen kan adderaren göras både mindre och snabbare, det har vi insett allt eftersom projektet har fortskridit. Dock har det inte varit läge att börja om från början, bara för att spara några kvadratmikrometer. Sist men inte minst kan vi konstatera att vi har lärt oss en hel del om LSI-konstruktion.

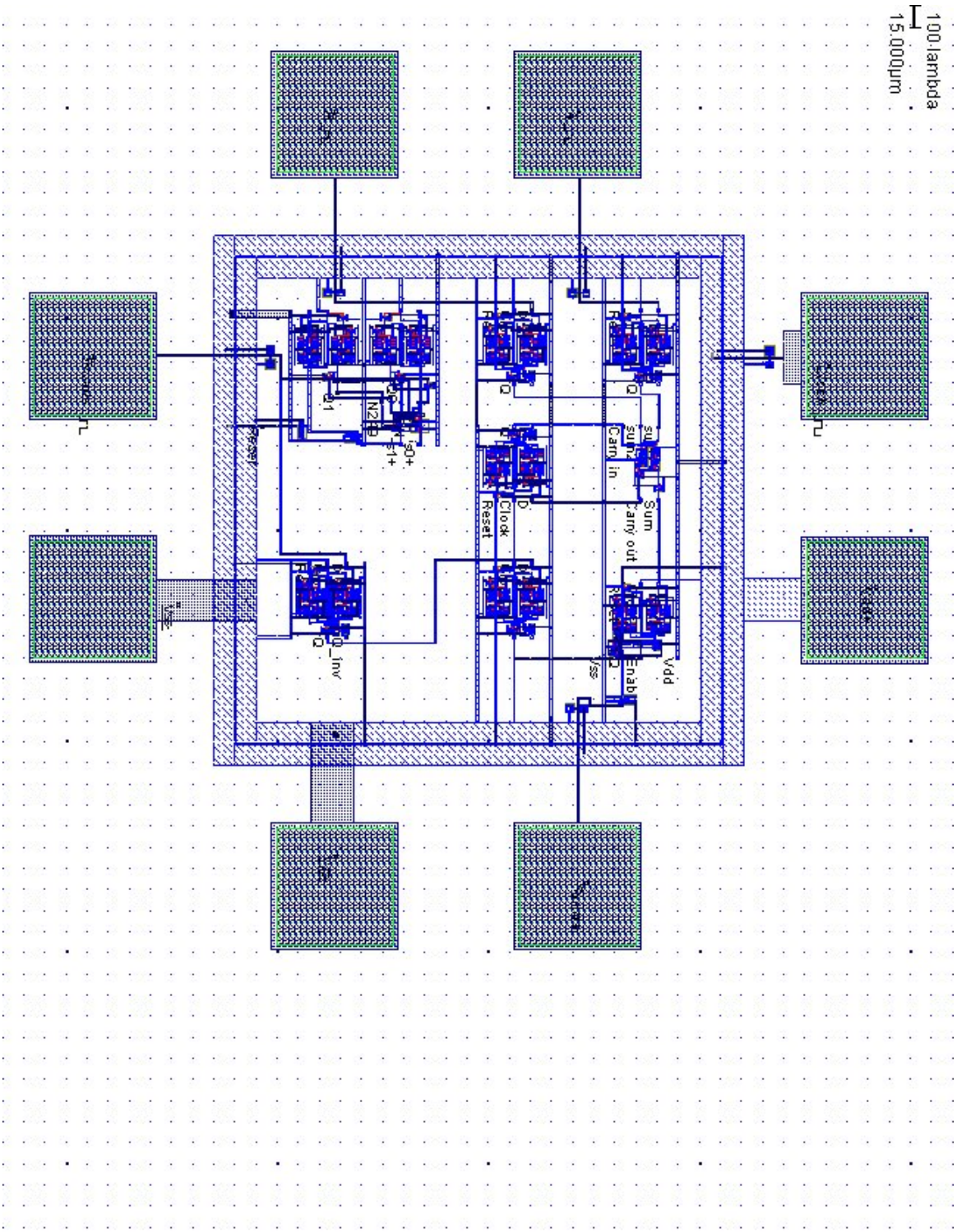
A Appendix

A.1 Tidsdiagram



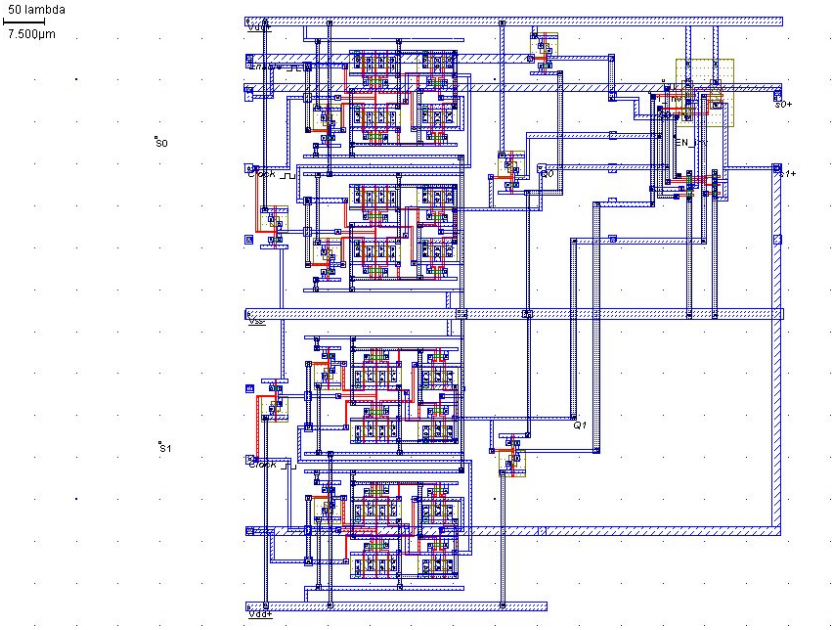
Figur 10: Tidsdiagram för simulering

A.2 Vår seriella adderare

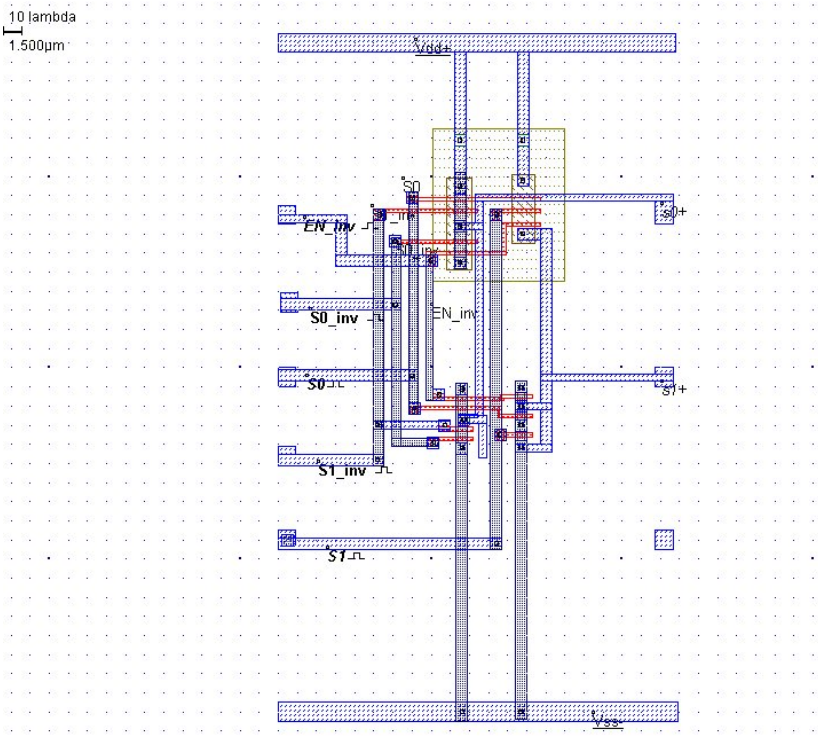


Figur 11: Vår seriella adderare

A.3 Statemaskinen

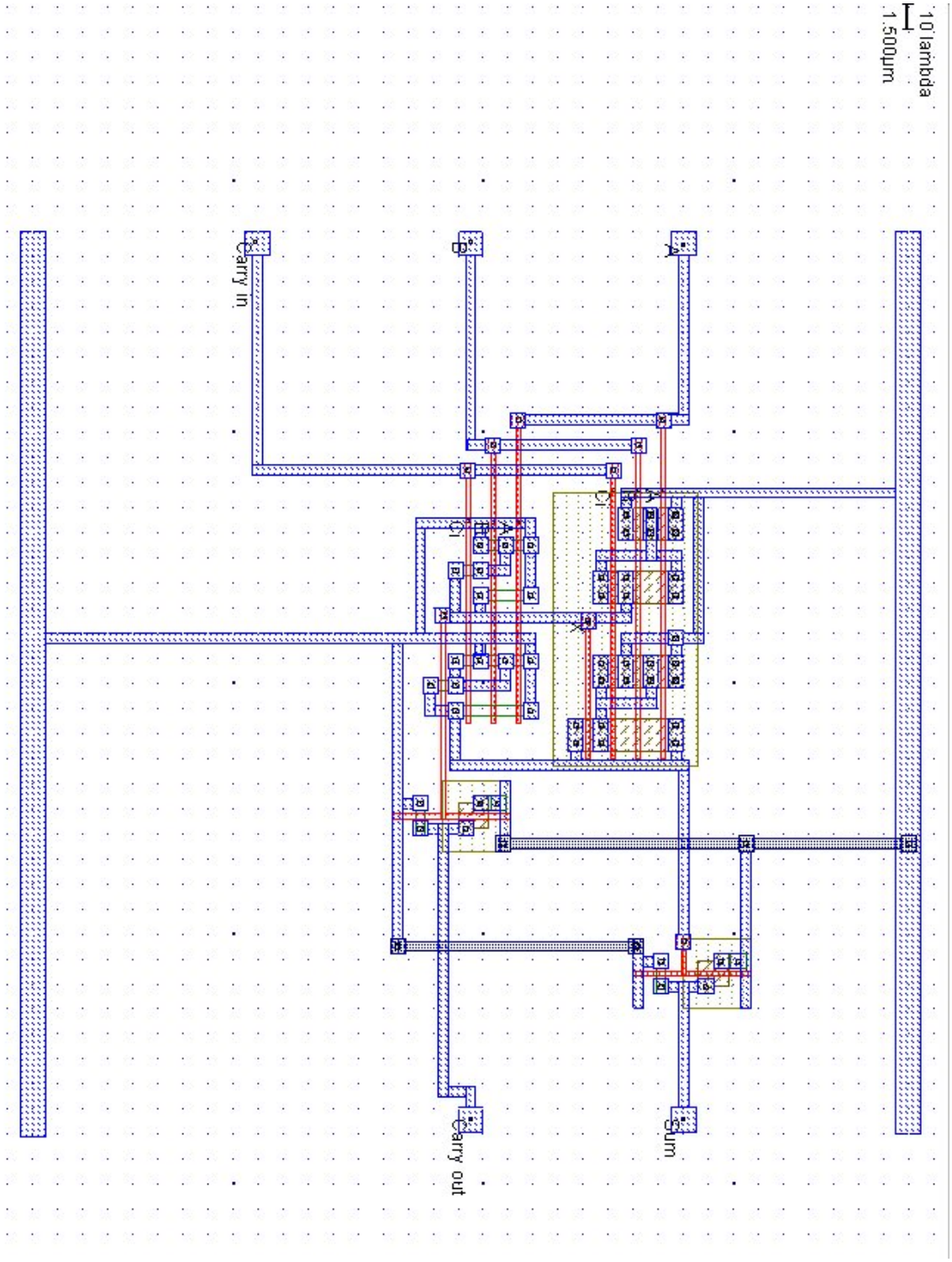


Figur 12: Hela statemaskinen inkl. register



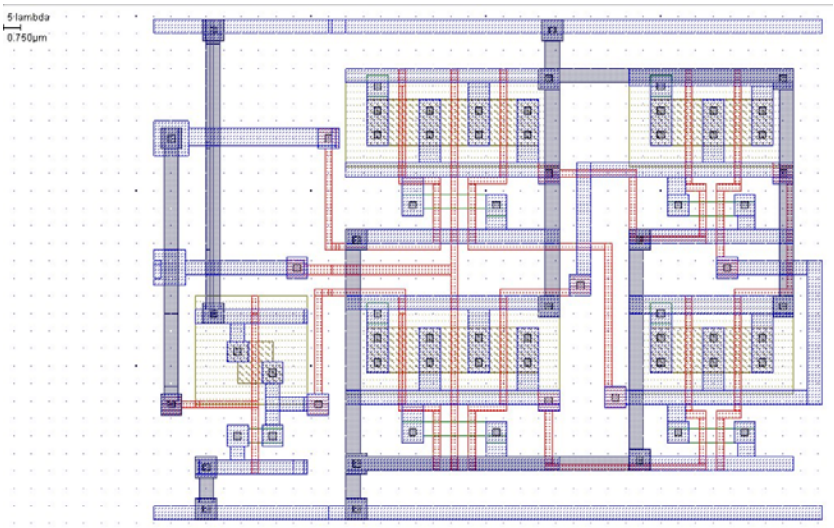
Figur 13: Statemaskinen, endast kombinatorik

A.4 Full adder

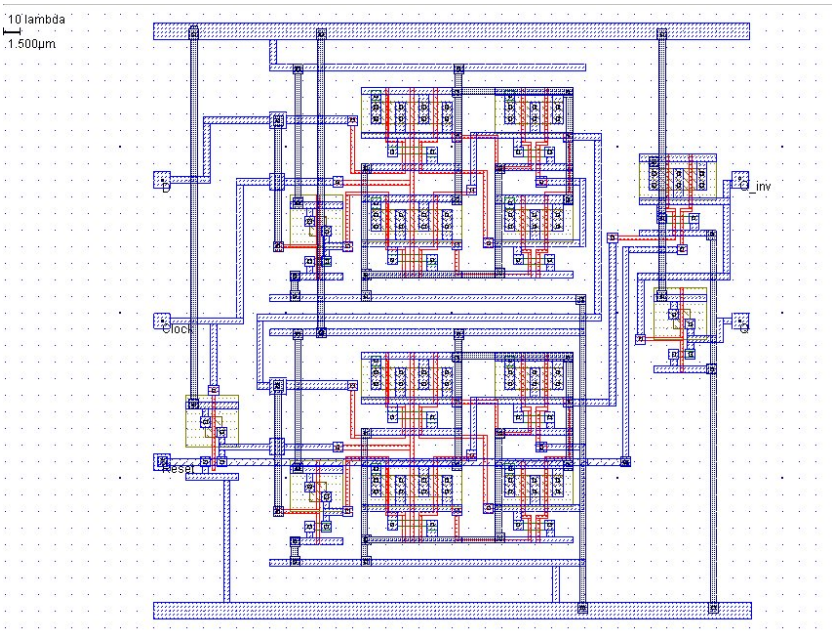


Figur 14: Vår full addder

A.5 Register



Figur 15: Del av registret, en dvippa



Figur 16: Register