LAYOUT-GENERATOR FÖR SIFFER-SERIELL TVÅPORTSADAPTOR

Examensarbete utfört i Elektroniksystem

vid Linköpings Tekniska Högskola

av

Tobias Almquist

LiTH-ISY-EX-ET-0242-2002

Linköping 2002

LAYOUT-GENERATOR FÖR SIFFER-SERIELL TVÅPORTSADAPTOR

Examensarbete utfört i Elektroniksystem

vid Linköpings Tekniska Högskola

av

Tobias Almquist

LiTH-ISY-EX-ET-0242-2002

Handledare: Oscar Gustafsson

Examinator: Mark Vesterbacka

Linköping, 5 December 2002.

Avdelning, Institution
Division, department

Department of Electrical Engineering
Elektroniksystem

Datum
Date

2002-12-19

Språk
Language

☒ Svenska/Swedish
☐ Engelska/English

☐ _____

Rapporttyp
Report: category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport

☐ _____

ISBN

ISRN LiTH-ISY-EX-ET-0242-2002

Serietitel och serienummer
Title of series, numbering

ISSN

URL för elektronisk version

http://www.ep.liu.se/exjobb/isy/2002/242

Titel
Title

Layout-generator för sifferseriell tvåportsadaptor

Layout generator for digit serial two-port adaptor

Författare
Author

Tobias Almquist

Sammanfattning
Abstract

Vid sifferseriell aritmetik används ett antal parallella bitarför varje siffra. För att jämföra prestanda och effektförbrukning i förhållande till antalet bitar behövde Institutionen för systemteknik (ISY) en layout-generator för att enkelt kunna generera layout för en sifferseriell tvåportsadaptor. Layouten skulle göras i 0.18 µm process. Antalet inkommande databitar och antalet koefficientbitar skulle vara variabelt.
Stor vikt lades vid planeringen av layouten för att genereringen av adaptorn skulle fungera smidigt oberoende av de variabla parametrarna. Kod skrevs för att koppla samman layout-instanserna och för att förenkla adaptorn.

Digit serial arithmetics uses a number of parallel bits in each digit. To compare performance and power consumption relative the number of bits, the Department of Electric Engineering (ISY) needed a layout generator to generate layout for a digit serial two-port adaptor. The layout should be done in 0.18 µm process. The number of bits of the incoming data and the number of bits of the coefficient should be variable. Great concern was put in the planning of the layout to make the generation of the adaptor work well independent of the parameters. Code was written to connect the layout instances and to simplify the adaptor.

Nyckelord
Keywords

layout, layout-generator, sifferseriell aritmetik, tvåportsadaptor

1 INTRODUKTION

1.1 Bakgrund

Institution för systemteknik, ISY, vid Linköpings Universitet forskar bl.a. kring utvärdering av prestanda i förhållande till effektförbrukning i elektroniska kretsar.

En stor del av dagens digitala elektronik utför beräkningar. Vid sifferseriell aritmetik utförs beräkningar genom att ett antal bitar ur datat används i taget. För att lätt kunna jämföra hur antalet bitar som används i beräkningen påverkar förhållandet mellan prestanda och effektförbrukning fanns önskemål om en modulgenerator som utifrån krav på antal databitar och koefficientbitar bygger upp en symmetrisk sifferseriell tvåportsadaptor för CSDC-koefficienter.

1.2 Syfte

Syftet med mitt examensarbete är att konstruera en layout-generator som med indata i form av ordlängden på insignalen och koefficienten genererar en sifferseriell tvåportsadaptor. Designen görs i 0.18 µm process och ska med hänsyn till koefficienten i den sifferseriella multiplikatorn förenklas så långt som möjligt.

1.3 Rapportens innehåll

Rapportens innehåller följande kapitel:

Kapitel 2 behandlar teorin bakom adaptorns funktion. Där beskrivs bl.a. talsystemet CSDC, sifferseriell logik, multiplikatorns funktion och uppbyggnad samt hur förenklingar i multiplikatorn utförs. Här redovisas adaptorns alla ingående kretsar.

Kapitel 3 beskriver arbetsgången under arbetet med examensarbetet. Från de första planerna för layouten till den färdiga layout-generatorn och hur problem behandlades och löstes på vägen.

Kapitel 4 är en kortfattad genomgång av de verktyg jag använt.

I kapitel 5 redovisas resultatet av arbetet. Problem tas upp, förslag till ändringar och förbättringar beskrivs.

I kapitel 6 finns avslutningsvis en förteckning över de källor som använts.

1.4 Tack

Ett stort tack till de som på något sätt hjälp mig under tiden med examensarbetet!

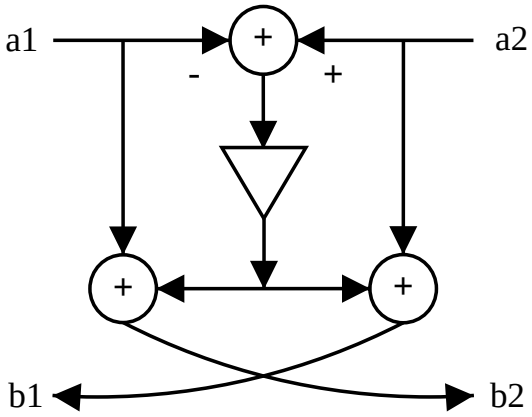
Framför allt vill jag tacka min handledare, Oscar Gustafsson, men även alla andra på ISY som genom sitt kunnande bidragit på olika sätt, däribland Emil Hjalmarson som varit till stor hjälp i arbetet med SKILL.

2 TEORI

Detta kapitel innehåller en översiktlig beskrivning av de teorier som ligger till grund för examensabetet. Eftersom arbetet främst har bestått i att göra layout och layout-generator är teorin beskriven översiktligt och tar främst upp delar som varit viktiga för utvecklandet av layout och kod i det här fallet. Jag beskriver teorin specifikt för de fall som varit aktuella under arbetets gång. Den som vill få en bredare, mer generell bild av den teoretiska bakgrunden, eller vill gå ännu djupare i bakrundsteorierna, hänvisas till källorna [5], [7], [9] och [10].

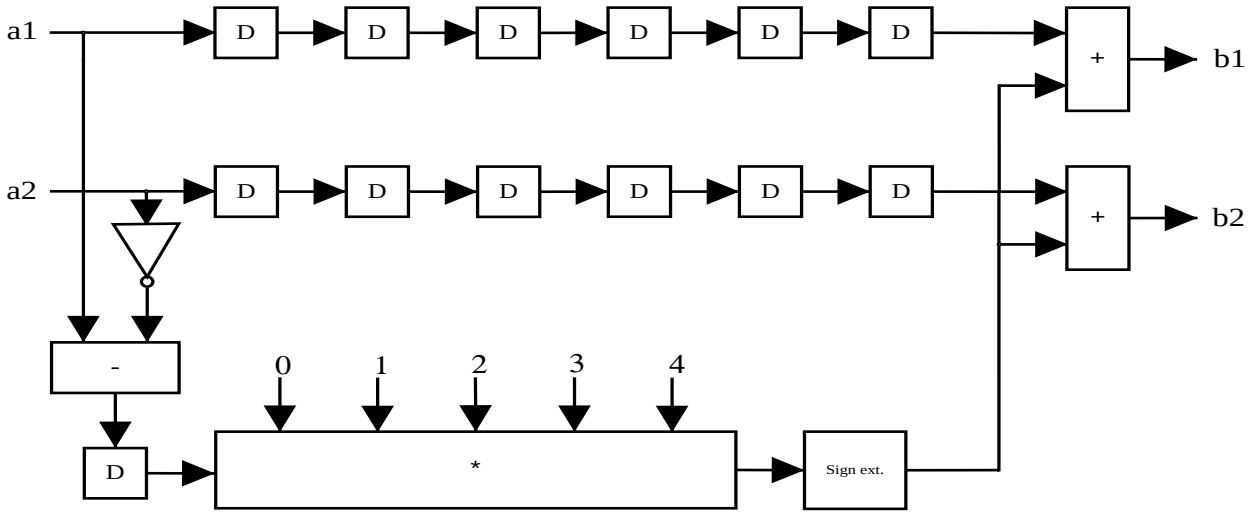
2.1 Adaptorns uppbyggnad

Adaptorn motsvarar sammanbindningar mellan olika kretselement i ett digitalt filter. Den simulerar kopplingen mellan två portar och beskriver reflektionen däremellan. Adaptorn som ska genereras här är en s.k. symmetrisk tvåportsadaptor. Dess upp-byggnad förklaras av nedanstående figur.



Figur 2.1.1 Symmetrisk tvåportsadaptor.

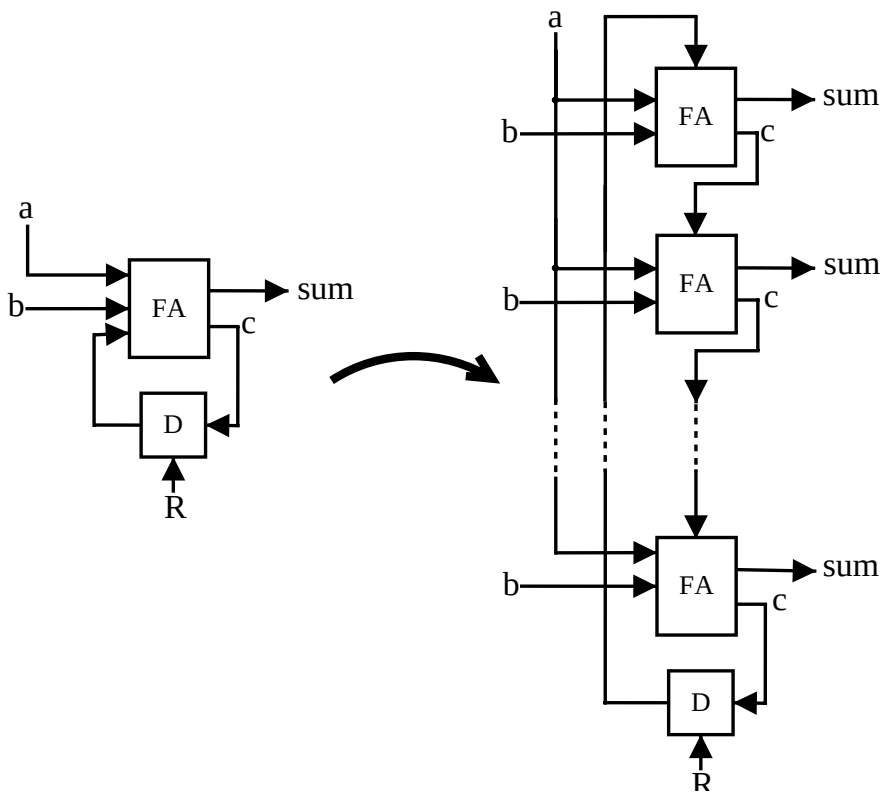
I signalflödesschemat ser vi att adaptorns huvuddelar är adderare och en multiplikator. I en mer specifik bild, figur 2.1.2, visas en adaptor där koefficienten består av fem bitar. Vi ser här mer i detalj att innehållet i adaptorn är multiplikator, heladderare och för-dröjningselement.



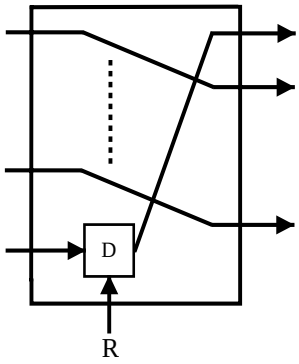
Figur 2.1.2 Bitseriell adaptor.

2.2 Sifferseriell logik

Syftet med examensarbetet är att skapa en layout-generator för en adaptor som använder sifferseriell logik. Det innebär att den ska ta indata med en siffra i taget representerad med ett antal parallella bitar. Principen blir densamma som för den bitseriella adaptor som beskrivits. Den stora förändringen jämfört med den bitseriella adaptorn blir utformningen av adderarna. Blocken i den bitseriella adaptorn överförs till sifferseriell form. Figuren 2.2.1 visar skillnaden mellan en bitseriell och en sifferseriell adderare och figur 2.2.2 ett bitparallellt skiftblock. Det bitparallella skift-blocket motsvarar den fördröjning (D-vippa) som finns efter varje heladderare (FA) vid bitseriell aritmetik.



Figur 2.2.1 Transformation av bitseriell adderare till sifferseriell adderare.



Figur 2.2.2 Sifferseriellt skiftblock.

2.3 Multiplikatorns uppbyggnad

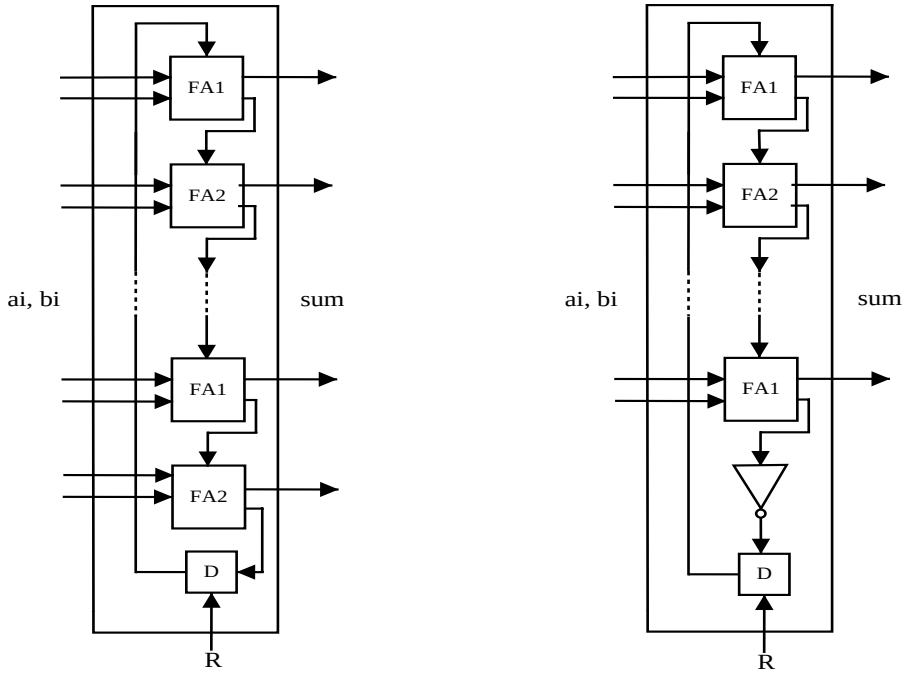
Den sifferseriella multiplikatorn är en viktig del i adaptorn och var också det som konstruerades först i exjobbet. Skillnaden mellan den bitseriella och sifferseriella adaptorn uppträder främst i multiplikatorn. Strukturen grundar sig på både bitseriell och parallell aritmetik. Antalet parallella adderare blir detsamma som antalet bitar i

indatat, d.v.s. ordlängden wd. Antalet adderare i serie blir detsamma som antalet bitar hos koefficienten.

2.4 Adderarnas uppbyggnad

Adderarna som används för den bitparallella aritmetiken i multiplikatorn och adaptorn i övrigt är av typen Ripple-Carry Adder, RCA. Indatat är tvåkomplementstal och summan av de två indata som ska adderas förs vidare som utsignal, medan minnes-siffran (carry) ges till nästföljande adderare. Minnessiffran från den sista adderaren ges efter en fördröjning till carry-ingången på den första adderaren och kan nollställas när ett nytt ord börjar matas in.

Adderarna består av två typer av heladderare som alterneras samt en nollställningsbar D-vippa. Strukturen skiljer sig hos adderarna mellan jämn ordlängd och udda ordlängd och båda visas i figur 2.4.1. Vid udda ordlängd läggs en inverterare efter sista hel-adderaren innan D-vippan.

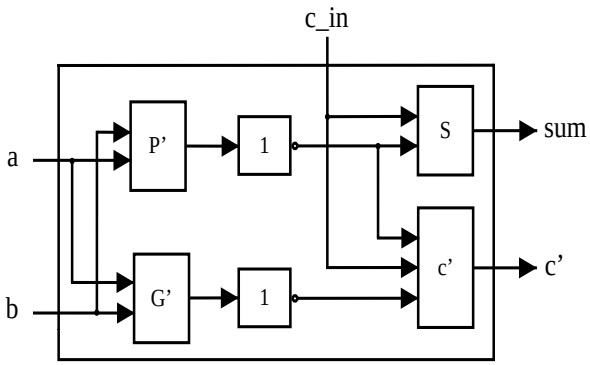


Jämn ordlängd Udda ordlängd

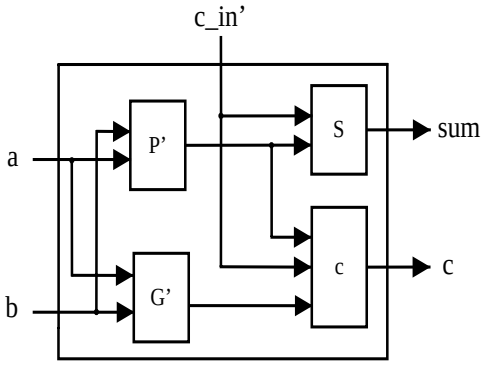
Figur 2.4.1 Adderare för jämna ordlängder. Adderare för udda ordlängder.

Gemensamt för adderarna är att de är uppbyggda av två sorters heladderare, FA1 och FA2. Heladderarna skiljer sig i uppbyggnad men har samma funktion. De tar emot a och b som insignal, ger summan, s, som utsignal och skickar carry-biten, c, från carry-

out till carry-in på nästa FA. Hur heladerarna i detalj är uppbyggda och skillnaderna mellan FA1 och FA2 beskrivs av figur 2.4.2.

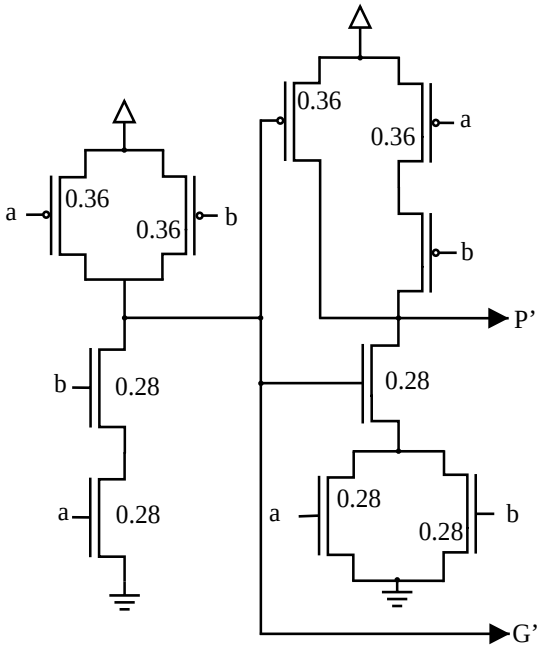


Figur 2.4.2 Byggblocken för FA1.

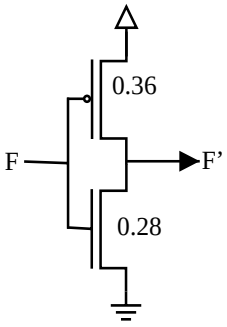


Byggblocken för FA2.

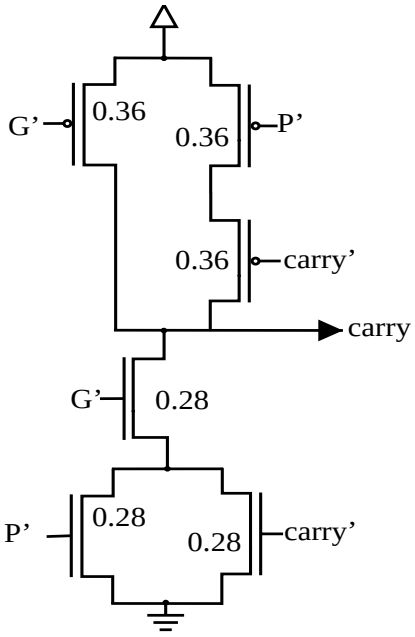
Nedanstående figurer visar kretsschema för alla de byggblock som adderarna är konstruerade av. Bredden för varje transistor anges också. Enheten är mikrometer (μm). Längden är 0.18 μm.



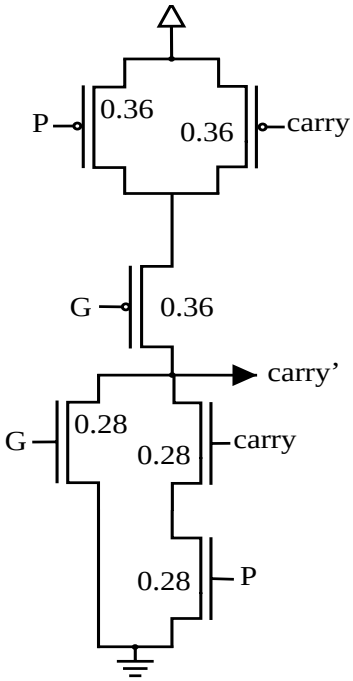
Generering/Propageringsgenerator



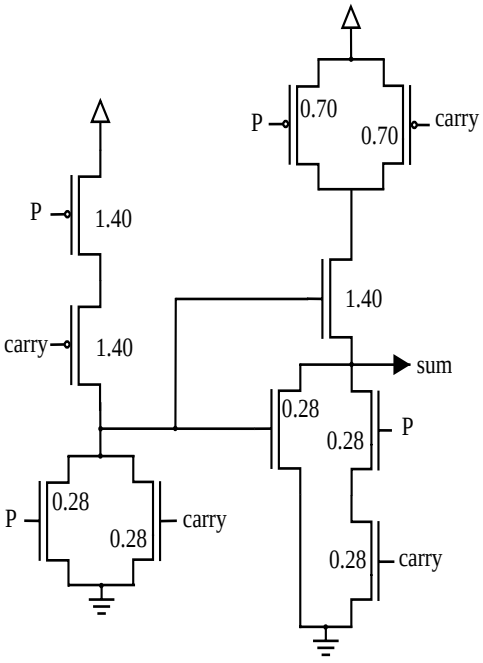
Inverterare



Carry-krets



Inverterande carry-krets



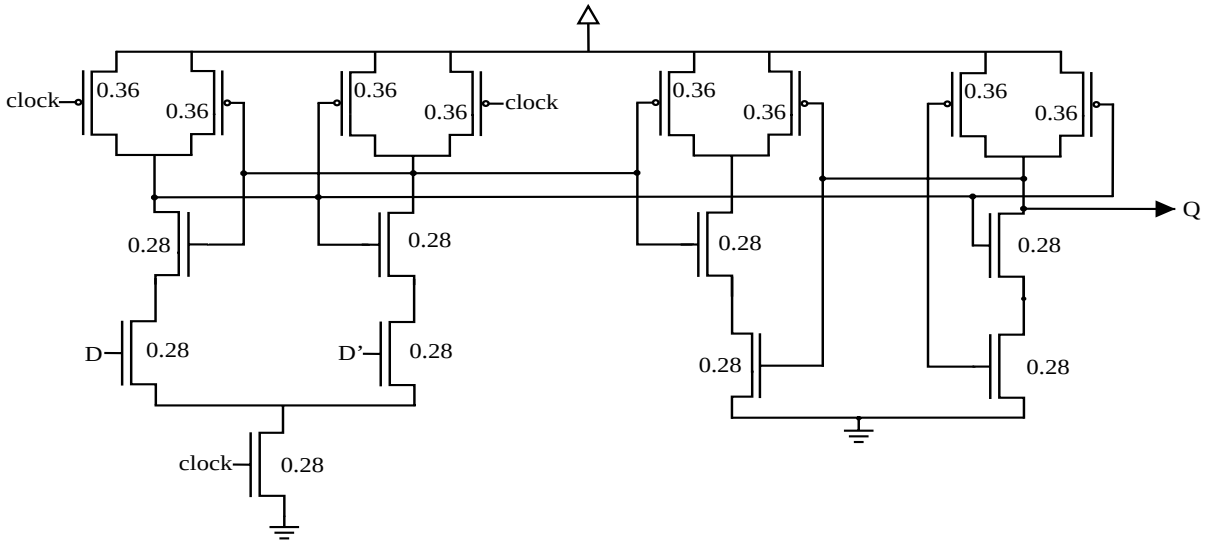
Summakrets

Figur 2.4.3 Kretsschema för adderarens byggblock.

Genom att byta ut den nollställbara D-vippan i adderaren mot en ettställbar D-vippa och sätta den till ett vid additionens början fås en subtraherare.

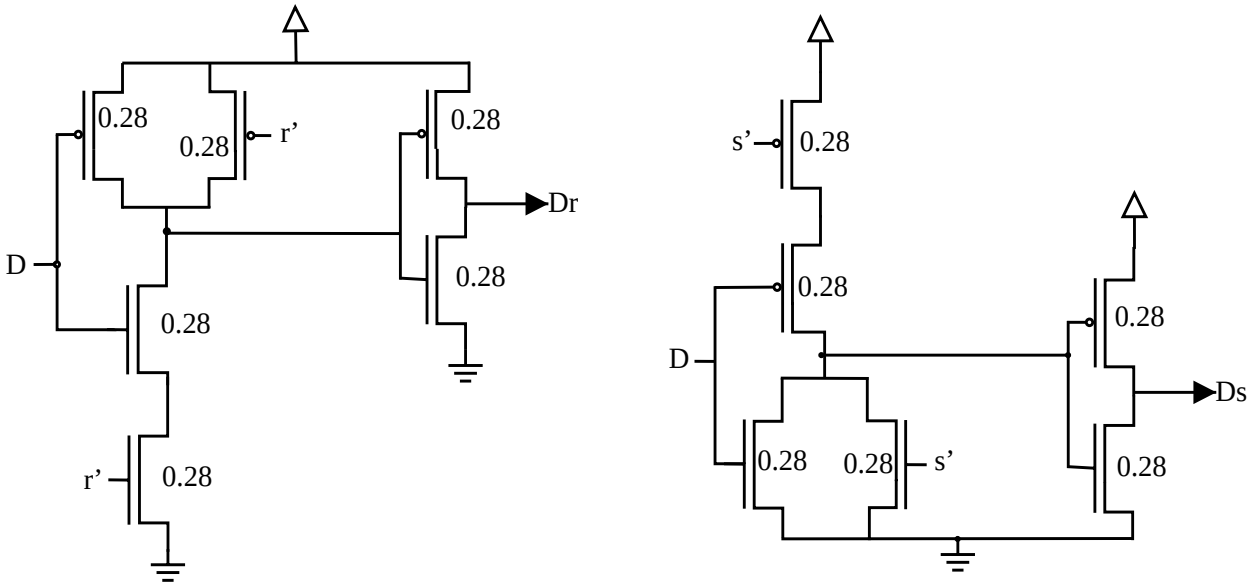
2.5 D-vippa

Den D-vippa som användes har följande kretsmässiga utseende. Den används för att fördröja insignalen en klockpuls och har ett antal fördelaktiga egenskaper [6].



Figur 2.5.1 Kretsschema för D-vippa.

För att få ettställbara och nollställbara D-vippor konstruerades två kretsar som kopplades till D-vippan. Dessa D-vippor fanns inte som färdiga kretslösningar utan fick konstrueras under arbetets gång. Figur 2.5.2 visar de kretsar som placerades före insignalen hos D-vippan i figur 2.5.1 för att göra den noll- respektive ettställbar.



Reset-del till D-vippa *Set-del till D-vippa*

Figur 2.5.2 Kretsar för ett- respektive nollställning av D-vippa.

2.6 CSDC

Adaptorns multiplikator använder fixa koefficienter av typen CSDC (Canonic Signed Digit Code). CSDC är en talrepresentation som är ett specialfall av SDC (Signed Digit Code). Till skillnad från SDC har CSDC-koden unik representation för varje tal. Det är en fördel eftersom talet kan kvantiseras utan problem.

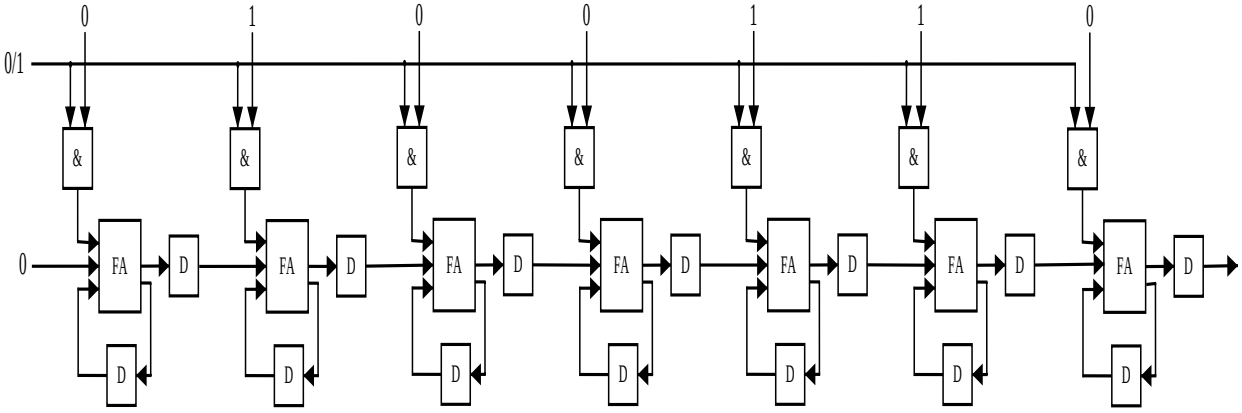
Ett tal, x, representeras i CSDC som

$$x = \sum_{i=1}^{Wd-1} x_i 2^{-i}$$

där $x_i = -1$, 0 eller 1. För denna typ av talrepresentation är det genomsnittliga antalet ettor ungefär wc/3, d.v.s. i genomsnitt är bara var tredje bit i ordet en etta, jämfört med t.ex. tvåkomplementstal där genomsnittliga antalet ettor är w/2. Ett koefficientord som består av många nollor är eftersträvansvärt i det här fallet där koefficienterna är fixa. Många nollor leder till att stora förenklingar kan göras i multiplikatorn.

2.7 Förenklingar i multiplikatorn

Eftersom coefficienten in i multiplikatorn är förutbestämd går det att förenkla multiplikatorn enligt logiska regler. Jag redovisar tillvägagångssättet med ett exempel. Exemplet är förenkling av en multiplikator med en bit indata i taget, men principen är densamma i det sifferseriella fallet. Figur 2.7.1 visar en multiplikator utan förenklingar med koefficienten 0 1 0 0 1 1 0.



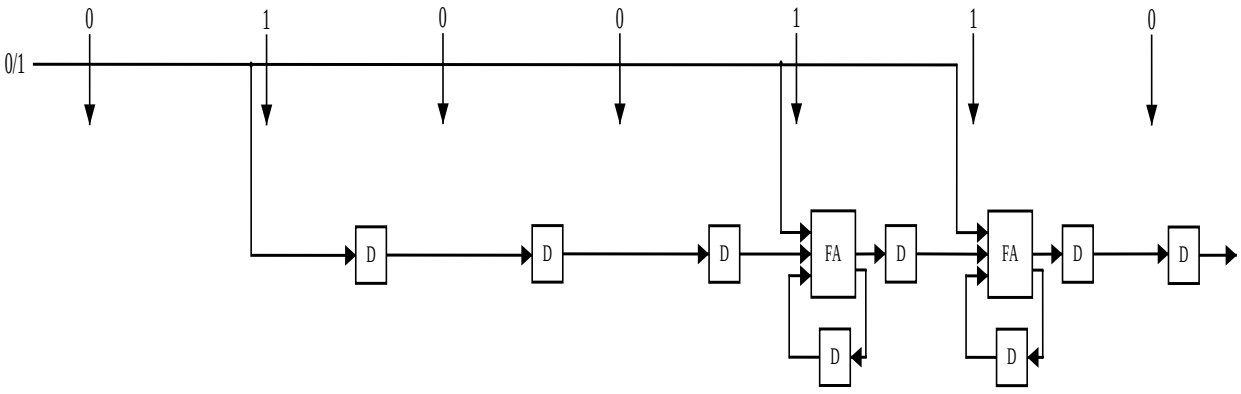
Figur 2.7.1 Multiplikator innan förenklingar utförts.

När multiplikatorn förenklas görs det genom att man undersöker vilka logiska möjligheter till utsignal som finns från varje OCH-grind och adderare.

T.ex.: Om koefficientbiten som ges som signal till OCH-grinden är en nolla kommer indatabiten till OCH-grinden inte ha någon betydelse för utsignalen. Utsignalen kommer alltid att bli noll. OCH-grinden är alltså överflödig och kan ersättas med en nolla. Med utgångspunkt från samma logiska regler [1] kan heladderaren helt tas bort i vissa fall eller ersättas med en fördröjning (D-vippa). På så sätt kontrolleras alla möjliga kombinationer av insignal tillsammans med den aktuella koefficientbiten och de förenklingar som är möjliga görs.

Tack vare de förenklingar som kan göras när alla koefficientbitar är kända blir det i praktiken endast indataordet som ges som insignal till multiplikatorn. Ledningen med inkommande koefficientbit och OCH-grinden kan alltid ersättas med en etta eller nolla in i adderaren. Adderaren kan ofta ersättas med ett fördröjningselement.

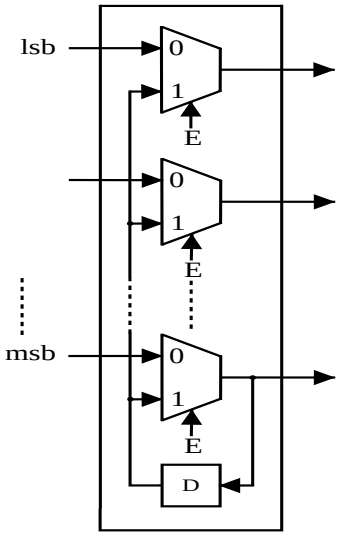
Figur 2.7.2 visar multiplikatorn i figur 2.7.1 förenklad så långt som möjligt.



Figur 2.7.2 Multiplikator efter att förenklingar utförts.

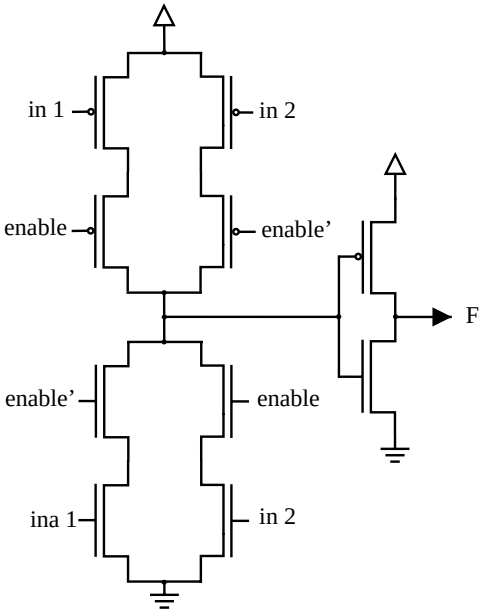
2.8 Teckenförlängning

Den teckenförlängare som ingår i adaptorn konstruerades av multiplexrar och en D-vippa enligt figur 2.8.1. Den är dynamisk och antalet multiplexar blir detsamma som wd.



Figur 2.8.1 Teckenförlängare.

Dess funktion är att släppa igenom ordet då enablesignalen är låg, och när enable-signalen är hög ge ordets mest signifikanta bit som utsignal (i praktiken kopiera den till utgången). Multiplexrarna realiserades med kretsen som visas i figur 2.8.2.



Figur 2.8.2 Kretsmässig realisering av multiplexer.

2.9 Kontrollenhet

För att adaptorn ska fungera behöver den signaler som kontrollerar de element som ordet passerar igenom. För att generera dessa kan en kontrollenhet anslutas. Kontroll-enheten byggs av en rad D-vippor som fördröjer en signal i samma takt som ordet förs genom adaptorn. Dess uppgift är att vid rätt tidpunkter sköta noll- eller ettställning av de olika element där det behövs. Set-signal till den första subtraheraren, reset-signal till multiplikatorns adderare, enable-signal till teckenförlängaren och reset-signal till de avslutande två adderarna. En klockpuls tas som insignal och skickas ut som kontroll-signal efter rätt antal D-vipppor. Elementen får signal då klockan går hög samtidigt som första biten i ett nytt ord når dem.

3 ARBETSGÅNG

Sättet som examensarbetet utförts på kan grovt delas in följande faser: Förarbete, planering, konstruktion av kretsschema, simulering, layoutdesign och programmering. Arbetsgången var till att börja med vad som brukar kallas Top-down-metodik. Adaptorn delades upp i mindre block som i sin tur delades upp ytterligare. När de mindre blocken sedan skulle konstrueras gjordes det enligt Bottom-up-metodik. Sammantaget kan alltså arbetsmetoden för exjobbet sägas vara Meet-in-the-Middle.

3.1 Förarbete

För att få förståelse för adaptorns funktion krävdes en hel del studerande. Dels av den teori som ligger bakom funktionen, som har beskrivits i kapitel 2, men också av andra liknande konstruktioner. Ändamålet var att lära av andras misstag och bygga vidare på deras framsteg.

Tidigare exjobb hade utförts inom området. Generator och layout av multiplikator hade gjorts som separata exjobb, [5], [7], [8], [10], men aldrig en hel adaptor från grunden med tillhörande layout-generator. Motsvarande arbete hade heller aldrig utförts i 0.18 μm process.

I den här fasen bestämdes vilka kretsar som skulle användas för att bygga heladderare, D-vippor m.m. De kretsar som valdes är de som har beskrivits i kapitel 2.

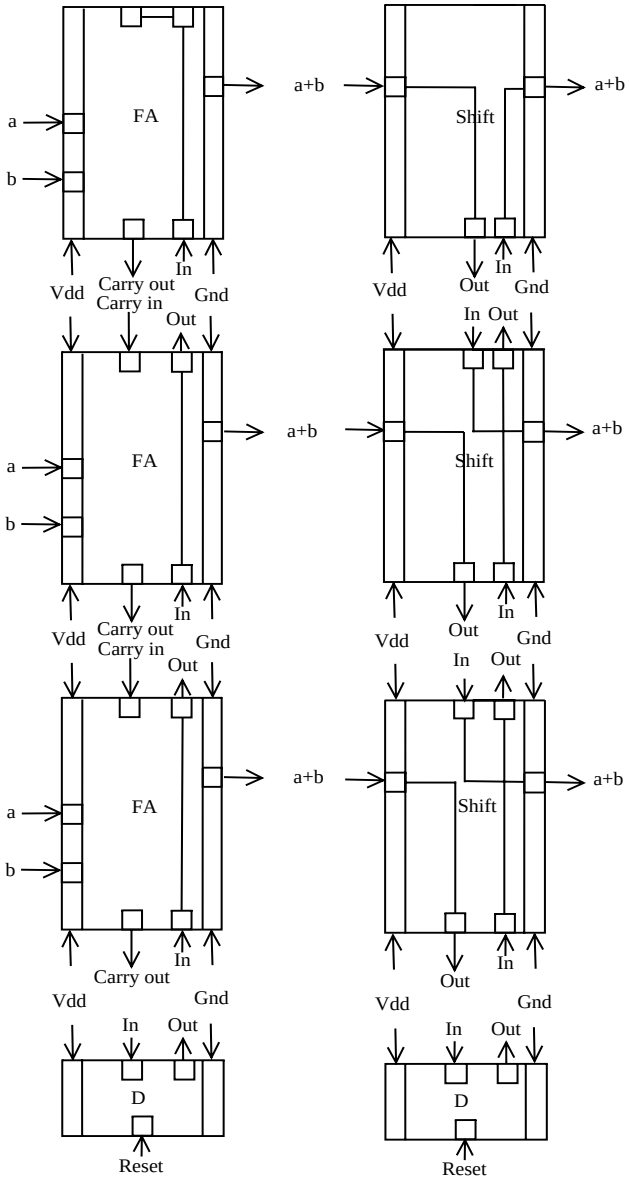
Mitt exjobb består dock i första hand av att göra layout och layout-generator för adaptorn, så en stor del av förarbetet koncentrerades på hur layout-tekniska problem skulle lösas. Även här gick arbetet ut på att studera andra liknande projekt och ta vara på de bra idéerna och utveckla de dåliga. Tid behövdes också för att sätta sig in i det kraftfulla, men också komplexa, Cadence-paketet som skulle användas som verktyg vid konstruktionen.

3.2 Planering

Efter att ha kommit fram till hur adaptorn, och då främst den sifferseriella multiplikatorn fungerar samt vilka block som skulle ingå i layouten gjordes en grov "skiss" över hur blocken skulle kunna se ut. Denna planering var nödvändig för att underlätta vid layout-arbetet och framförallt vid programmeringen.

Den största och mest komplexa delen i adaptorn är multiplikatorn, därför var det den första del som konstruerades och de andra delarna designades sedan med avseende på hur multiplikatorn utvecklades.

Med hänsyn till vilka block som skulle kunna kopplas samman planerades var ut- och ingångar samt vdd och jord skulle placeras. Det som utgör byggblocken i multiplikatorn är adderare, skiftblock och D-vippor. Efter några olika förslag på strukturen hos dessa valdes strukturen i figur 3.2.1 som visar grunden för hur layouten skulle utformas. Skiftblocken byggs upp av SKILL-koden.



Figur 3.2.1 Skiss för design av multiplikatorns adderare och D-vippor.

Multiplikatorn är den del i adaptorn som förändras mest vid olika antal bitar och koefficienter. Den kommer också att förenklas beroende på koefficienten. Den utgör i de flesta fall den största delen av adaptorn.

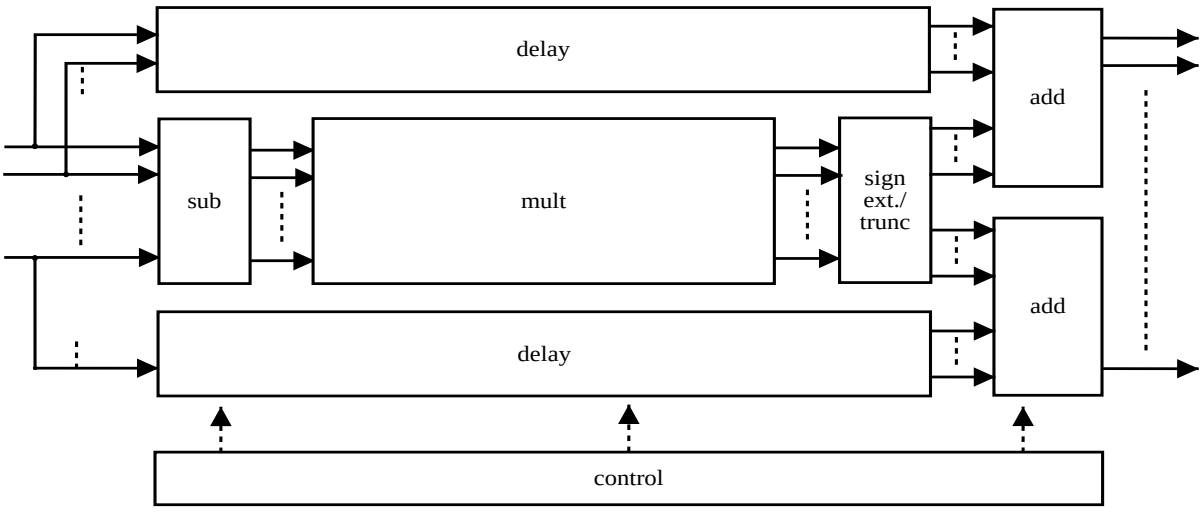
För att slippa onödig ledningsdragning och för att kunna koppla samman blocken på flera sätt efter förenklingen blev lösningen att ha ledningarna för matningsspänning och jord vertikalt, insignalerna a och b till vänster, summautgången s till höger och carry in/out i övre respektive nedre delen av heladderaren. Skiftblocken och de nollställningsbara D-vipporna anpassades sedan efter detta.

Ledningar för inkommande koefficientbit kommer i praktiken alltid att kunna förenklas bort och ersättas med en etta eller nolla som insignal in i adderaren, d.v.s. dras direkt till matningsspänningsledning eller jord.

Eftersom adaptorn är både generell och specifik krävdes en hel del planering för att undvika onödigt komplicerad lösning. Adaptorn är generell så till vida att antalet bitar i insignalen kan sättas godtyckligt av användaren. Även koefficienten sätts av användaren utan krav på antal bitar. Adaptorn blir dock specifik på så sätt att när koefficienten är given ska multiplikatorn förenklas med avseende på den.

En skiss över layouten för hur hela adaptorn skulle se ut gjordes också och tanken var att oavsett storlek och förenklingar som skulle bli följden vid olika fall skulle konstruktionen hållas kompakt. Målet var att förhållandet i storlek mellan alla block ska behållas så att inga tomrum uppstår när antalet bitar ändras.

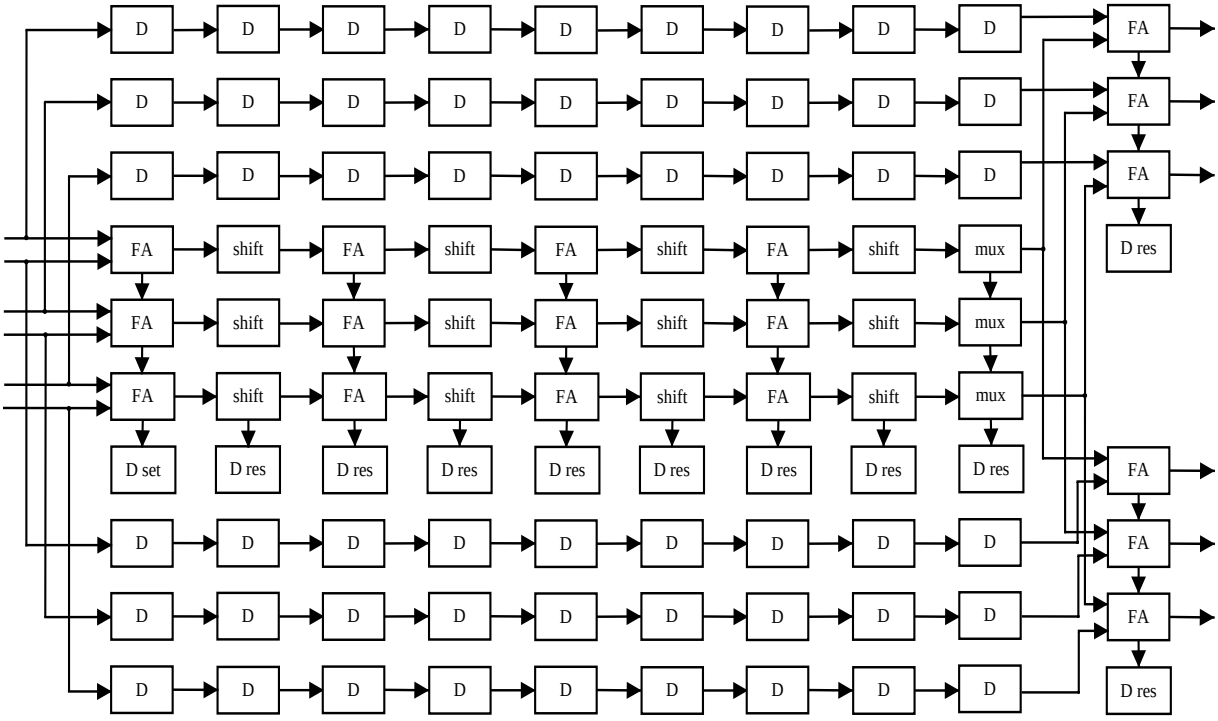
Den färdiga adaptorn är uppdelad i de block som ses i figur 3.2.2. Figuren visar också hur blocken placeras i förhållande till varandra.



Figur 3.2.2 Adaptorns layoutmässiga uppbyggnad.

En mer detaljerad beskrivning för ett specifikt fall arbetades också fram. Den visar innehållet i alla block utom kontrollenheten och visar även hur signalerna går genom

kretsen. Figur 3.2.3 nedan visar en adaptor utan förenklingar för indata med tre bitars ordlängd och en trebitars koefficient. Skissen användes som grund vid layout och programmering eftersom strukturen i princip inte kommer att ändras för större antal koefficientbitar eller fler bitar i indataordet.



Figur 3.2.3 Översiktlig bild av adaptor med tre bitars ordlängd och tre koefficienter.

3.3 Konstruktion av kretsschema

Innan arbetet med att göra layout började ritades kretsschema för alla kretsar som skulle ingå. Detta för att kontrollera funktionen och andra viktiga egenskaper och för att bredderna på transistorerna skulle kunna simuleras fram. De bredder som bestämdes efter simulering visas i de figurer av kretsarna som finns i kapitel 2.

Till att börja med gjordes grundläggande generella kretsar, -inverterare, och/eller-grindar osv. Av dessa konstruerades sedan summakretsar, carry-kretsar m.m. Slutligen sattes de ihop på den översta nivån, d.v.s. de block (heladerare, D-vippor o.s.v.) som layoutgeneratorn ska använda för att bygga upp adaptorn.

En symbol skapades för varje krets för att de enkelt skulle kunna användas i andra större kretsar.

3.4 Simulering

De olika kretsarna som kretsschema konstruerades för i ovanstående fas simulerades kontinuerligt, samtidigt som olika bredder på transistorerna fastställdes.
Bredderna på transistorerna fastställdes genom att svepa bredden över ett rimligt intervall, och förfina det efterhand, samtidigt som man kontrollerar när utsignalen från kretsen uppfyller krav på tillräcklig utspänning och stigtid.

En variant av varje symbol gjordes också för att användas vid simulering. Till dem kopplades pulser till ingångarna i form av spänningskällor och en last till utgången i form av en kondensator.

Vid simulering användes som insignaler en puls med stig- och falltid satt till 100 ps, 3.3 V. Som last användes en 20 fF kondensator.

Bl.a. för att underlätta vid design av layouten utfördes matchning mellan transistorerna där det var lämpligt på så sätt att närliggande transistorer gavs samma bredd.

De krav på kretsarna som kontrollerades vid simuleringen var i första hand krav på funktion och prestanda i form av snabbhet och utspänningsnivåer, d.v.s. att de klarade av att ge utsignaler med i princip samma tidsprestanda som de insignaler som satts och en spänningsnivå lika med 3.3 V. Kretsarna simulerades och modifierades efter hand med utgångspunkt från simuleringarna för att klara kraven, dels som enskilda kretsar men också när de kopplats samman till de större delblock som adaptorn består av.

3.5 Layout

När layouten utformades användes de skisser som inledningsvis planerades i kombination med de kretsscheman som gjorts som låg till grund för utformningen. I stort visade de sig fungerade bra, men på några punkter fick de justeras och förfinas efter hand. Skisserna var gjorda för att ge en översiktlig bild av strukturen. Nu bestämdes exakta mått och placeringar för utgångar, ingångar o.s.v. Hur nära transistorer och ledningar kan placeras påverkade också till viss del utformningen av layouten.

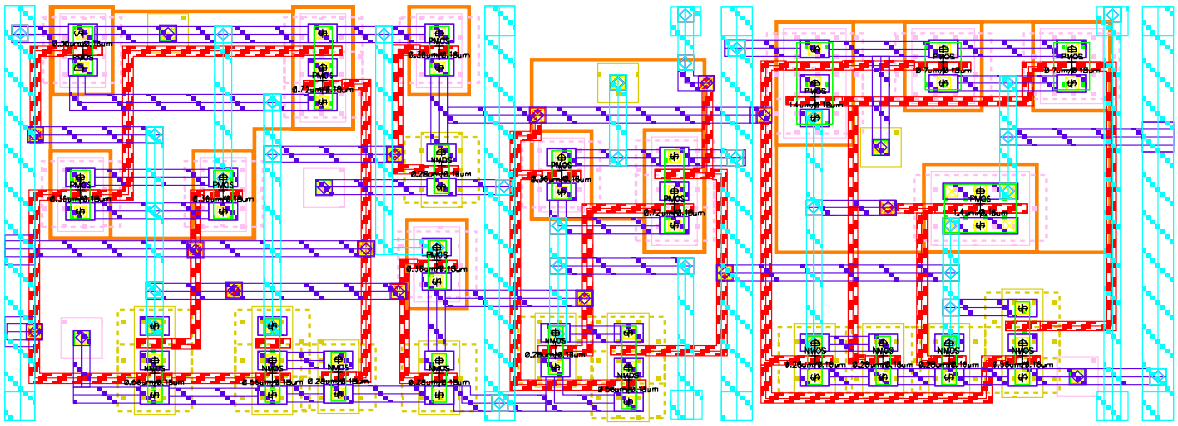
Motsvarande layout gjordes för varje krets som konstruerats och vissa av dessa block sattes ihop till större instanser. När layouten designades var målet dels att få en så kompakt layout som möjligt, men framför allt togs hänsyn till att delblocken skulle ingå i den större krets som skulle genereras och därför skulle de också vara smidiga att sätta samman på mer än ett sätt. T.ex. placerades ut- och ingångar från delblock så att ledningsdragning mellan blocken skulle undvikas i möjligaste mån, d.v.s. när blocken

placeras bredvid varandra ska de ledningar som ska kopplas samman vara placerade så att de automatiskt ansluts och onödig ledningsdragning och kod kan undvikas.

Tre ledningslager användes. "Metall 1" vertikalt, "Metall 2" horisontellt, och "Poly" i första hand mellan gate på transistorerna.

Att minimera en delkrets leder inte automatiskt till att hela kretsen blir liten. En standardhöjd sattes för att sammankopplingen av blocken skulle gå smidigt. Höjden fick utgå från de större blocken. Eftersom bredden inte var förutbestämd kunde de block som kräver mindre yta ändå göras relativt små genom att deras bredd minskades.

Figur 3.5.1 nedan visar layouten för en fulladderare av typ 1 (FA1). Heladderarna är de layoutinstanserna som visade sig kräva störst chipyta. Därför satte de måtten för D-vipporna som ska anslutas till dem. Höjden för varje instans sattes till 8.40 µm. Tanken var att kunna ansluta jord och spänningsledningar mellan de olika blocken utan extra ledningar och kontakter och att alternera jord och spänningsledningarna för att enkelt kunna ansluta en "gaffel" till dem när hela layouten är klar. För att klara spänningsförsörjningen har blocken två vertikala spänningsledningar och motsvarande två jordledningar. Placering av dessa är också bestämt för att alla block ska kunna kopplas samman på höjden.



Figur 3.5.1 Layout för heladderare typ 1 (FA1).

Varje in- och utgång från instanserna har en kontakt. Detta för att blocken ska kunna användas även i andra konstellationer. I detta fall behöver dock inte ledning dras mellan kontakterna eftersom in- och utgångar från blocken har placerats så att de kommer på samma höjd och automatiskt ansluts till en ledning när instanserna läggs intill varandra.

Eftersom adaptorn ska vara generell och dessutom ska kunna förenklas kunde inte allt för stora enheter sättas samman för hand. De kunde inte heller byggas så specifika som annars hade varit möjligt i avseende på kompakthet.

Hänsyn togs alltså i första hand till att den färdiga layouten skulle vara kompakt, mer än att alla delblock skulle vara det, eftersom den färdiga adaptorn skulle vara generell och teoretiskt kan bli hur stor som helst.

3.6 Programmering

Efter att alla delblock och enheter var gjorda skrevs kod i programspråket SKILL. Koden skrevs för att koppla samman alla instanser och förenkla multiplikatorn.

Indatats antal bitar och koefficienten matas in av användaren. Utifrån koefficienten förenklas multiplikatorn genom att adderare kan tas bort och ersättas med D-vippor. Genom sättet som förenklingarna görs på behövdes inga insignaler i form av koefficientbitar i den färdiga kretsen. Detta gör generatorn mer specifik. De förenklingar som kan göras i adaptorn utförs med hjälp av villkorssatser i SKILL-koden. Antalet bitar i indataordet, d.v.s. ordlängden (wd), tas som indata tillsammans med CSDC-koefficienten.

Kodningen blev en viktig del av exjobbet och här följer en beskrivning av principen för hur koden bygger upp adaptorn.

Programmet bygger adaptorn i kolumner som sedan sätts ihop till en helhet. Programmet går igenom koefficientbitarna en efter en där antalet avgör hur många kolumner som adaptorn kommer bestå av. Programmet kontrollerar om koefficientbiten är en nolla eller etta och utifrån vissa villkor förenklas kolumnen. Antalet bitar i ordet, wd, avgör hur många rader varje kolumn består av. Med rader menas här en färdig layoutenhet, t.ex. en heladderare, och med kolumn avses bredden av en eller flera rader sammankopplade på höjden.

4 VERKTYG

De verktyg som användes under arbetets gång för testning, simulering, layout och kodning ingår alla i Cadence-paketet Custom IC Design Tools, Front to Back Design Environment 4.4.5.42. Här tänker jag redogöra för vilka verktyg jag använt, de viktigaste funktionerna och på vilket sätt jag utnyttjat dem.

4.1 Composer-Schematic

Cadence Composer-Schematic användes för att göra kretsschema till de kretsar (inverterare, OCH/ELLER-grindar m.fl.) som skulle bygga upp de större kretsarna (FA1 & FA2, D-vippor, MUX o.s.v.). Här placeras symboler för transistorer ut och kopplas samman. En kontroll görs för att kontrollera att alla ledningar och instanser är anslutna, men ingen hänsyn tas till om layouten är rimlig i fråga om avstånd och ledningsdragning.

4.2 Composer-Symbol

När varje krets var konstruerad användes Cadence Composer-Symbol för att göra en symbol för kretsen som kunde användas när större kretsar skulle kopplas samman. I Composer-Symbol görs en symbol med ingångar, utgångar och anslutningar för jord och vdd för ett kretsschema.

4.3 Analog Environment

När kretsarna simulerades användes verktyget Analog Environment. Där görs en specifik setup för varje krets, med inställningar av vilken form av simulering som ska göras, i det här fallet oftast transientanalys, vilka signaler som ska visas, och simuleringstiden. När bredder på transistorerna skulle simuleras fram användes funktionen “Parametric Analys”. Där sveps en variabel, här transistorbredden, över ett intervall i ett antal steg och utspänningen plottas.

4.4 Virtuoso

Cadence Virtuoso är verktyget som används för att göra layout. Transistorer, kontakter och olika ledningslager placeras ut för hand och layouten byggs upp grafiskt. För att kontrollera att layouten uppfyller designkrav finns funktionen DRC, Design Rule Check. Med det kommandot kontrolleras bl.a. att transistorer och ledningar av samma typ och inte placerats för nära varandra.

I Cadence Virtuoso gjordes layouter för de delblock som simulerats. Varje layout

sparades som mindre block och som ihopsatta enheter för att kunna anropas av layout-gerneratorn.

4.5 SKILL

Den kodning som utförts har här gjorts i programspråket SKILL. SKILL är en del i Cadence-paketet och används för att med hjälp av kod skapa layout [2]. Detta är praktiskt vid generering av större layouter uppbyggda av ett antal instanser. I mitt fall var kodning av den här typen nödvändig eftersom adaptorn skulle vara generell och kunna ändras utifrån användarens önskemål. SKILL fungerar utmärkt när man vill skapa en layout-generator.

SKILL har funktioner för att placera ut, flytta och koppla samman layout-enheter. Programspråket har likheter med många andra programmeringsspråk, t.ex. C, i form av for, while och if-satser, vilket gör det användbart när man snabbt vill bygga stora kretsar under vissa villkor.

Koden kan skrivas i en texteditor, t.ex. Emacs, och anropas sedan via ett kommando-fönster i Cadence.

5 RESULTAT OCH AVSLUTNING

5.1 Resultat

Resultatet av examensarbetet är en layout-generator för en symmetrisk sifferseriell tvåportsadaptor. Layouten förenklas med avseende på koefficienten. Sättet som generatorn är byggd på leder inte till att layout-arean minskar vid förenkling, men antalet transistorer och ledningar minskas avsevärt vilket leder till minskad effektförbrukning och förbättrade prestanda. Layouten behåller i princip förhållandet mellan bredd och längd vid olika antal databitar och koefficienter.

En av fördelarna med denna form av layout-generering är att layouten ska kunna göras teoretiskt obegränsat stor. Vid generering av mycket stora layouter kräver detta att hänsyn tas till drivningsförmåga vid långa ledningar. P.g.a. av bl.a. examensarbetets begränsande tidsramar fick dock detta åsidosättas eftersom det i praktiken sällan finns någon anledning till att sätta krav på att layouter ska kunna bli obegränsat stora. Inom de användningområden generatorn förväntas användas kommer detta troligen inte vara något problem.

Generatorn är konstruerad så att användaren anger antalet bitar i inkommande siffra, wd, och CSDC-koefficienten på formen *layout(wd list(α1, α2, α3...))* för att generera layout, t.ex: *layout(8 list(1 0 1 1 0 1))*.

5.2 Problem

De problem som uppstod berodde oftast på okunskap om hur programmet arbetar. Cadence är ett kraftfullt hjälpmedel vid konstruktion, utveckling och simulering av elektroniska kretsar. Att kunna utnyttja alla funktioner kräver dock viss erfarenhet, vilket det inte fanns tid för att skaffa under exjobbets gång. Hade större kunskap funnits om programmets funktioner hade troligen mer tid kunnat läggas på att förfina detaljer i layouten.

Arbetet inriktades efter hand mer och mer på att utforma layout och skriva kod. Därför fick detaljförståelsen för kretsarnas funktion ibland sättas i andra hand, och redan befintliga lösningar fick accepteras.

5.3 Förbättringsmöjligheter

Att hitta fungerande optimala layout-lösningar tar ofta lång tid. P.g.a. brist på tid, okunskap om programvara och liten erfarenhet av layout- och kodningsarbete finns vissa punkter som kan förbättras.

En kontrollenhet måste anslutas. Layouten är förberedd för detta. Ledningar kan dras i "Metal-3"-lagret och alla enheter som behöver kontrollsignaler har ingångar med namnet "control" för detta.

Vill man jobba vidare med adaptorns layout finns möjligheter att hitta mer optimala lösningar i fråga om kompakt layout. Med större kunskap om programmeringsspråket SKILL går det säkert även där att finna mer kompakta lösningar.

Fler och mer ingående simuleringar skulle också kunna leda till att göra kretsen bättre. Man skulle på detta sätt kunna lägga större vikt vid t.ex. störningskänslighet än vad som gjorts.

Layoutgeneratorn skulle också kunna utökas till att kunna hantera varierande koefficient, men då kompliceras förenklingen av multiplikatorn.

5.4 Slutsats

Modulgeneratorn som konstruerats under detta examensarbete bygger en symmetrisk tvåportsadaptor för CSDC-koefficienter och med variabelt antal parallella indatabitar.

Att generera layouter med layoutgeneratorer är fördelaktigt när man snabbt vill bygga layouter för t.ex. simulering. Att konstruera generatorn kräver att man tar hänsyn till alla de fall användaren vill kunna utnyttja.

6 REFERENSER

[1] Hemert, Lars-Hugo (1992), *Digitala kretsar.* Andra upplagan. Studentlittertur, Lund. ISBN 91-44-00099-5

[2] Hjalmarson, Emil (2001), *Introducion to Skill.* Linköping.

[3] Johns, David A. & Martin, Ken (1997), *Analog Integrated Circuits Design.* John, Wiley & Sons Inc., Canada. ISBN 0-471-14448-7

[4] Kang, Sung-Mo & Leblebici, Yusuf (1999), *CMOS Digital Integrated Circuits Analysis and Design.* Andra upplagan. Mc Graw Hill, Boston. ISBN 0-07-292507-8

[5] Lundkvist, Jonas (1999), *VHDL-generator för sifferseriell två-portsadaptor.* Examensarbete vid ISY, Linköpings Universitet, Linköping. Reg. nr.: LiTH-ISY-EX-ET-0178

[6] Oelman, Bengt & Xue Shang (2001), *Comparative studie of low-voltage performance of standard-cell flip-flops.* Department of Information Technology and Media, Mid-Sweden University. Sundsvall

[7] Olofsson, Johannes (2001), *Modellering av interpolatorfilter i VHDL.* Examensar bete vid ISY, Linköpings Universitet, Linköping. Reg. nr.: LiTH-ISY-EX-3110

[8] Ramström, Markus (1999), *Modulgenerator för sifferseriella allpasslänkar.* Examensarbete vid ISY, Linköpings Universitet, Linköping. Reg. nr.: LiTH-ISY-EX-ET-0153

[9] Wanhammar, Lars (1999), *DSP Integrated Circuits.* Academic Press, San Diego. ISBN: 0-12-734530-2

[10] Zervos, Gerasimos (1999), *Sifferseriell multiplikator förenklad för CSDC-koefficienter.* Examensarbete vid ISY, Linköpings Universitet, Linköping. Reg. nr.: LiTH-ISY-EX-ET-0167