School of Mathematics and Systems Engineering

Reports from MSI - Rapporter från MSI

AM Modulator/Demodulator i VHDL

AM Modulator/Demodulator in VHDL

Henrik Johannesson

Juni
2001

MSI
Växjö University
SE-351 95 VÄXJÖ

Report 01051
ISSN 1650-2647
ISRN VXU/MSI/EL/E/--01051/--SE

Organisation/ Organization
VÄXJÖ UNIVERSITET
Matematiska och systemtekniska institutionen
Växjö University/
School of Mathematics and Systems Engineering

Författare/Author(s)
Henrik Johannesson

Dokumenttyp/Type of document
Examensarbete/ Diplomawork

Titel och undertitel/Title and subtitle

AM Modulator/Demodulator i VHDL

AM Modulator/Demodulator in VHDL

Sammanfattning (på svenska)

VHDL har snabbt blivit ett populärt hårdvarubeskrivande programmeringsspråk. Detta examensarbete ger ett förslag på hur det är möjligt att ersätta en analog AM modulator/de-modulator med en digital lösning. Förslaget som beskrivs är programmerat i VHDL och ska implementeras i en FPGA.

Resultatet som visar att det är möjligt att göra en digital AM modulator/demodulator med möjlighet till flexibel datahastighet och modulationsindex är baserade på simuleringsresultat.

Nyckelord

VHDL, testbänk, amplitudmodulering

Abstract (in English)

VHDL has quickly become a popular hardwaredescription programming language. This thesis work gives a proposal about how it is possible to replace an analogue AM modulator/demodulator with a digital solution. The proposal that is described is programmed in VHDL and will be implanted in a FPGA.

The result that shows it is possible to do a digital AM modulator/demodulator with opportunity to flexible datarate and modulationindex is based on simulatingsresult.

Key Words

VHDL, testbench, amplitudemodulation

Utgivningsår/Year of issue	Språk/Language	Antal sidor/Number of pages
2001	Svenska/ Swedish	41

Internet/WWW http://www.msi.vxu.se

INNEHÅLLSFÖRTECKNING

BILAGOR 21

1 Inledning

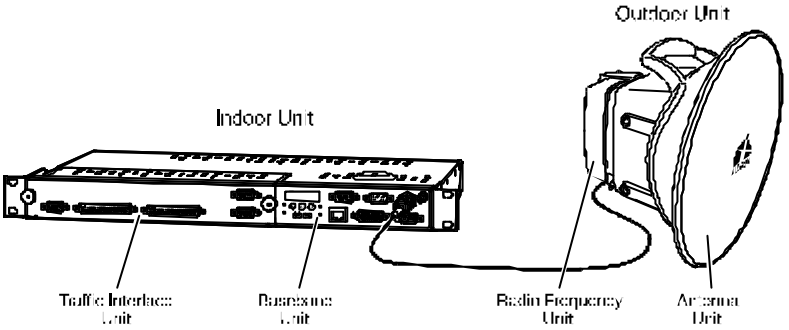
Som avslutning på ingenjörsutbildningar ingår det att göra ett examensarbete på ett externt företag. Jag utförde mitt arbete på Allgon Microwave AB i Göteborg. Uppgiften bestod i att göra om en befintlig analog AM modulator/demodulator till en digital konstruktion. Arbetet motsvarade 10 högskolepoäng.

1.1 Bakgrund

Allgon Microwave AB ingår som ett affärsområde i en större koncern. Hela koncernen har produkter såsom: antennsystem, combiners, repeaternät, mobiltelefonantenner, radiolänkutrustning och blåtandprodukter.

Allgon Microwave AB utvecklar, tillverkar och marknadsför mikrovågslänkar. Med radiolänk avses trådlös överföring där informationen överförs via en kedja av radiosändare och mottagare med riktade antenner vid frekvenser över 3 GHz. Överföringskapaciteten för Allgon Microwave AB:s radiolänkprogram är upp till 36 Mbit/s.

Figur 1 visar hur Allgon Microwave AB:s mikrovågslänk ser ut. Länken består av en inomhusenhet, en utomhusenhet och en antenn.



Figur 1. Mikrovågslänk

För kommunikation mellan inomhusenhetens och utomhusenhetens processorer används idag en analog AM modulator/demodulator. För att få en mer flexibel AM modulator/demodulator samt för att utnyttja överkapaciteten i befintliga FPGA-kretsar (**F**ield **P**rogrammable **G**ate **A**rray) kan en lösning vara att göra en digital AM modulator/demodulator.

1.2 Syfte

Syftet med arbetet är att hitta en implementering av AM modulator/demodulator med hjälp av en FPGA, A/D- och D/A-omvandlare. Den nya lösningen skall vara flexibel så till vida att modulationsindex och datahastighet ska kunna varieras.

1.3 Mål

Huvudmålen med examensarbetet var att:

- Analysera problemet på befintlig lösning samt gränssnitts specifikationer.
- Systemera ny lösning med avseende på hårdvaruplattform och VHDL funktion.
- Implementera och simulera funktionen i VHDL.
- Realisera funktionen på LAB om tidsutrymme finns.

1.4 Avgränsning

Tyngdpunkten på examensarbetet har lagts på VHDL-funktionen. Beräkningar på antalet bitar som krävs i A/D- resp D/A-omvandlare för att erhålla en riktig funktion har endast korfattat tagits upp i slutet av rapporten.

2 TM/TC-kanal

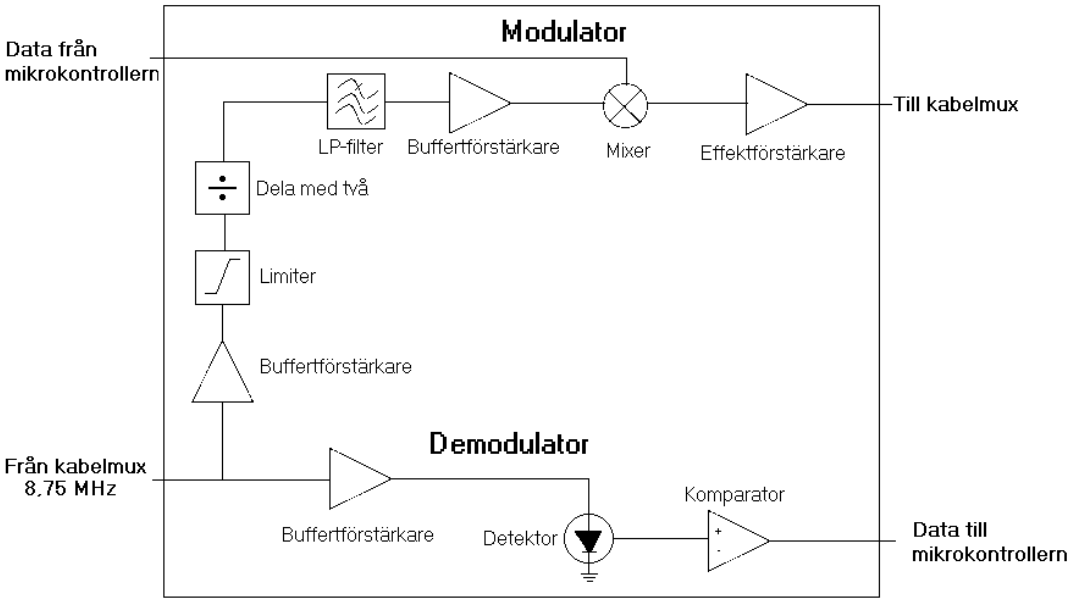
Det block som använder den AM modulerade/demodulerade signalen kallas TM/TC-kanal. TM står för **T**ele**M**etri och TC för **T**ele **C**ommand.

2.1 Användningsområde för TM/TC-kanalen

Kommunikation som går via TM/TC är till för att konfigurera och monitorera radioenheten. Eftersom Allgon Microwave AB:s radiolänkprogram består av många olika konfigurationsmöjligheter behöver man ställa in radioenheten beroende på vilken frekvens som skall användas. För att säkerställa funktion för radiolänken överförs även mätdata via TM/TC-kanalen.

2.2 Befintlig konstruktion

Den nuvarande lösningen beskrivs nedan i korta ordalag. Figur 2 visar ett blockschema över den analoga lösningen. Nackdelar med befintlig konstruktion är att den tar stor plats på kretskortet och att den är svår att modifiera. I kabelmuxen går signaler både från och till TM/TC-kanalen. Även nyttoinformation som ska överföras via radio-länken går i kabelmuxen. Bilaga 1 och 2 visar mätresultat över hur signalen ser ut efter några av blocken i figur 2.



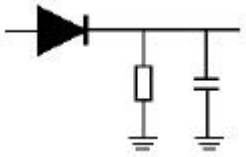
Figur 2. Analog AM modulator/demodulator

2.2 .1 Nuvarande AM-modulator

Bärvågen som används för att modulera datainformationen erhålls från bärvågen i nerlänken (8,75 MHz). Nerlänken innehåller signalen som ska demoduleras. I ”dela med två”, se figur 2, erhålls rätt bärvågsfrekvens (4,375 MHz) m h a en JK-vippa. För att modulera signalen används en pin-diod modulator. Som avslutning på modulationsgrenen sitter en effektförstärkare. Den ger rätt amplitudnivå ut till kabelmuxen plus att den ger anpassing (50 ohm) mot kabelmuxen.

2.2.2 Nuvarande demodulator

För att få anpassning mot kabeln samt för att förstärka upp inkommande signal behövs det en buffertförstärkare, se figur 2, innan demoduleringen. För att detektera AM-signalen använder man sig av en toppvärdesdetektor, som består av en schottky-diod (diod med lågt framspänningsfall, typiskt 0,4 V och snabba omslag) och ett RC-filter.



Figur 3. Toppvärdesdetektor

Ut från demoduleringen vill man ha digitalsignaler. Därför används en komparator som avslutning på demoduleringsgrenen. Komparatorn ser till att man får ”TTL-nivåer” (0 - 5 V) ut.

3 Teori

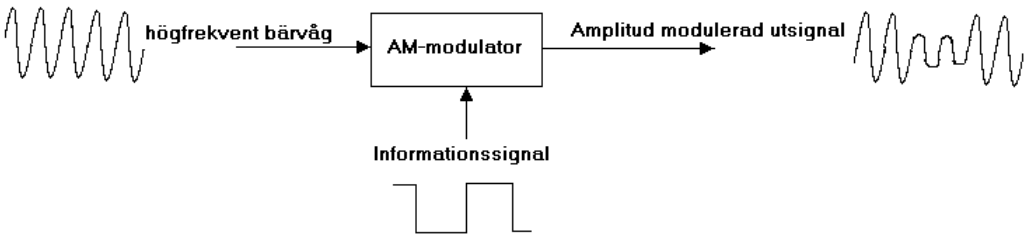
I den här delen beskrivs komponenter som den föreslagna digitala AM modulatorn/demodulatorn är uppbyggd kring.

3.1 AM-modulering

Modulering är en operation som utförs för att anpassa signalformen till den kanal på vilken överföringen ska ske.

Signalen som används för att ”bära” informationen kallas bärvåg. Den högfrekventa bärvågen har tre olika parametrar som man kan variera för att den ska bära informationen. Parametrarna är amplitud, fas och frekvens.

Vid AM-modulering (AM = **A**mplitud **M**odulering), se figur 4, låter man informationen man vill sända förmedlas med bärvågens toppar, amplituden.



Figur 4. AM modulering av fyrkantsvåg

Matematiskt beskrivs en AM-signal enligt:

$$e(t) = (E_c + m(t))\cos 2\pi f_c t \qquad (1)$$

Där $e(t)$ är amplitudmodulerad utsignal, $E_c(t)$ är bärvågsamplitud, $m(t)$ är nyttosignalen och f_c är bärvågsfrekvens.

3.1.1 Modulationsindex

Modulationsindex, m_a, är ett mått på hur stor del den amplitudmodulerade signalen består av nyttosignal respektive av bärvåg. Modulationsindex är definierad som:

$$m_a = \frac{\max|m(t)|}{E_c} \qquad (2)$$

3.2 Korta fakta om VHDL

Förkortningen VHDL står för **V**HSIC **H**ardware **D**escription **L**anguage och VHSIC är förkortning för **V**ery **H**igh **S**peed **I**ntegrated **C**ircuit. VHDL är ett språk som kan beskriva beteendet hos digitala kretsar.

Syntaxen för VHDL påminner i stora delar om kända språk såsom t ex ADA och Pascal men beteendet är helt olika. Medan ADA och Pascal utför sina konstruktioner seriellt, dvs en instruktion i taget, är VHDL till stora del parallellt uppbyggd. Fördelen med parallell uppbyggnad är att prestanda praktiskt taget alltid blir högre än för motsvarande seriella uppbyggnad.

En sak som gör det överskådligt att jobba med VHDL är att språket stödjer användandet av blockscheman. Man kan beskriva sin konstruktion på flera olika nivåer för att sedan på översta nivån sätta samman sina delblock till ett huvudblock. Detta gör att av koden blir lättläst och lätt att förstå för andra än den som skrivit koden.

Beståndsdelarna i VHDL heter Components. Dessa kan lagras i bibliotek för att sedan kunna återanvändas i andra konstruktioner. Har man t ex beskrivit en adderare som fungerar bra både vid simulering och hårdvara är det lätt att återanvända den i andra konstruktioner. Använder man sig av Generics, komponenter som automatiskt kan förändras, får man en kod som lätt är anpassningsbar.

VHDL ger användaren möjligheter att använda sig av olika utvecklingsmetodiker, Top-Down, Bottom-Up eller någon mix. Absolut vanligast är att man använder sig av Top-Down metodik när man konstruerar, eftersom den metodiken har flera fördelar bl a:

- Kortar av konstruktionstiden.
- Möjliggör snabb prototypframtagning av krets.
- Ökar återanvändningsgraden.

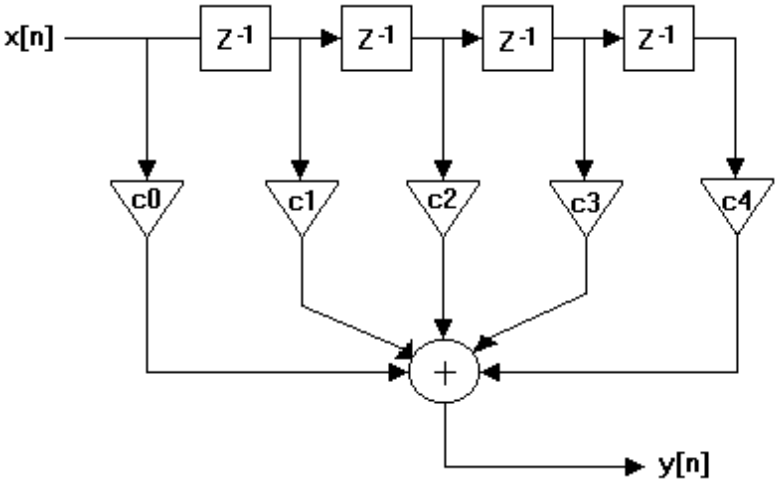
VHDL började som ett specifikations- och modelleringsspråk men är idag ett av de viktigaste standardspråken för att verifiera och konstruera elektronik.

3.3 FIR-filter

FIR-filter är ett digitalt filter där FIR betyder **F**inite **I**mpulse **R**esponse, dvs ändligt impulssvar. Ett FIR-filter har en del fördelar gentemot andra filter. De kan alltid implementeras med exakt linjär fas och är alltid stabila. En annan fördel är att FIR-filter lätt kan implementeras i hårdvara. En nackdel är att ett FIR-filter kräver högre gradtal än motsvarande IIR-filter (**I**nfinite **I**mpulses **R**esponse) vilket innebär att konstruktionen tar mer area i en konstruktion och ger en större fördröjning.

Ett FIR-filters egenskaper bestäms dels av antalet nollställen, dels av hur filter-koefficienterna väljs. När det gäller att bestämma filterkoefficienterna är ett matematikprogram, t ex MATLAB, ett utmärkt hjälpmedel.

Ett typiskt utseende på ett flödesschema för ett FIR-filter av 4:e ordningen visas i figur 7. I figuren är x[n] den samplade insignalen medan y[n] är den digitala utsignalen. c0, c1, c2, c3 och c4 är filterkoefficienterna som bestämmer egenskaperna för filtret.



Figur 7. FIR-filter av 4:e ordningen

4 Förslag på digital lösning

I detta avsnitt beskrivs arbetsgången för den nya lösningen.

4.1 Mätningar på befintlig lösning

I början av projektet ägnades några dagar till att studera kretsscheman över den befintlig lösningen. Syftet med detta var dels att identifiera de delblock som AM modulatorn/demodulatorn är uppbyggd av, se figur 2, dels att kunna genomföra lämpliga mätningar.

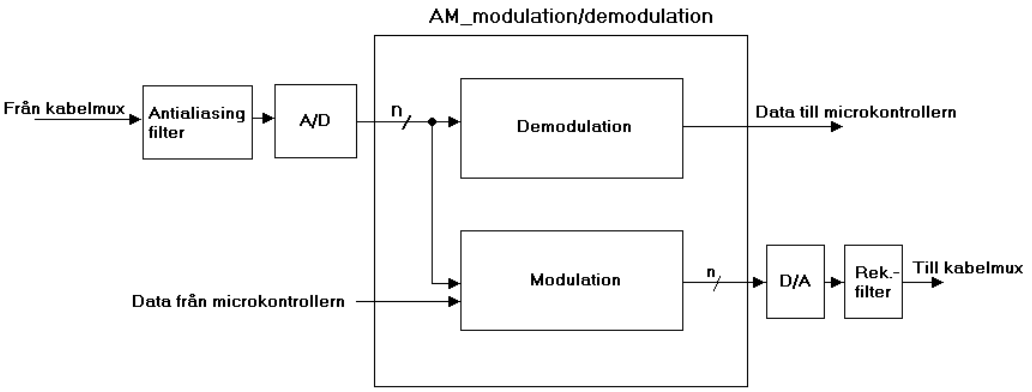
Genom att mäta på kretsen och förstå hur signalen ser ut innan och efter t ex toppvärdesdetektorn är det lättare att sätta sig in i hur VHDL-koden ska se ut. Mätningarna utfördes på Allgon Microwave AB:s laboratorium.

I bilaga 1 och 2 kan mätningarna studeras och relateras till blocken som VHDL-koden är uppbyggd kring.

4.2 VHDL-koden

VHDL-koden har skrivits med antagandet att A/D-omvandlarens ordlängd är åtta bitar. Samma antal har ansetts lämpligt att använda in till D/A-omvandlaren. Vid implementering av koden i en programmerbar krets måste naturligtvis antalet bitar för att uppfylla gällande krav beräknas mycket noggrant.

På toppnivå i koden finns två block, demodulation och modulation, se figur 7. Vad blocken gör i detalj kommer att förklaras här nedan.



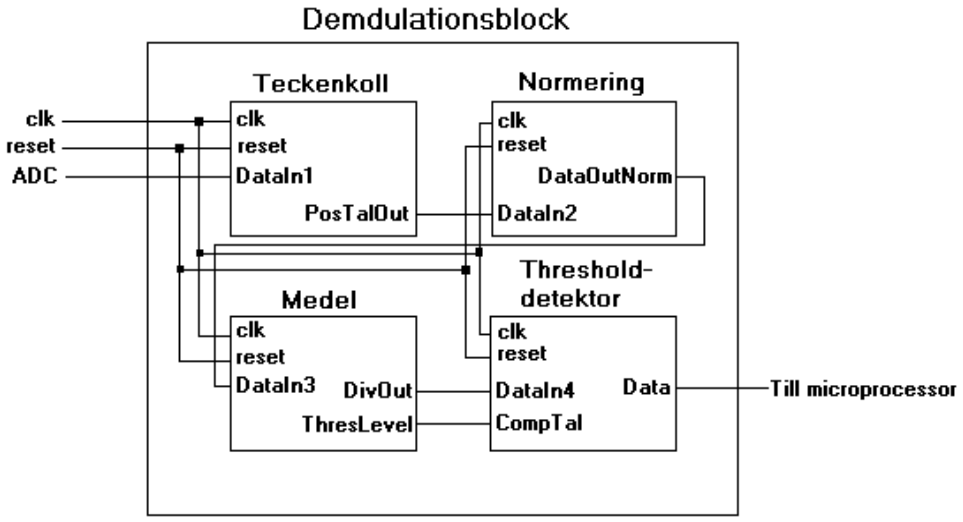
Figur 7. Blockschema på toppnivå.

4.2.1 Demodulationsblockets uppbyggnad

Insignaler till blocket är en AM-modulerad signal som A/D-omvandlaren gör om till en tidsdiskret signal, plus en resetsignal. Signalen från A/D-omvandlaren klockas in med frekvensen 78 MHz. Den AM-modulerade signalen består av bärvåg med frekvensen 8,75 MHz och nyttosignal med bithastigheten 19,2 kbit/s.

Utsignal från blocket är en demodulerad signal som visar vilken information som har mottagits. Den demodulerade signalen går vidare från blocket in till en microprocessor som tolkar bitströmmen. Samtliga block som ingår i demodulationsblocket klockas med 78 MHz och styrs av samma resetsignal.

Hur blocket är uppbyggd i detalj visas i figur 8.



Figur 8. Demodulationsblocket i detalj.

4.2.1.1 Teckenkollblockets uppbyggnad

Blockets in- och utsignaler visas i figur 8. Signalen från A/D-omvandlaren, signal ADC i figur 8, består av både negativa och positiva pulser. Tanken med blocket är att göra en likriktning motsvarande vad dioden gör i den analoga toppvärdesdetektorn, se figur 2 och mätresultat i bilaga 1.

Genom att använda vector-typen **signed** kan negativa respektive positiva pulser detekteras. Använder man signed är MSB:n ((**M**ost **S**ignificant **B**it) en teckenbit. Genom att titta på MSB:n kan man avgöra om talet som kom in är positivt eller negativt. Är talet positivt förblir talet orört och om talet är negativt sätts talet till noll.

4.2.1.2 Normeringsblockets uppbyggnad

Signalen som kommer in från radioenheten via kabelmuxen varierar i amplitud. Till kabelmuxen kommer signalen från utomhusenheten via en koaxialkabel. Hur mycket den kabeln dämpar beror på kabelns längd. För att samma antal MSB ska vara satta oberoende av dämpningen, har ett normeringsblock lagts in. Efter att en resetsignal gått hög letas ett maxvärde upp och därefter väljs vilket talområde som ska användas.

4.2.1.3 Medelvärdesblock

Medelvärdesblocket motsvarar den analoga RC-länken som finns i toppvärdesdetektorn, se figur 3 och bilaga 1. Syftet med blocket är att som utsignal erhålla enveloppen av AM-signalen. Medelvärdesbildningen sker på 128 sampel. Insignalen till blocket kommer från normeringen.

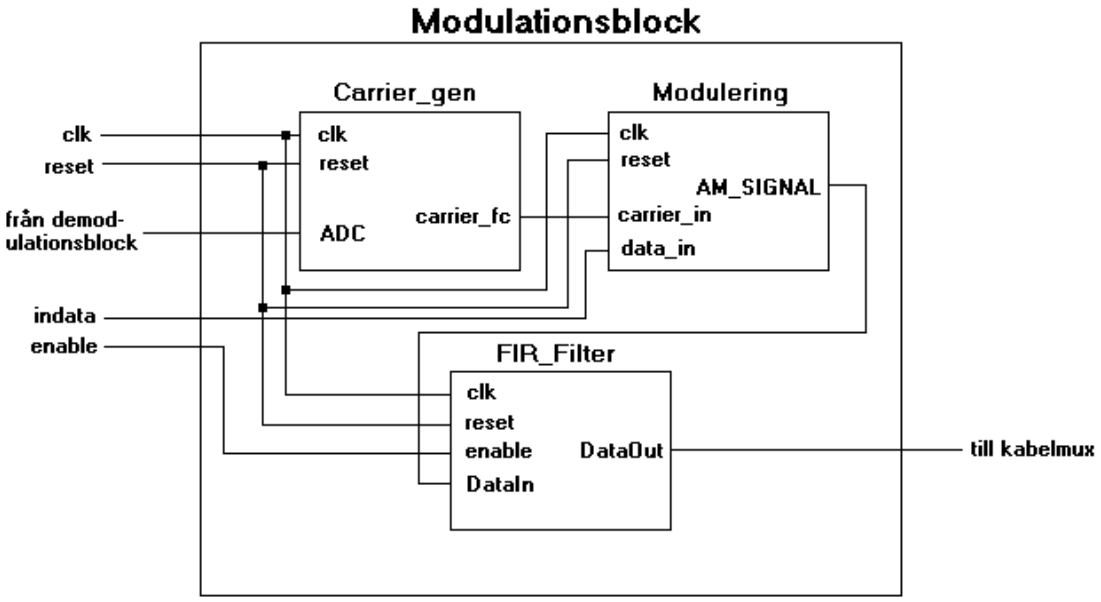
I detta block bestäms även en tröskelnivå som används i nästa block för att få en nivå att kunna detektera data på. För att få fram tröskelnivån letas en max-nivå och en min-nivån upp på medelvärdessignalen. Därefter beräknas tröskelnivån genom att ta medelvärdet av max- och. min-nivån.

4.2.1.4 Tröskeldetektor

I det här blocket detekteras vilken bitström som sänts. Detta block kan jämföras med komparatorn i den analoga lösningen. Insignalen till blocket är enveloppen av AM-signalen och kommer från medelvärdesblocket. Från det blocket kommer också den nivå som ligger till grund för att kunna bestämma om det som skickades var en ’1’ eller en ’0’. Vad blocket gör helt enkelt en jämförelse om enveloppen ligger över eller under tröskelnivån från medelvärdesblocket. Ligger enveloppen över tröskelnivån har en etta sänts och ligger den under har en nolla skickats. Utsignal från blocket är informationen som sändes.

4.2.2 Modulationsblockets uppbyggnad

Insignaler till modulationsblocket är informationen som ska moduleras och bärvågen. Bärvågen som används kommer från demodulationsblocket och har frekvensen 8,75 MHz. Blocken som ingår i modulationen styrs av samma klock- och reset signal och nämns därför inte som insignal i resp. delblock. Figur 9 visar hur modulationsblocket är uppbyggt i detalj.



Figur 9. Modulationsblockets uppbyggnad i detalj

4.2.2.1 Carrier_gen-blockets funktion.

Det här blocket skapar en bärvåg på 4,375 MHz. Kopplingen till det analoga delen är i det här fallet JK-vippan, se figur 2 och bilaga 2. Insignalen till carrier_gen-blocket kommer från demodulationsblocket. Från den signalen återskapas bärvågen i detta block. Bärvågen som kommer in har frekvensen 8,75 MHz Genom att använda sig av vector-typen **signed** och titta på om teckenbiten är satt eller inte får man information om bärvågens frekvens. Får att få en bärvågen på 4,375 MHz delas den inkommande frekvensen med två. Utsignal från blocket är en signal med frekvensen 4,375 MHz.

För att minska risken för att brus påverkar bärvågsfrekvensen är ett alternativ att låta medelvärdesbilda frekvensen över en längre period.

4.2.2.2 Moduleringsblockets funktion

I det här blocket sker AM-moduleringen. Detta block kan jämföras med pin-diod modulatorn i den analoga lösningen, se figur 2 och bilaga 2. Insignaler till blocket är bärvågen och data som ska moduleras. Vad koden gör är att titta på två lägen för bärvågen och databiten:

Data hög och bärgvåg hög ⇒ ut : (max-amplituden bärvåg + max-amplitud data)

Data hög och bärvåg låg ⇒ ut: (min-amplitud data - max-amplitud bärvåg)

Data låg och bärvåg hög ⇒ ut: max-amplitud bärvåg

Data låg och bärvåg låg ⇒ ut: min-amplitud bärvåg

Max- och min-amplitud för bärvåg respektive max- och min-amplitud för data är konstanter. Hur konstanterna väljs beror dels på vilket modulationsindex som ska användas och dels på vilken upplösning som väljs på D/A-omvandlaren.

Utsignal från blocket är en AM-modulerad signal.

4.2.2.3 FIR-filterblocket

Anledningen till att ett FIR-filter behövs är för att dämpa övertoner som fyrkantsvågen ger upphov till, detta för att inte störa andra signaler i kabeln. I kabeln som används går signaler med många olika frekvenser. Bland annat går signalen som koden ska demodulera i samma kabel. Den har frekvensen 8,75 MHz, därför måste FIR-filtret i det här blocket dämpa signalen till kabeln tillräckligt mycket för att inte störa den.

Insignal till blocket är den AM-modulerade signalen samt en enable-signal. FIR-filtret är sista blocket i modulationsgrenen och utsignalen till kabeln är en lågpassfiltrerad signal. Filtret har brytfrekvensen 4,4 MHz (-3 dB). FIR-filtret i den här konstruktionen är inte optimalt på något sätt utan ska ses som ett exempel på hur ett FIR-filter relativt enkelt kan skrivas i VHDL. Det här filtret har endast 7 koefficienter och dämpar bara 6 dB vid 8,75 MHz. Filtrets egenskaper har tagits fram mha Matlab. Alla koefficienterna är 2-potenser för att slippa använda multiplikatorer.

De flesta tillverkare av FPGA:er har möjlighet att generera VHDL-kod som relativt enkelt kan användas. Nackdelarna är att man blir beroende av just den tillverkaren, eftersom deras lösning endast är maximal för sin krets, samt att filtret inte blir optimalt för sin egna konstruktion.

4.2.3 Utvecklingsverktyg vid VHDL-kodningen

Koden skrevs i en texteditor, Codewright 5.1, från Codewrigt. För att verifiera konstruktionerna användes simuleringsprogrammet Modelsim XE från Modeltech. Vid syntes av VHDL-koden användes syntesverktyget Synplify 5.2 från Synplicity.

5 Simuleringsresultat

För att verifiera att VHDL-koden blir enligt specifikation har varje block simulerats kontinuerligt. Vid simulering finns möjlighet att låta simuleringsverktyget ge instimuli till konstruktionen eller att skriva en testbänk som generera insignaler.

5.1 Testbänk

Att skriva en testbänk är nödvändigt vid en större konstruktion där man har många in- och utsignaler. Fördelen med en testbänk är att det går snabbt att simulera medan en nackdel är att testbänken kan innehålla samma tankefel som konstruktionen. En möjlighet är att låta en annan person än konstruktören skriva testbänken för att på så sätt få en annan synvinkel.

I det här projektet har två testbänkar använts, en för demodulationsblocket och en för modulationsblocket. Möjlighet finns att bara använda en testbänk och låta utsignalen från demodulationsblocket (datatakten) vara insignal till testbänken för att moduleras av bärvågen och att låta utsignalen från modulationsblocket vara insignal till demodulationsblocket.

En nackdel med projektets testbänk är att datatakten är ”0101…”, ”000..” eller ”111…” med andra ord kan inte alla olika insignaler testas.

5.2 Simuleringsresultat för demodulationsblocket

Simuleringsresultatet visar att det är möjligt att beskriva en AM-demodulator i VHDL. Bilaga 3 visar simuleringsresultatet och kan jämföras med mätningar gjorda på den analoga lösningen, se figur 12 i bilaga 1.

5.3 Simuleringsresultat för modulationsblocket

Figur 10 i bilaga 3 visar signalen efter AM-moduleringen, den figuren kan jämföras med figur 14 i bilaga 2, som visar signalen efter den analoga AM-moduleringen. Båda figurerna visar en AM-signal med modulationsindex 50 %.

6 Implementeringsförslag för VHDL-koden

Ett av målen med detta examensarbete var att implementera koden i en FPGA. Kretsen som koden implementerades i var en i Xilinx XC4000-serie.

Indata till syntesverktyget är VHDL-koden, teknologifil och constraints. Utdata är en rapportfil och en nätlista. Från rapportfilen får man information om hur syntesen har utförts och hur resultatet blev. Nätlista är ett schema med grindar. Constraints är de parametrar med vilket syntesverktyget kan påverkas. Parametrarna kan t ex vara periodtider för klockingångar, temperaturområde, areakrav eller timingkrav.

Vid syntes av den nya lösningen sattes inga speciella tids- eller areakrav. Resultatet från syntesen är att koden går att implementera.

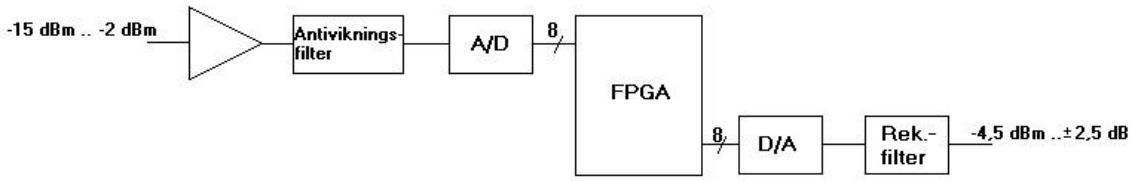
Eftersom det är tänkt att den nya lösningen av AM modulatorn/demodulatorn ska integreras i redan befintliga FPGA:er laddas den nya lösningen enligt tidigare fastställda rutiner, dvs laddas från CPU:n.

Två parametrar är möjliga att ställa vid laddning av FPGA:n

- Modulationsindex (ModIndex i VHDL-koden).
- Maximal utsignalnivå (P_Ut i VHDL-koden) på AM-signalen.

7 Förslag till gränssnitt mot FPGA

Gränssnittet mot FPGA:n när det gäller den mottagna signalen utgörs av en OP-förstärkare, ett antivikningsfilter och en A/D-omvandlare. På sändarsidan utgörs gränssnittet av en D/A-omvandlare och ett rekonstruktionsfilter. Figur 10 visar en principskiss över gränssnittet.



Figur 10. Principskiss över gränssnitt mot FPGA.

Vid beräkning av OP-förstärkarens gain måste signalnivån in från kabelmuxen, pass-bandsrippel i antivikningsfilter samt vilken spänningsnivå A/D-omvandlaren använder beaktas.

Antivikningsfiltret är av analog filter av lågpasskaraktär. Mottagna signalen har frekvensen 8,75 MHz och samplingsfrekvensen är 78 MHz. Filtret ska dämpa 50 dB vid spegelfrekvensen 69,25 MHz alltså behöver filtret ha tre poler (20dB/dekad).

Signalen från kabelmuxen kan max vara dämpad 20 dB. För att dektektion av mottagen signal ska vara möjlig krävs 4 (24 dB) bitar in till tröskeldetektorn. Antal bitar i A/D-omvandlaren baseras på dessa fakta. 20 dB + 24 dB = 44 dB gör att A/D-omvandlaren måste ha en dynamik på 48 dB, dvs 8 bitar.

Antalet bitar i D/A-omvandlaren bestäms av vilka egenskaper FIR-filtret ska ha. I den förslagna VHDL-koden används 8 bitar i FIR-filtret. Rekonstruktionsfiltret ska dämpa 78 MHz ± 4,375 MHz (samplingsfrekvens och nyttosignalen).

Förslag till OP-förstärkare: AD8041 från Analog Devices.
Förslag till A/D-omvandlare (8 bitar): AD9057 från Analog Devices.
Förslag till D/A-omvandlare (8 bitar): AD9708 från Analog Devices.

Datablad om ovan angivna komponenter finns på Analog Devices hemsida:
http://www.analog.com.

8 Diskussion och slutsats

Mycket tid i början av projektet ägnades åt att förstå den befintliga lösningen genom att läsa kretsscheman och genom mätningar. Den tiden var väl använd eftersom det underlättade arbetet med att ta fram en struktur för VHDL-koden. Många komponenter i VHDL-koden är direkt jämförbara med motsvarande komponent i den befintliga lösningen.

Demoduleringsblocket fungerar som tänkt. En svaghet är tröskelnivåberäkningen som beräknar ett medelvärde på endast två tal.

I moduleringsblocket är FIR-filtret endast ett förslag på hur ett filter kan göras i VHDL. Det finns ett krav på att filtret ska dämpa 30 dB vid 8,75 MHz vilket filtret i den nya lösningen inte klarar av. Eftersom en D/A-omvandlare med efterföljande filter och kabelmuxfilter finns efter FPGA:n behöver förmodligen inte FIR-filtret dämpa 30 dB.

Förhoppningen när examensarbetet startade var att hinna med hela konstruktionskedjan, från specifikation till färdig krets. Tyvärr fanns inte utrymme för att få fram en färdig krets att mäta på, utan slutsatsen grundas på simuleringsresultaten.

Simuleringsresultatet visar att det är möjligt att beskriva en digital AM modulator/demodulator med hjälp av VHDL. Hur konstruktionen fungerar i hårdvara återstår dock att se.

9 Referenser

[1] Stefan Sjöholm, Lennart Lindh, "VHDL för konstruktion", Studentlitteratur, 1999, 3:e upplagan.

[2] Paul H. Young, "Electronic Communication Techniques", Prentice Hall 4:e upplagan.

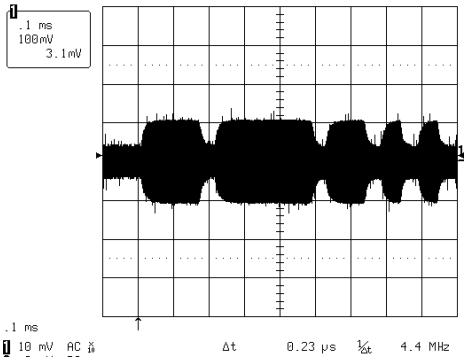
[3] Xilinx hemsida, http://xilinx.com/apps/xapp.htm

[4] Analog Devices hemsida, http://www.analog.com

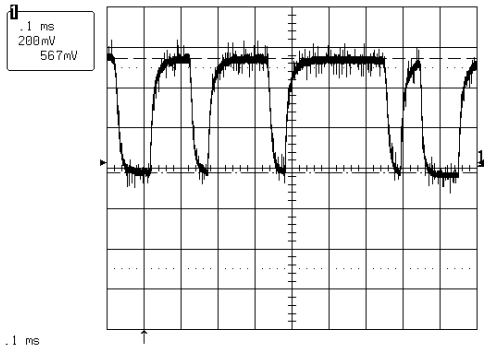
Bilaga 1

Mätningar på analog AM demodulator

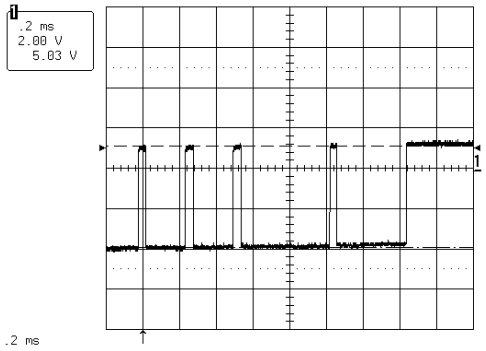
Figurerna 11, 12 och 13 visar hur AM-signalen ser ut från det att signalen kommer in från kabelmuxen tills det att bitströmmen har detekterats. Figur 11 visar den signal som kommer in till A/D-omvandlaren i den nya lösningen.



Figur 11. Signal från kabelmux.



Figur 12. Signal efter toppvärdesdetektorn.

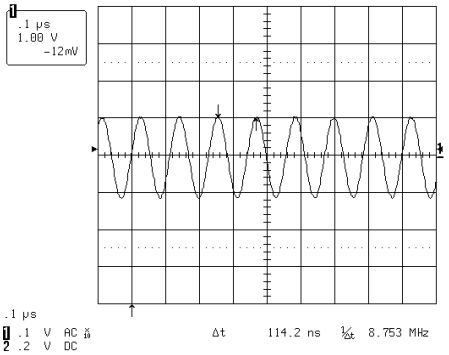


Figur 13. Signal efter komparator.

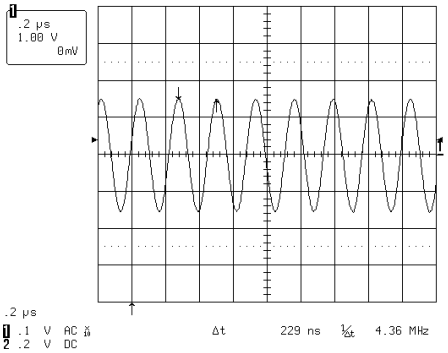
Bilaga 2

Mätningar på analog AM modulator

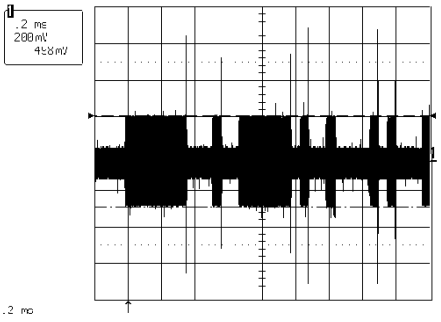
Figurerna 13, 14 och 15 visar signalen från bärvågsåtervinning tills det att signalen har amplitud-modulerats.



Figur 13. Signal efter limitern



Figur 14. Signal efter division med två.

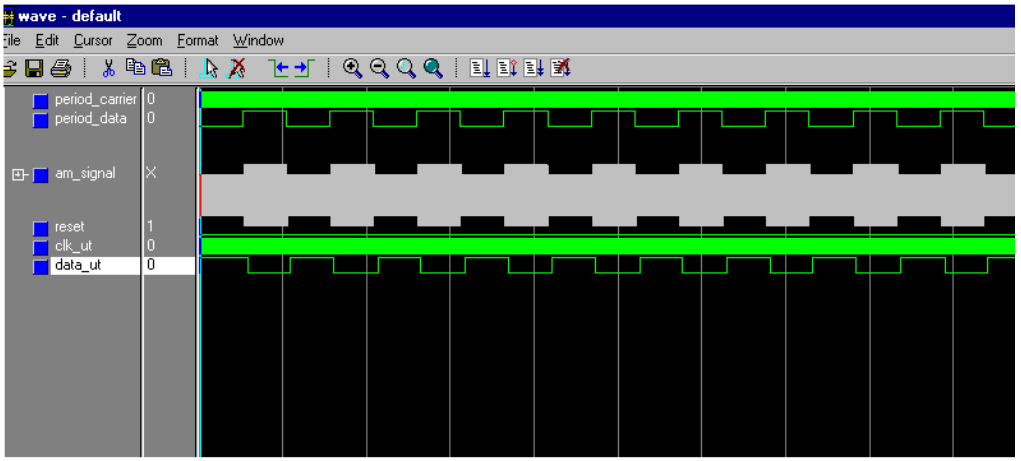


Figur 15. AM-modulerad signal

Bilaga 3

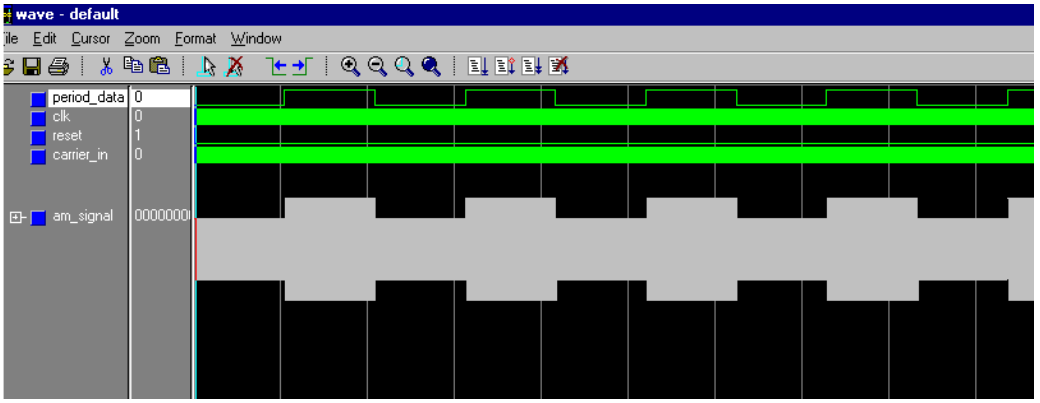
Simuleringsresultat

Figuren 16 visar simuleringsresultatet av AM demoduleringen. Am_signal i figuren är den simulerade signalen från kabelmuxen. Data_ut är resultatet efter demoduleringen.



Figur 16. Simuleringsreslutat från AM demodulatorn.

Figur 17 visar simuleringsresultatet av AM moduleringen. Am_signal är signalen efter moduleringen. Period_data är signalen som moduleras.



Figur 17. Simuleringsresultat från AM modulatorn.

Bilaga 4

```
------------------------------------------------------------------------------
--                                                                          --
--  Copyright Allgon Microwave AB, Göteborg, Sweden 2001                    --
--                                                                          --
------------------------------------------------------------------------------
--                                                                          --
-- Project:              Examensarbete                                      --
-- Subsystem:            TM/TC                                              --
-- Modulname:            Modulationsblock                                   --
-- Description:          Modulerar en bittakt på 19,2 kbit/s                --
--                       med en bärvåg frekvensen 4,375 MHz.                --
-- Edition:               010505                                            --
------------------------------------------------------------------------------

------------------------------------------------------------------------------
--        Genererar bärvågen genom att spara teckenbiten                    --
------------------------------------------------------------------------------
Library ieee;
Use ieee.std_logic_1164.all;                    --bibliotek
Use ieee.std_logic_signed.all;
Use ieee.std_logic_arith.all;
------------------------------------------------------------------------------
Entity carrier_gen is
        port(
              ADC:                              in std_logic_vector(7 downto 0);
              reset:                            in std_logic;
              clk:                              in std_logic;
              carrier_fc:                       out std_logic
             );
end;

Architecture ark_carrier_gen of carrier_gen is

signal                                          count:std_logic;
signal                                          tal_1:integer range -256 to 256;
signal                                          tal_2:signed(7 downto 0);
signal                                          A1:std_logic;
signal                                          ut:std_logic;

begin
   tal_1<=conv_integer(ADC);                     -- std_logic_vector => integer
   tal_2<=conv_signed(tal_1,8);                  --integer => signed

        process(clk,reset)
        begin
             if reset='1' then
                count<='0';
                A1 <= '0';
             elsif clk'event and clk='1' then
                          A1 <= tal_2(7);
                        if A1 > tal_2(7) then
                            count <= not(count);
                        end if;
              end if;
```

```
        end process;
 carrier_fc<=count;
end;



-------------------------------------------------------------------------------------------------------
-- Modulerar infobitarna genom att addera resp. subtrahera bärvåg med infobitarna     --
-------------------------------------------------------------------------------------------------------
Library IEEE;
Use IEEE.std_logic_1164.all;
Use IEEE.std_logic_signed.all;                          --bibliotek
Use IEEE.std_logic_arith.all;
-------------------------------------------------------------------------------------------------------
Entity modulering is
        port(
              clk:              in std_logic;
              reset:            in std_logic;
              carrier_in:       in std_logic;
              data_in:          in std_logic;
               P_Ut:            in integer range 0 to 255;     --parametersättning
               ModIndex:        in integer range 0 to 255;     --parametersättning
               AM_SIGNAL:       out signed(7 downto 0)
                );
end;

Architecture ark_modulering of modulering is

signal carrier:                 integer range -127 to 128;
signal data:                    integer range -127 to 128;
signal summa:                   integer range -127 to 128;
signal max_amplitud_data:       integer range -127 to 128;
signal max_amplitud_carrier:    integer range -127 to 128;
signal min_amplitud_carrier:    integer range -127 to 128;

begin
        process(reset,ModIndex,P_Ut)
        begin
                if reset = '1' then
                        max_amplitud_data<=0;
                        max_amplitud_carrier<=0;
                        min_amplitud_carrier<=0;
                else
                        max_amplitud_data<=ModIndex;
                        max_amplitud_carrier<=(P_Ut-ModIndex);
                        min_amplitud_carrier<=-(P_Ut-ModIndex);
                end if;
        end process;

        process(clk,reset)
        begin
          if reset = '1' then
                summa<=0;
                carrier<=0;
                data<=0;
          elsif clk'event and clk = '1' then
```

```
                if  data_in='1' and carrier_in='1' then
                        summa<=(max_amplitud_data + max_amplitud_carrier);
                elsif data_in='1' and carrier_in='0' then
                        summa<=(min_amplitud_carrier - max_amplitud_data);
                elsif data_in='0' and carrier_in='1' then
                        summa<=max_amplitud_carrier;
                else
                        summa<=min_amplitud_carrier;
                end iF;
          end if;
        end process;
AM_SIGNAL<=conv_signed(summa, 8);
end;

-------------------------------------------------------------------------------------------------------
-- FIR-filter för att få ner övertoner samt för att ej störa andra signaler i kabelmuxen.          --
-- Filtret dämpar 6 dB vid 8,75 MHz, dvs filtret uppfyller ej kravet på 30 dB dämpning!!!!   --
-------------------------------------------------------------------------------------------------------

Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_arith.all;

Entity FIR_Filter is
        port(
             DataIn:                    in signed(7 downto 0);
             clk:                       in std_logic;
             reset:                     in std_logic;
             enable:                    in std_logic;
             DataOut:                   out std_logic_vector(7 downto 0)
             );
end;

Architecture ark_FIR_Filter of FIR_Filter is

  signal d1:                signed(7 downto 0);
  signal d2:                signed(7 downto 0);
  signal d3:                signed(7 downto 0);
  signal d4:                signed(7 downto 0);
  signal d5:                signed(7 downto 0);
  signal d6:                signed(7 downto 0);
  signal b0:                signed(7 downto 0);
  signal b1:                signed(7 downto 0);
  signal b2:                signed(7 downto 0);
  signal b3:                signed(7 downto 0);
  signal b4:                signed(7 downto 0);
  signal b5:                signed(7 downto 0);
  signal b6:                signed(7 downto 0);
  signal b7:                signed(7 downto 0);
  signal i1:                integer range -256 to 255;
  signal i2:                integer range -256 to 255;
  signal i3:                integer range -256 to 255;
  signal i4:                integer range -256 to 255;
  signal internDataOut:     integer range -2048 to 2047;

        begin
```

```
process(clk,reset)
 begin
    if reset='1' then
          d1<=(others=>'0');
          d2<=(others=>'0');
          d3<=(others=>'0');
          d4<=(others=>'0');
          d5<=(others=>'0');
          d6<=(others=>'0');
          b0<=(others=>'0');
          b1<=(others=>'0');
          b2<=(others=>'0');
          b3<=(others=>'0');
          b4<=(others=>'0');
          b5<=(others=>'0');
          b6<=(others=>'0');
         b7<=(others=>'0');
        internDataOut<=0;
  elsif clk'event and clk='1' then
    if enable='1' then
          if DataIn(7)='0' then
                b0<="000000" & DataIn(6 downto 5);        --0,0156
           else b0<="111111" & DataIn(6 downto 5);        --0,0156
           end if;
          if d1(7)='0' then
                b1<="0000" & d1(6 downto 3);              --0.0625
          else b1<="1111" & d1(6 downto 3);     --0.0625
          end if;
          if d2(7)='0' then
                b2<="00" & d2(6 downto 1);         --0.25
          else b2<="11" & d2(6 downto 1);          --0.25
          end if;
          if d3(7)='0' then
                 b3<='0' & d3(6 downto 0);         --0.5
          else b3<='1' & d3(6 downto 0);           --0.5
          end if;
          if d4(7)='0' then
                 b4<="00" & d4(6 downto 1);
          else b4<="11" & d4(6 downto 1);
         end if;
          if d5(7)='0' then
                 b5<="0000" & d5(6 downto 3);
        else b5<="1111" & d5(6 downto 3);
          end if;
          if d6(7)='0' then
                  b6<="000000" & d6(6 downto 5);
           else b6<="111111" & d6(6 downto 5);
           end if;

           d1<=DataIn;
           d2<=d1;
           d3<=d2;
           d4<=d3;
           d5<=d4;
           d6<=d5;                  --skiftar data
```

```
                end if;

                i1<=conv_integer((b0+ b1));
                i2<=conv_integer((b2+ b3));
                i3<=conv_integer((b4+ b5));
                i4<=conv_integer(b6);
                internDataOut<=(i1+ i2+ i3 + i4);

                 end if;
                end process;
DataOut<=conv_std_logic_vector(internDataOut,8);
end;
```

```
-------------------------------------------------------------------------------------------
--                    Huvudprogram för modulationsblocket                                --
-------------------------------------------------------------------------------------------
LIBRARY ieee;
Use ieee.std_logic_1164.all;                    --bibliotek
Use ieee.std_logic_signed.all;
Use ieee.std_logic_arith.all;

Entity modulations_block is
            port(
                  InfoData:          in std_logic;
                  SignalIn:          in std_logic_vector(7 downto 0);
                  clk:               in std_logic;
                  reset:             in std_logic;
                  enable:            in std_logic;
                  P_Ut:              in integer range 0 to 255;
                  ModIndex:          in integer range 0 to 255;
                  AM_SignalOut:  out std_logic_vector(7 downto 0)
                  );
end;

Architecture ark_modulations_block of modulations_block is

component carrier_gen
port(
      ADC:                              in std_logic_vector(7 downto 0);
      reset:                            in std_logic;
      clk:                              in std_logic;
      carrier_fc:                       out std_logic
      );
end component;

component modulering
port(
       clk:                             in std_logic;
       reset:                           in std_logic;
       carrier_in:                      in std_logic;
       ModIndex:                        in integer range 0 to 255;
```

```
        P_Ut:                           in integer range 0 to 255;
        data_in:                        in std_logic;
        AM_SIGNAL:                      out signed(7 downto 0)
            );
end component;

component FIR_Filter
port(
        DataIn:                 in signed;
        clk:                    in std_logic;
        reset:                  in std_logic;
        enable:                 in std_logic;
        DataOut:                out std_logic_vector(7 downto 0));
end component;


signal i1:std_logic;
signal i2:signed(7 downto 0);

begin
     gate_carrier_gen:carrier_gen   port map(
                                            clk=>clk,
                                            reset=>reset,
                                            carrier_fc=>i1,
                                            ADC=>SignalIn
                                             );

   gate_modulering:modulering       port map(
                                              P_Ut=>P_Ut,
                                              ModIndex=>ModIndex,
                                              clk=>clk,
                                              reset=>reset,
                                              AM_SIGNAL=>i2,
                                              carrier_in=>i1,
                                              data_in=>InfoData
                                               );

  gate_FIR_Filter:FIR_Filter        port map(
                                             clk=>clk,
                                             reset=>reset,
                                             enable=>enable,
                                              DataOut=>AM_SignalOut,
                                              DataIn=>i2
                                              );
end;
```

```
--------------------------------------------------------------------------------
--                                                                            --
--  Copyright Allgon Microwave AB, Göteborg, Sweden 2001                      --
--                                                                            --
--------------------------------------------------------------------------------
--                                                                            --
-- Project:              Examensarbete                                        --
-- Subsystem:            TM/TC                                                --
-- Modulname:            Testbänk                                             --
-- Description:          Testbänk för modulering av AM-signal                 --
-- Edition:              010505                                               --
--------------------------------------------------------------------------------

------------------------------------------------------------------------------------
--            Testbänken genererar insignaler till modulationsblocket             --
------------------------------------------------------------------------------------
Library IEEE;
Use IEEE.std_logic_1164.all;
Use IEEE.std_logic_arith.all;
--------------------------------------------------------------------------------
Entity test_testbench_modulering is
        port(
                data_ut:                out std_logic_vector(7 downto 0)
                );
end;

architecture ark_test_testbench_modulering of test_testbench_modulering is

component modulations_block
        port(
                InfoData:               in std_logic;
                SignalIn:               in std_logic_vector(7 downto 0);
                clk:                    in std_logic;
                reset:                  in std_logic;
                enable:                 in std_logic;
                P_Ut:                   in integer range 0 to 255;
                ModIndex:               in integer range 0 to 255;
                AM_SignalOut:           out std_logic_vector(7 downto 0)
                );
end component;


for U1: modulations_block use entity work.modulations_block(ark_modulations_block);

constant P_Ut_testb:            integer :=40;
constant ModIndex_testb:        integer :=20;

signal period_carrier:          std_logic :='0';
signal period_data:             std_logic :='0';
signal reset_ut:                std_logic :='0';
signal clk_ut:                  std_logic :='0';
signal summa:                   integer range -127 to 128;
signal AM_signal:               std_logic_vector(7 downto 0);  -- signed(7 DOWNTO 0);
signal enable_ut:               std_logic;
signal max_amplitud_data:       integer;
signal max_amplitud_carrier:    integer;
```

```
signal min_amplitud_carrier:      integer;

begin
        clk_ut<=not clk_ut after 6410 ps;                       --78 MHz
        reset_ut<='1', '0' after 80 ns;                         --reset vid start
        period_carrier<=not period_carrier after 57 ns;         --8.75 MHz
        period_data<=not period_data after 52 us;               --19.2 kbit/s
        enable_ut<='0', '1' after 80 ns;

        process(reset_ut)
        begin
                if reset_ut ='1' then
                        max_amplitud_data<=0;
                        max_amplitud_carrier<=0;
                        min_amplitud_carrier<=0;
                else
                        max_amplitud_data<=ModIndex_testb;
                        max_amplitud_carrier<=(P_Ut_testb-ModIndex_testb);
                        min_amplitud_carrier<=-(P_Ut_testb-ModIndex_testb);
                end if;
        end process;

  process(period_carrier,reset_ut)
  begin
            if reset_ut = '1' then
                summa<=0
            elsif period_carrier'event then
                if  period_data='1' and period_carrier='1' then
                        summa<=(max_amplitud_data + max_amplitud_carrier);
                elsif period_data='1' and period_carrier='0' then
                        summa<=(min_amplitud_carrier - max_amplitud_data);
                elsif period_data='0' and period_carrier='1' then
                        summa<=max_amplitud_carrier;
                else
                        summa<=min_amplitud_carrier;
                end if;
            end if;
end process;
am_signal<=conv_std_logic_vector(summa,8);


U1:modulations_block      PORT MAP(
                                reset=>reset_ut,
                                clk=>clk_ut,
                                SignalIn=>AM_signal,
                                InfoData=>period_data,
                                enable=>enable_ut,
                                AM_SignalOut=>data_ut,
                                ModIndex=>ModIndex_testb,
                                P_Ut=>P_Ut_testb
                                );
end;
```

```
--------------------------------------------------------------------------------
--                                                                            --
--  Copyright Allgon Microwave AB, Göteborg, Sweden 2001                      --
--                                                                            --
--------------------------------------------------------------------------------
--                                                                            --
-- Project:              Examensarbete                                        --
-- Subsystem:            TM/TC                                                --
-- Modulname:            Demoduleringsblock                                   --
-- Description:          AM demodulering                                      --
-- Edition:              010505                                               --
--------------------------------------------------------------------------------

------------------------------------------------------------------------------------
-- Teckenkoll läser av teckenbiten, negativa tal sätts till 0 och positiva tal --
--              tal förblir orörda                                            --
------------------------------------------------------------------------------------
Library ieee;
Use ieee.std_logic_1164.all;                    --bibliotek
Use ieee.std_logic_signed.all;
Use ieee.std_logic_arith.all;
--------------------------------------------------------------------------------
Entity TeckenKoll is
        port(
            DataIn1:            in std_logic_vector(7 downto 0);
            reset:              in std_logic;
            clk:                in std_logic;
            PosTalOut:          out std_logic_vector(7 downto 0));
end;

Architecture ark_TeckenKoll of TeckenKoll is

signal                          InternOut:signed(7 downto 0);
signal                          Intern1:integer range -255 to 255;
signal                          Intern2:signed(7 downto 0);

begin
        process(clk,reset)
        begin
            Intern1<=conv_integer(DataIn1);             -- std_logic_vector => integer
            Intern2<=conv_signed(Intern1,8);            --integer => signed
         if reset='1' then
              InternOut<=(others=>'0');
             Intern1<=0;
             Intern2<=(others=>'0');
        elsif clk'event and clk='1' then
                if Intern2(7)='0' then
                      InternOut<=Intern2;
                else InternOut<=(others=>'0');
                end if;
         end if;
        end process;
        PosTalOut<=conv_std_logic_vector(InternOut,8);
        end;
```

```
-------------------------------------------------------------------------------
--          Normeringsblock, utsignalen är 4 MSB oberoende av vilken   --
--          nivå som kommer in från A/D-omvandlaren                    --
-------------------------------------------------------------------------------

Library ieee;
Use ieee.std_logic_1164.all;                    --bibliotek
Use ieee.std_logic_unsigned.all;

Entity normering is
        port(
            DataIn2:            in std_logic_vector(7 downto 0);
            clk:                in std_logic;
            reset:              in std_logic;
            DataOutNorm:        out integer     range 0 to 255
    );
End;

Architecture ark_normering of normering is

signal Intern1:                 std_logic_vector(7 downto 0);
signal Intern2:                 std_logic_vector(3 downto 0);
signal StartValue:              std_logic_vector(7 downto 0);
signal count:                   std_logic_vector(3 downto 0);

begin
        process(clk,reset)
        begin
          if reset='1' then
              Intern1<=(others=>'0');
              Intern2<=(others=>'0');
              StartValue<=(others=>'0');
          elsif clk'event and clk='1' then
                Intern1<=DataIn2;
                if Intern1 > StartValue then
                    StartValue<=Intern1;
                end if;
                if StartValue(7)='1' then
                    Intern2<=Intern1(7 downto 4);
                    DataOutNorm<=conv_integer(Intern2);
                elsif StartValue(6)='1' then
                    Intern2<=Intern1(6 downto 3);
                    DataOutNorm<=conv_integer(Intern2);
                elsif StartValue(5)='1' then
                    Intern2<=Intern1(5 downto 2);
                    DataOutNorm<=conv_integer(Intern2);
                elsif StartValue(4)='1' then
                    Intern2<=Intern1(4 downto 1);
                    DataOutNorm<=conv_integer(Intern2);
                elsif StartValue(3)='1' then
                    Intern2<=Intern1(3 downto 0);
                    DataOutNorm<=conv_integer(Intern2);
                elsif StartValue(2)='1' then
                    Intern2<=Intern1(2 downto 0) & '0' ;
                    DataOutNorm<=conv_integer(Intern2);
                elsif StartValue(1)='1' then
```

```
                        Intern2<=Intern1(1 downto 0) & "00" ;
                        DataOutNorm<=conv_integer(Intern2);
                    else Intern2<=Intern2;
                    end if;
                end if;
            end process;
end;


-------------------------------------------------------------------------------------------------
-- Beräknar medelvärdet på 128 samples för att kunna läsa av enveloppen.                --
--          Blocket beräknar också ett tröskelvärde som används i thresholddetektorn.   --
-------------------------------------------------------------------------------------------------
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_arith.all;                                                   --bibliotek
Use ieee.std_logic_unsigned.all;
-------------------------------------------------------------------------
Entity medel is
        port(
            DataIn3:            in integer range 0 to 32767;
            clk:                in std_logic;
            reset:              in std_logic;
            ThresLevel:         out std_logic_vector(9 downto 0);
            DivUt:              out std_logic_vector(9 downto 0)
            );
end;

Architecture ark_medel of medel is

constant            AntalSamples:integer :=128;
signal Tal1:        integer range 0 to 32767;
signal Tal2:        integer range 0 to 32767;
signal Count:       integer range 0 to AntalSamples;
signal SummaVector: std_logic_vector(10 downto 0);
signal troskel:     std_logic_vector(10 downto 0);
signal DivIntern:   std_logic_vector(10 downto 0);
signal max_Level:   std_logic_vector(10 downto 0);
signal min_Level:   std_logic_vector(10 downto 0);

begin
    process(clk,reset)
    begin
        if reset='1' then
            Tal1<=0;
            Tal2<=0;
            Count<=0;
            SummaVector<=(others=>'0');
            DivIntern <= (others => '0');
        elsif clk'event and clk='1' then
            if Count < AntalSamples then
                Count<=Count +1;
                Tal1<=(Tal1 + DataIn3);
            else
                Tal2<=Tal1;
                count<=0;
```

```
                    Tal1<=DataIn3;
                    SummaVector<=conv_std_logic_vector(Tal2,11); --integer => std_logic_vector
                    DivIntern<=shr(SummaVector,"110");
                end if;
                end if;
            end process;

            process(reset,DivIntern)
            begin
                if reset='1' then
                    min_level<="00000001111";
                    max_level<="00000001111";
                    troskel <=(others => '0');
                else
                    if DivIntern > troskel then
                    max_level <= shr(DivIntern + max_level,"1");
                else
                    min_level <= shr(DivIntern + min_level,"1");
                end if;
                    troskel<=shr((max_level + min_level),"1");
            end if;
    end process;
   ThresLevel<=troskel(9 downto 0);
   Divut<=DivIntern(9 downto 0);
end;
-------------------------------------------------------------------------------------------
--          Bestämmer när data_ut = 1 resp. data_ut = 0, beroende                --
--          på vilken tröskelvärde som kommer in från normeringsblocket.   --
-------------------------------------------------------------------------------------------
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
-----------------------------------------------------------------------------
entity threshold_detektor is
            port(
                DataIn4:        in std_logic_vector(9 downto 0);
                CompTal:        in std_logic_vector(9 downto 0);
                clk:            in std_logic;
                reset:          in std_logic;
                Data:           out std_logic);
end;

Architecture ark_threshold_detektor of threshold_detektor is

signal DataIntern:          std_logic;
signal Intern:              integer range 0 to 255;
signal Intern2:             integer range 0 to 255;

begin
            process (clk,reset)
            begin
              if reset='1' then
                DataIntern<='0';
                Intern<=0;
              elsif clk'event and clk='1' then
                    if DataIn4 >= CompTal then
```

```
                        DataIntern<='1';
                    else DataIntern<='0';
                    end if;
                end if;
            end process;
Data<=DataIntern;
end;

--------------------------------------------------------------------------------
--          Huvudprogram för demodulering av AM-signal        --
--------------------------------------------------------------------------------
Library ieee;
Use ieee.std_logic_1164.all;                    --bibliotek
Use ieee.std_logic_arith.all;
--------------------------------------------------------------------------------
Entity demod is
        port(
                SignalADC:          in std_logic_vector(7 downto 0);
                clk:                in std_logic;
                reset:              in std_logic;
                DataOut:            out std_logic
                );
end;

Architecture ark_demod of demod is

component TeckenKoll
        port(
              DataIn1:              in std_logic_vector(7 downto 0);
              reset:                in std_logic;
              clk:                  in std_logic;
              PosTalOut:            out std_logic_vector(7 downto 0)
              );
end component;

component normering
        port(
              DataIn2:              in std_logic_vector(7 downto 0);
              clk:                  in std_logic;
              reset:                in std_logic;
              DataOutNorm:          out integer range 0 to 255
             );
end component;

component medel
        port(
                DataIn3:            in integer range 0 to 32767;
                clk:                in std_logic;
                reset:              in std_logic;
                thresLevel:         out std_logic_vector(9 downto 0);
                Divut:              out std_logic_vector(9 downto 0)
                );
end component;

component threshold_detektor
        port(
```

```
                DataIn4:        in std_logic_vector(9 downto 0);
                Comptal:        in std_logic_vector(9 downto 0);
                clk:            in std_logic;
                reset:          in std_logic;
                Data:           out std_logic
              );
end component;

signal i1:          std_logic_vector(7 downto 0);
signal i2:          integer range 0 to 255;
signal i3:          std_logic_vector(9 downto 0);
signal i4:          std_logic_vector(9 downto 0);

begin

        gate_TeckenKoll:TeckenKoll          port map(
                                                    DataIn1=>SignalADC,
                                                    PosTalOut=>i1,
                                                    clk=>clk,
                                                    reset=>reset
                                                    );

        gate_normering:normering            port map(
                                                    clk=>clk,
                                                    DataIn2=>i1,
                                                    reset=>reset,
                                                DataOutNorm=>i2
                                                    );

        gate_medel:medel                    port map(
                                                    DataIn3=>i2,
                                                    clk=>clk,
                                                    reset=>reset,
                                                    thresLevel=>i3,
                                                    DivUt=>i4
                                                    );

        gate_threshold_detektor:threshold_detektor  port map(
                                                    DataIn4=>i3,
                                                    clk=>clk,
                                                    reset=>reset,
                                                    data=>DataOut,
                                                Comptal=>i4
                                                    );
end;
```

```
--------------------------------------------------------------------------------
--                                                                            --
--  Copyright Allgon Microwave AB, Göteborg, Sweden 2001                      --
--                                                                            --
--------------------------------------------------------------------------------
--                                                                            --
-- Project:              Examensarbete                                        --
-- Subsystem:            TM/TC                                                --
-- Modulname:            Testbänk för demoduleringsblocket                    --
-- Description:          Genererar insignaler för att testa                   --
--                       om demodulationsblocket fungerar                     --
-- Edition:              010505                                               --
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Library IEEE;
Use IEEE.std_logic_1164.all;
Use IEEE.std_logic_arith.all;
--------------------------------------------------------------------------------
Entity test_testbench_demod is
        port(
              data_ut:                  out std_logic);
end;

Architecture ark_test_testbench_demod of test_testbench_demod is

component demod
        port(
              SignalADC:          in std_logic_vector(7 downto 0);
              clk,reset:          in std_logic;
              DataOut:            out std_logic
             );
end component;

for u1: demod use entity work.demod(ark_demod);

constant max_amplitud_data:    integer :=40;
constant max_amplitud_carrier: integer :=80;
constant min_amplitud_carrier: integer :=-80;

signal period_carrier:              std_logic :='0';
signal period_data:                 std_logic :='0';
signal reset_ut:                    std_logic :='0';
signal clk_ut:                      std_logic :='0';
signal summa:                       integer range -255 to 256;
signal AM_signal:                   std_logic_vector(7 downto 0);

begin
        clk_ut<=not clk_ut after 6410 ps;                   --78 MHz
        reset_ut<='1', '0' after 80 ns;                     --reset vid start
        period_carrier<=not period_carrier after 57 ns;     --8.75 MHz
        period_data<=not period_data after 52 us;           --19.2 kbit/s

          process(period_carrier,reset_ut)
          begin
            if reset_ut = '1' then
```

```
                summa<=0;
                AM_signal<=(others=>'0');
            elsif period_carrier'event then
                if  period_data='1' and period_carrier='1' then
                    summa<=(max_amplitud_data + max_amplitud_carrier);
                    AM_SIGNAL<=conv_std_logic_vector(summa,8);

                elsif period_data='1' and period_carrier='0' then
                    summa<=(min_amplitud_carrier - max_amplitud_data);
AM_SIGNAL<=conv_std_logic_vector(summa,8);
                elsif period_data='0' and period_carrier='1' then
                     summa<=max_amplitud_carrier;
                    AM_SIGNAL<=conv_std_logic_vector(summa,8);
                else
                     summa<=min_amplitud_carrier;
                    AM_SIGNAL<=conv_std_logic_vector(summa,8);
                end if;
            end if;
        end process;

U1: demod port map(
                    reset=>reset_ut,
                    clk=>clk_ut,
                    SignalADC=>AM_SIGNAL,
                    DataOut=>data_ut
                    );
end;
```

```
------------------------------------------------------------------------------------------
--                                                                                      --
--  Copyright Allgon Microwave AB, Göteborg, Sweden 2001                                --
--                                                                                      --
------------------------------------------------------------------------------------------
--                                                                                      --
-- Project:              Examensarbete                                                  --
-- Subsystem:            TM/TC                                                          --
-- Modulname:            Huvudprogram för AM modulator/demodulator                      --
-- Edition:              010505                                                         --
------------------------------------------------------------------------------------------

Library IEEE;
Use IEEE.std_logic_1164.all;

Entity HuvudProgram is
        port(
            P_Ut:           in integer range 0 to 255;  --parameter
            ModIndex:       in integer range 0 to 255;  --parameter
            DataIn:         in std_logic;
            clk:            in std_logic;
            reset:          in std_logic;
            enable:         in std_logic;
            A_D_Signal:     in std_logic_vector(7 downto 0);
            DataTakt:       out std_logic;
            D_A_Signal:     out std_logic_vector(7 downto 0)
            );
end;

Architecture ark_HuvudProgram of HuvudProgram is

component modulations_block
        port(
            InfoData:       in std_logic;
            SignalIn:       in std_logic_vector(7 downto 0);
            clk:            in std_logic;
            reset:          in std_logic;
            enable:         in std_logic;
            P_Ut:           in integer range 0 to 255;
            ModIndex:       in integer range 0 to 255;
            AM_SignalOut:   out std_logic_vector(7 downto 0)
            );
end component;

component demod
        port(
            SignalADC:      in std_logic_vector(7 downto 0);
            clk:            in std_logic;
            reset:          in std_logic;
            DataOut:        out std_logic
            );
end component;

begin
        gate_modulations_block:modulations_block port map(
                                                        P_Ut=>P_Ut,
```

```
                                                    ModIndex=>ModIndex,
                                                    clk=>clk,
                                                     reset=>reset,
                                                    enable=>enable,
                                                    SignalIn=>A_D_Signal,
                                                    InfoData=>DataIn,
                                                    AM_SignalOut=>D_A_Signal
                                                     );

            gate_demod:demod            port map(
                                                clk=>clk,
                                                reset=>reset,
                                                SignalADC=>A_D_Signal,
                                                DataOut=>DataTakt
                                                 );
end;
```

Växjö universitet

Matematiska och systemtekniska institutionen
SE-351 95 Växjö

tel 0470-70 80 00, fax 0470-840 04
www.msi.vxu.se