VASA UNIVERSITET

TEKNISKA FAKULTETEN

AUTOMATIONSTEKNIK

Mats Bo-Gustav Björkqvist

FPGA-PLATTFORM FÖR BILDBEHANDLING

Licentiatavhandling som överlämnats för granskning för Teknologie licentiatexamen i automationsteknik, Vasa, 8/4 2017.

Licentiatavhandlingens övervakare Jarmo Alander

Licentiatavhandlingens handledare Janne Koljonen,
Petri Välisuo

FÖRORD

Denna teknologie licentiatavhandling i automationsteknik har gjorts vid Institutionen för el- och energiteknik vid Tekniska fakulteten vid Vasa universitet. Avhandlingen framställer en flexibel FPGA-plattform för bildbehandling av bilder i realtid och still-bilder med serveregenskaper och TCP- och UDP-kommunikation emot Ethernet, Internet eller lokal nod. Ämnet för avhandlingen har varit givande och intressant och har gett mig gott utbyte med tanke på FPGA-teknologin och -tekniken och dess hjälpmedel.

Jag vill rikta ett stort tack till professor Timo Vekara, enhetschef för Institutionen för el- och energiteknik vid Tekniska fakulteten vid Vasa universitet, då jag fått möjlighet att fullfölja min TkL-avhandling vid institutionen. Vidare vill jag tacka övervakaren av avhandlingen, professor Jarmo Alander vid Tekniska fakulteten vid Vasa universitet och mentor, TkD Janne Koljonen, laboratorieingenjör samt mentor, TkD Petri Välisuo, forskardoktor vid Tekniska fakulteten vid Vasa universitet för ert stöd, era goda råd och kommentarer under avhandlingens fullföljande, 17/6 2013–8/4 2017.

Ytterligare riktar jag ett tack till FM Olli Rauhala för en snabb introduktion till Altera Monitor-programmet och övriga goda kommentarer. Jag riktar även ett tack till TkD Dag Björklund, Embedded Software Specialist vid Comsel System och FM Linda Storås för genomgång och kontroll av min svenska och stilistik. Sammalunda riktar jag ett tack till TkD Bertil Brännbacka, för allt stöd vid mekanisk implementering av systemet och initiala mättester i samband med inbyggnad av systemet till en fysisk lösning. Jag framför ett stort tack till alla övriga vid Tekniska fakulteten vid Vasa universitet för stöd och intressanta synpunkter till min TkL-avhandling i automationsteknik.

Vasa 8/4 2017.

Mats Björkqvist

INNEHÅLLSFÖRTECKNING

FIGURLISTA

TABELLISTA

SYMBOLER OCH FÖRKORTNINGAR

A/D	Analog-till-digital
ADC	Analog to Digital Converter
ALM	Adaptive Logical Module
API	Application Program Interface
ASIC	Application Specific Integrated Circuit
B	Byte
b	Bit
Bl	Blockerad
BSP	Board Support Package (implementation av en specifik stödmjukvara för ett givet system, som överensstämmer med ett givet operativ-system)
CDT	C/C++ Development Toolkit
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
D/A	Digital-till-analog
DCT	Discrete cosine transform
DDR SDRAM	Double Data Rate Synchronous Dynamic Random Access Memory
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access (IP-kärna för snabbkopiering av data)
DSP	Digital Signal Processor/Processing
DVI	Digital Visual Interface
EBl	Ej blockerad
EDS	(Nios II) Embedded Design Suite
EEPROM	Electronically Erasable Programmable Read Only Memory
FPGA	Field Programmable Gate Array
Gbps	Giga bit per second
GPIO	General Purpose I/O
M.h.a.	Med hjälp av
HAL	Hardware Abstraction Layer

HDL	Hardware Description Language
HLS	High-Level Synthesis
HSMC	High Speed Mezzanine Connector
HSTC	High Speed Terasic Connector
HW	Hardware (hårdvaran i IPS-systemet)
IDE	Integrated Design Environment
I.o.m.	I och med
I/O	Input/output
IoT	Internet of Things
IP	Internet Protocol
IPS	Image Processing Server (även kallad bildbehandlingsplattform)
ISR	Interrupt Service Routine
IVPP	Image and Video Processing Platform
kB	Kilobyte
kbps	Kilobit per second
LAN	Local Area Network
LE	Logiskt element
MAC (matematisk operation)	En hårdvaruenhet, som gör en Multiply and Accumulate-operation. Produkten av två tal (b och c) beräknas och adderas till ackumulator a. Den nya summan skrivs till ackumulator a ($a = a + (b \times c)$).
MB	Megabyte
Mbps	Megabit per second
MFHME	Multi Frame Hierarchical Motion Estimation
MGSH	Maximi genomströmningshastighet
MMI	Man-Machine Interface (användargränssnitt)
MMU	Memory Management Unit
MPU	Memory Protection Unit
MRF	Markov Random Field
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
OS	Operativsystem

OTG USB	On-The-Go USB (en anordning som kan ha både master- och slavroll)
PCB	Printed Circuit Board
PIO	Parallel Input/Output
RAM	Random Access Memory
RGB	Red Green Blue (color)
RISC	Reduced Instruction Set Computing
ROM	Read Only Memory
RTL	Register Transfer Level
RTOS	Real Time Operating System
SBT	Software Build Tools
SD	Secure Data
SoC	System on Chip
SoPC	System on Programmable Chip
SW	Software (mjukvaran i IPS-systemet)
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
VGA	Video Graphics Array
VHDL	VHSIC (Very-High-Speed Integrated Circuit) Hardware Description Language (programmeringsspråk för hårdvara)
VPP	Video Processing Platform

SVENSK-ENGELSK TERMINOLOGI

Abstraktion (vissa egenskaper utelämnas till förmån för andra egenskaper)	Abstraction
Arbetsturgivning	Arbitration
Bildpunkt (pixel används även på svenska)	Pixel
Bitström (bitstream används även på svenska)	Bitstream
Brytpunkt	Breakpoint
Bussledande, den som sköter busstrafiken (Bus-master används även på svenska)	Bus-master
Cacheminne (ett snabbt lokalt minne för processorn för att undvika inladdning av data eller instruktioner från ett ordinarie minne)	Cache memory
Chipp (integererad krets används även på svenska)	Integrated circuit/chip
Dataflöde	Dataflow
Domän (större helhet)	Domain
Dubbelkärnig	Dual-core
Exekverande flerprocessoperativsystem	Multitasking operating system
Exekvering	Execution
Exekveringsmiljö	Runtime environment
Expansionskontakt	Expansion header
Felsöka (utföra debug eller debugga används även på svenska)	Debug
Flerkort (multi-board används även på svenska)	Multi-board

Framåtmatande	Feed forward
Följa upp var mjukvara exekverats	Trace
GNU-verktygskedja (GNU är en akronym av engelskans GNU's not Unix)	GNU toolchain
Gränssnitt	Interface
HW/SW Co-Design (vedertaget begrepp, samma ord används även på svenska)	HW/SW Co-Design
Hårdvarubeskrivningsspråk	Hardware Description Language
Hårdvarukärna	Hardcore
Händelse	Event
Högnivå syntesverktyg	High-Level Synthesis Design Tool
I allmänt bruk/för allmänt ändamål	General purpose
Inbyggt system (inbäddat eller embedded system används även på svenska)	Embedded system
Intern sammankoppling	Interconnect
IP-kärna/-enhet	Intellectual Property core/IP-core/IP-unit/Qsys-component
Kompilator	Compiler
Konstruktionsflöde	Design flow
Logiskt element	Logic element
Lokalt nätverk	Local Area Network
MAC-adress	Media Access Control (address in an Ethernet frame)
Masterenhet för kommunikation	Master Communication Controller
Matris (array används även på svenska)	Array/table
Multi-CPU (samma ord används även på svenska)	Multi-CPU
Multi-thread-miljö	Mulit-thread environment
Mjukvarukärna	Softcore
Nyttodata	Payload

Nätverk-i-krets	Network-on-(a-)chip
Partitionering (uppdelning används även på svenska)	Partitioning
Pekpanel	Touch Panel
Process (en SW-process eller uppgift i ett operativsystem) (task används även på svenska)	Task
RAM-i-chipp	On-chip RAM
RAM-utanför-chipp	Off-chip RAM
Realtidsbehandling vid dataöverföring	On-line processing
Realtidsoperativsystem	Real-time operating system
Rekommenderande/refererande konstruktion/referenskonstruktion	Tutorial
Samarbetsprocessor	Co-processor
Semafor	Semaphore
Skapa ett utdrag	Extract
Slavarbetsturgivningssystem	Slave arbitration system
Stiftsplaneringssteg/-fas (i FPGA)	Pin planner phase
Suboptimal (under önskvärd nivå)	Suboptimal
Synergi (när två eller flera influenser tillsammans bildar en starkare influens än vid direkt addition)	Synergy
Syntetisera (sammankoppla eller kombinera olika ting för att skapa eller göra något (nytt))	Synthesize
System för visuell visning	Monitor system
Sändtagare (sändare och mottagare i samma krets)	Transceiver
Teckenruta	Display
Tidsfördelning	Time scheduling

Tidsgräns	Deadline
Tillståndsmaskin (state-machine används även på svenska)	State-machine
Utvecklingskort	Development Kit
Upprepning	Iteration
Vakthund styrd av en timer	Watchdog timer
Åtkomst	Access
Änd-till-änd	Point-to-point

VASA UNIVERSITET
Tekniska fakulteten
Författare: Mats Björkqvist
TkL-avhandlingens namn: FPGA-plattform för bildbehandling
Övervakare: Professor Jarmo Alander
Handledare: TkD Janne Koljonen, laboratorieingenjör
TkD Petri Välisuo, forskardoktor
Examen: Teknologie licentiat
Ämne: Automationsteknik
Startår för studierna: 2013
TkL-avhandlingens examensår: 2017 **Antalet sidor:** 124

ABSTRAKT:

I denna licentiatavhandling i automationsteknik planeras, förverkligas och testas en FPGA-plattform för bildbehandling, som fungerar som en bildbehandlingsserver på Ethernet och Internet. Plattformen kan utföra ett stort antal databehandlingsmetoder och -tillämpningar inom höghastighetskommunikation i realtid. Med hjälp av Altera- och Eclipse-verktygen, Terasics ALTERA DE3-utvecklingskort med Alteras Stratix III FPGA och HSMC-NET- och minneskort och VHDL-, Verilog-, C- och Assembler-programmeringsspråket skapas en 1 Gbps FPGA-plattform för bildbehandling.

Vidare behandlas till lösningen hörande begrepp, som en FPGA-plattform som ett inbyggt system, orsak till val av FPGA-hårdvara och förväntningarna på ett utvecklingskort. Viktiga verktyg, hjälpmedel och komponenter vid konstruktion av en bildbehandlingsplattform samt möjligheter för pipeline och parallellism kartläggs. Konstruktions- och implementeringsmetoder vid planering och konstruktion av hårdvara och mjukvara presenteras speciellt gränssnitt mellan hårdvara och mjukvara och deras verktygs roller i ett HW/SW Co-Design-system. Implementering av hårdvara och mjukvara, hårdvaran, moder-, dotter- och minneskortet med sammankopplingar och implementerade funktioner beskrivs. Mjukvaran beskrivs med implementerade mjukvarufunktionsgrupper såsom system start-up-, operativsystem-, bildbehandlings- och avbrottsrutiner.

Det utfördes manuella och prestandatester med plattformen. De manuella TCP- och UDP-testerna visar att alla kommandon och operationer fungerar korrekt i alla lager och på alla nivåer. Prestandatesterna visar att plattformen kan hantera både låg- och högbelastande TCP- och UDP-trafik med stigande och sjunkande längd på testdata. Alla tester visar samma struktur och trend för genomströmning. Maximigenomströmningen för plattformen är ca 7,5 Mbps med en Nios II/f-processor och arbetsfrekvens på 50 MHz.

Mitt bidrag har varit att bygga en mer omfattande funktionell mjukvara med hjälp av basprogramvara samt att bygga en omfattande funktionell hårdvara i IPS. Dessutom att bygga en omfattande funktionell testprogramvara för PC – alla med nödvändiga funktioner och komponenter.

NYCKELORD: FPGA, Bildbehandling, SoPC, 1 Gbps Ethernet.

VAASAN YLIOPISTO
Teknillinen tiedekunta
Tekijä: Mats Björkqvist
TkL-tutkielman nimi: FPGA-kuvankäsittelyalusta
Valvojan nimi: Professori Jarmo Alander
Ohjaajien nimet: TkT Janne Koljonen, laboratorioinsinööri
TkT Petri Välisuo, tutkijatohtori
Tutkinto: Tekniikan lisensiaatti
Oppiaine: Automaatiotekniikka
Opintojen aloitusvuosi: 2013
TkL-tutkielman valmistumisvuosi: 2017 **Sivumäärä:** 124

TIIVISTELMÄ:

Tässä automaatiotekniikan lisensiaattitutkielmassa suunnitellaan, toteutetaan ja testataan kuvankäsittelyyn tarkoitettu FPGA-alusta, joka toimii kuvankäsittelypalvelimena Ethernetissä ja Internetissä. Alustalla voidaan suorittaa kuvankäsittelymenetelmiä ajantasaisesti. Tutkielmassa esitetään Altera- ja Eclipse-työkalujen, Alteran Stratix III FPGA-piirin Terasicin ALTERA DE3-kehitys-, HSMC-NET- ja muistikortin ja VHDL-, Verilog-, C- ja Assembler-ohjelmointikielen avulla 1 Gbps FPGA-kuvankäsittelyalustan ratkaisu.

Lisäksi käsitellään ratkaisuun liittyviä käsitteitä kuten FPGA-alusta sulautettuna järjestelmänä, FPGA-laitteistoratkaisun edellytykset ja kohdekehitysalustan valintakriteerit. Kartoitetaan tärkeät työkalut, apuvälineet ja komponentit kuvankäsittelyalustan suunnittelussa ja toteuttamisessa sekä liukuhihna- ja rinnakkaislaskennan mahdollisuudet. Esitellään suunnittelu- ja toteutusmenetelmiä laitteiston ja ohjelmiston suunnittelussa ja luomisessa erityisesti rajapinta ohjelmiston ja laitteiston välillä sekä työkalujen roolit HW/SW Co-Design-järjestelmässä. Kuvataan laitteiston ja ohjelmiston toteutusta. Esitellään laitteisto, emo-, tytär- ja muistikortin yhteenkytkennät ja toteutetut funktiot. Ohjelmistosta kuvataan toteutetut funktioryhmät kuten käynnistys-, käyttöjärjestelmä-, kuvankäsittely- ja keskeytysrutiinit.

Alustan kanssa suoritettiin sekä manuaali- että suorituskykytestejä. Manuaaliset TCP- ja UDP-testit osoittavat, että kaikki komennot ja toiminnot toimivat oikein ja asianmukaisesti kaikissa kerroksissa ja kaikilla tasoilla. Suorituskykytestit osoittavat, että alusta kykenee käsittelemään TCP- ja UDP-liikennettä matalalla ja korkealla kuormituksella sekä nouseva- että laskevapituisilla testisanomilla. Alustan maksimiläpivirtaus on noin 7,5 Mbps Nios II/f-prosessorilla ja toimintataajuudella 50 MHz.

Minun panokseni on ollut rakentaa kattavamman toiminnallisen ohjelmiston käyttäen pohjaohjelmiston sekä rakentaa kattavan toiminnallisen laitteiston IPS:ään. Lisäksi rakentaa kattavan toiminnallisen testiohjelmiston PC:hen – kaikilla tarpeellisilla toiminnoilla ja komponenteilla.

AVAINSANAT: FPGA, Kuvankäsittely, SoPC, 1 Gbps Ethernet.

UNIVERSITY OF VAASA
Faculty of technology
Author: Mats Björkqvist
Topic of Lic. Sc. (Tech.) Thesis: FPGA Image Processing Platform
Supervisor: Professor Jarmo Alander
Instructors: D.Sc.(Tech.) Janne Koljonen, Laboratory Engineer
D.Sc.(Tech.) Petri Välisuo, Assistant Professor
Degree: Licentiate of Science (Technology)
Major of Subject: Automation Technology
Year of starting studies: 2013
Year of completing the Thesis: 2017 **Pages:** 124

ABSTRACT:

In this Licentiate of Science thesis in Automation Technology is planned, implemented and tested an FPGA image processing platform that works as an image processing server connected to Ethernet and Internet. The platform can provide image processing methods and applications with high-speed communication in real time. A 1 Gbps FPGA image processing platform is built up with help of Altera and Eclipse tools, Terasic's ALTERA DE3 development kit including Altera's Stratix III FPGA, HSMC-NET and memory board and VHDL, Verilog, C and Assembly programming languages.

Furthermore, is examined concepts such as FPGA platform as an embedded system, the premises for choosing an FPGA hardware solution and the expectations of a development kit. A survey of important relevant tools, utilities and components in a design of an Image Processing Platform and possibilities with pipeline and parallelism is conducted. Design and implementation methods in planning and creation of a hardware and software are presented especially interfaces between hardware and software and roles of the tools in a HW/SW Co-Design system. Implementation of hardware and software, hardware, mother and daughter boards and memory card with their interconnections and functions are presented. Software is presented with implemented software function groups such as system start-up, operating system, image processing and interrupt routines.

Manual and throughput TCP and UDP tests were performed with the platform. Manual tests show that all commands work properly in and at all layers and levels as expected. Throughput tests show that the platform is able to handle both low and high loaded communication in rising and falling length of test data. The maximum throughput of the platform is about 7.5 Mbps with a Nios II/f processor and executing frequency at 50 MHz. All tests show the same structure and trend of throughput.

My contribution has been to build a more comprehensive functional software using a base software, as well as to build a comprehensive functional hardware into IPS. In addition, to build a comprehensive functional test software into PC – all with necessary functions and components.

KEYWORDS: FPGA, Image Processing, SoPC, 1 Gbps Ethernet.

1 INLEDNING

Idag finns behov av att utveckla FPGA-plattformar inom forskning att skapa nya kombinationer av metodik för bättre och snabbare bildbehandling. Målet med en dylik plattform är att ha goda möjligheter att testa och utveckla framtagna nya bildbehandlingsmetoder och kombinationer av dessa i FPGA-omgivning för bildbehandling.

Plattformen skall arbeta emot Internet med protokollen TCP och UDP, ha snabb busskommunikation och god konfigurerbarhet, bra designverktyg och god testbarhet för olika bildbehandlingsmetoder. Plattformen måste vara användar- och utvecklingsvänlig och förmånlig (även vid prototypframställning), ha god programmerbarhet av hårdvara och mjukvara där nya bildbehandlingsmetoder kan testas men även senare vara passande i undervisningsmiljö.

I denna avhandling utvecklas en plattform där man inte hänvisas till färdiga system på marknaden men ändå uppfyller utvecklarens och/eller användarens önskemål. Det visas metodik hur man med Terasics ALTERA DE3-utvecklingskort, Alteras Stratix III FPGA, HSMC-NET-kort och Alteras utvecklingsverktyg skapar en plattform för bildbehandling med serveregenskaper, en Image Processing Server (IPS). Den kan tillämpas vid behandling av bilder och video via Ethernet. Utvecklingskortet erbjuder lämplig prestanda och storlek på hårdvara i och med Stratix III FPGA. Plattformen skall även kunna arbeta som datormoln, beräknarserver, systemanalysator och vara mjukvarusensor för IoT-noder på Internet och därmed underlätta deras behov av tunga beräkningar och analyser. Avhandlingen berör vetenskapsområdena datateknik, signalbehandling, datakommunikation och elektronik.

Detta system kan även erbjuda förstärkning i undervisning på FPGA-nivå vid universitet med tanke på skapande, utvecklande, påbyggande och kompletterande av hårdvara i FPGA-system. Vidare, med tanke på mjukvara, att fördjupa sig i användning av operativsystem, meddelandehantering, sedvanlig mjukvara, drivrutiner, skapande och skötsel av avbrottssystem och utvecklande och kompletteringar av de samma men även slutlig-

en att skapa innovation inom och influens och inflytande på och av FPGA-system och -teknologi.

Plattformen med hårdvara och mjukvara har Micriums realtidsoperativsystem, µC/OS-II (även kallat MicroC/OS-II), som är enkelt att implementera med lämpligt antal taskar, erbjuder enkel meddelandehantering mellan taskarna och där man också kan prioritets-hantera dessa. Utvecklingskortet ger en snabb lösning till större system till ett relativt lågt pris. IPS erbjuder möjlighet för behandling av t.ex. bilddata eller annan data via TCP och UDP på Ethernet-bussen.

I mjukvaran finns kommandotolkar för både Ethernet- och JTAG-buss för att styra olika skeenden i IPS. Det finns två snabba DMA-kanaler, som sköter in- och utförsel av data från och till respektive krets på det externa Ethernet-kortet. I denna lösning har även ett mät- och styrgränssnitt öppnats genom en GPIO på expansionskontakt i utvecklingskor-tet varifrån och vart man kan koppla signaler för mätning respektive styrning av kortet. Ytterligare passar detta utvecklingskort för flerbords-, server-, Ethernet- och processor-lösningar.

En FPGA-plattformbaserad IPS betyder att hårdvara och mjukvara implementeras i en FPGA-krets med dimensioner som att en viss del av SW kan fullföljas i och lika snabbt som HW men även att sedvanlig mjukvara exekveras av en implementerad processor i systemet. Genom att FPGA-teknik används kan snabba uppdateringar (under 30 min.) av SW och HW göras. Fördelar med FPGA är att dessa inkluderar möjligheten för snabba omprogrammeringar på fältet, kompletteringar, förändringar och -bättringar och därmed erhålls en kortare tid till marknaden-faktor och man håller nere återkommande tekniska kostnader.

1.1 Implementering med FPGA

Utvecklingskortet ALTERA DE3 (figur 1) erbjuder en snabb metod att fullfölja bildbe-handling då detta innehar den viktiga komponenten FPGA.

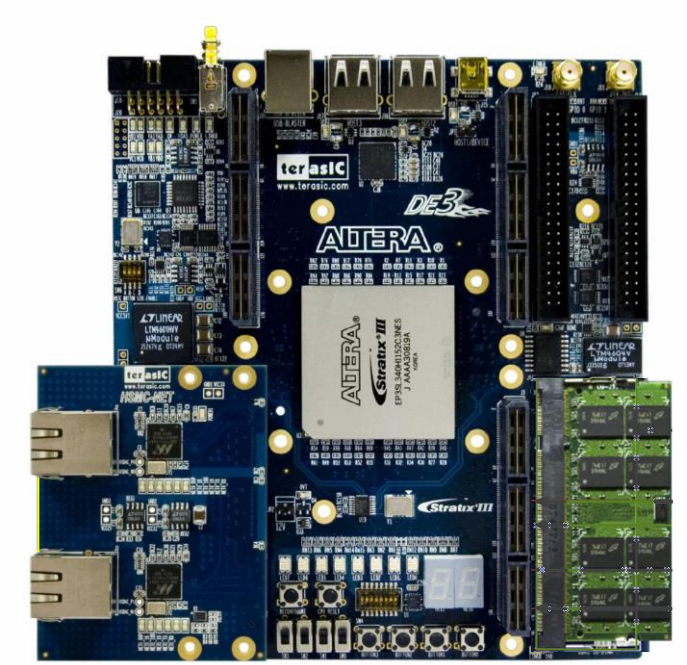


Fig. 1. Terasics ALTERA DE3-utvecklingskort med Alteras Stratix III FPGA. HSMC-NET-dotter-, ALTERA DE3-utvecklings- och 1 GB minnestilläggskortet sammankopplade. Anpassad från (Terasic, 2010) och (KAMAMI, 2009) och kompletterad av författaren.

Utvecklingskortet erbjuder t.ex. laddning av hårdvara och mjukvara, har en FPGA med goda arbetsverktyg och uppfyller de krav på HW och SW som ett dylikt system behöver för och med Internet-kommunikation och operativsystem.

Det innehar möjlighet för anslutning av ett 1 Gbps Ethernet-gränssnitt och externa mät-, test- och monitoreringsinstrument då man har behov av testgränssnitt för kontroll av systemet och uppföljning av kommunikationen till yttre system.

I figur 2 ges huvudkomponenterna i ett koncept för en FPGA-plattform och vad ett dylikt system bör inkludera för att kunna fullföljas med testning och utveckling av huvud- och underdelar. Värddatorn använder en programvara för kommunikation via ett 1 Gbps Ethernet-kort. I utvecklingskortet implementeras hårdvara och mjukvara, som bildar en bildbehandlingsplattform tillsammans med ett dotterkort med Ethernet-gränssnitt.

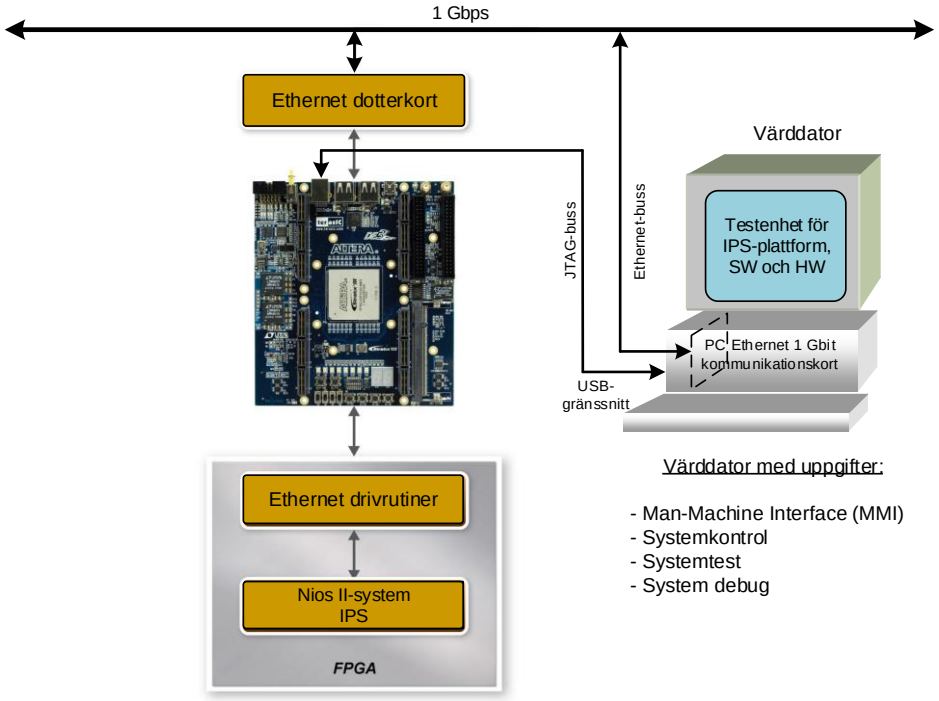


Fig. 2. Översikt över ett passande logiskt koncept för huvud- och delkomponenter i ett system till en FPGA-plattform för bildbehandling och testgränssnitt. Anpassad från (Terasic, 2010) och kompletterad av författaren.

1.2 Forskningsmetod

Forskningsmetoden i avhandlingen är den hypotetisk-deduktiva metoden. För fördjupning i denna metod hänvisas läsaren till Molander (1998). Sammanfattningsvis är huvud- och understeg som exempelvis:

- Inledande fas bestående av *problemet*
- Utvecklingsfas (tillämpad med konstruktiv empirisk forskning) bestående av iterationer av underfaserna *hypotes*, *tester* och *falsifieringar*
- Avslutande fas bestående av *fastställning av en färdig lösning* och *avslutning*

För att styra forskningen i avhandlingen finns ytterligare grundlitteratur, litteratur för arbetsredskap, Internet, litteratur för målobjektet och tutorial till handa. (KTH, 2008)

1.3 Avhandlingens uppbyggnad

Teknologie licentiatavhandlingen består av kapitel ett, som innefattar en inledning och vad forskningsproblemet, utgångspunkterna, bakgrunden, målet och utmaningarna är för avhandlingen. I kapitel två visas tidigare forskning, som gjorts inom liknande system. Kapitel tre behandlar aktuella begrepp och verktyg, som behövs för att bygga upp en bildbehandlingsplattform i avhandlingen. Kapitel fyra beskriver viktiga faktorer i och angående uppbyggnaden av FPGA-bildbehandlingsplattformen. I kapitel fem konkretiseras och implementeras hårdvaran och mjukvaran i plattformen med kommunikationskanaler och bildbehandlingsfunktioner och här beskrivs och visas hur plattformen är uppbyggd. I kapitel sex beskrivs monitoreringsverktyg, fullgörs tester och testresultat presenteras. I kapitel sju avslutas avhandlingen med sammanfattning, kommentarer, synpunkter och slutsats. I avhandlingen används kursiv stil för att betona speciella eller viktiga systemfunktioner, egenskaper och attribut hos systemet men även beskrivande ord och uttryck för systemet. I början av avhandlingen finns inkluderat en ordlista över svensk-engelsk terminologi.

2 BILD- OCH VIDEOPLATTFORMAR

I detta kapitel följer en genomgång av tidigare forskning och utveckling av bildbehandlings- och videobearbetningssystemen.

2.1 Lösningar med FPGA och inbyggd processor

Finc, Trost, Zajc, och Žemva (2003) presenterar en modulär och anpassningsbar plattform för videobearbetning och bildbehandling i realtid. Plattformen bygger på FPGA-teknik och en RISC-processor för databehandling. Detta minskar tillämpningens konstruktionssteg och förkortar konstruktionsiterationer särskilt med tanke på de senare konstruktionsstegen. Arkitekturen hos plattformen kan delas in i två huvudsakliga funktionella domäner:

- kommunikation (videokortet)
- databehandling (databehandlingsmoduler).

Videobearbetningsplattformen, presenterad i figur 3, består av utvecklingskortet (Video board), som hanterar utbytet av uppgifter mellan de olika oberoende databehandlingsmodulerna (Input module, Output module och Nios Development Board).

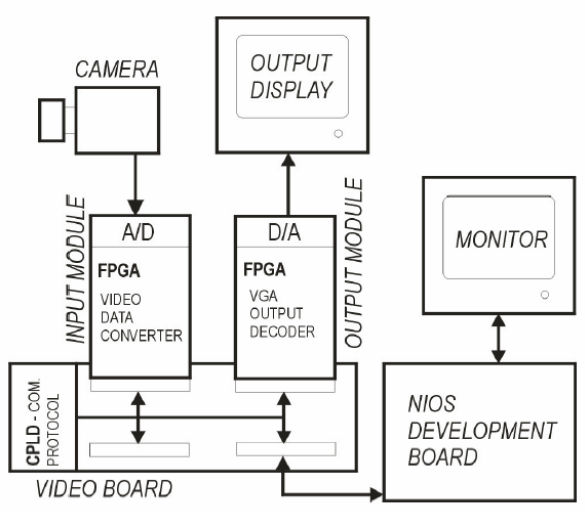


Fig. 3. Videobearbetningsplattform med kamera, VGA-monitor och Nios-utvecklingskort med monitor. (Finc, Trost, Zajc, & Žemva, 2003)

Överföringen av uppgifterna hanteras och sköts av en masterenhet för kommunikation, genomförd med en CPLD. Modulerna kommunicerar via en delad buss med ett speciellt dataprotokoll. I protokollet hanterar den modul, som handhar registren dataöverföringen mellan modulerna. Dataöverföringen är enkelriktad. Genom god konfigurerbarhet, parametrisering och ett integrerat designverktyg i kombination med flexibel hårdvaru- och mjukvaruintegration uppnås ett vidare spektra av realtidsbildbehandlings- och videobearbetningsapplikationer. För demonstration har man genomfört två detektionsalgoritmer av en rörelse. Enligt uppnådda resultat är differentialalgoritmer lämpliga för enkla praktiska applikationer för rörelsedetektion. Man framhåller att Markov Random Field (MRF) algoritmen (Lacassagne, Milgram, & Garda, 1999) konkurrerar framgångsrikt med andra experimentella system vilka mestadels är DSP- eller multi-CPU baserade system eller system som exekveras på mycket höga frekvenser. Avslutningsvis konstaterar författarna att, baserat på konfigurerbar teknik (CPLD, FPGA, CPU-mjukvarukärna), är deras prototypplattform för bildbehandling utformad för att stödja *HW/SW Co-Design* och partitionering.

Det finns även s.k. bildbehandlings- och videobearbetningsplattformar (IVPP), baserade på FPGA-teknik. (Desmouliers, Oruklu, Aslan, Saniie, & Vallina, 2012, ss. 414–425) Plattformen visas i figur 4. Denna plattform tillämpar *HW/SW Co-Design* och är uppbyggd i en SoC FPGA (Xilinx Virtex-5) med högnivå syntes. Med denna plattform kan man förverkliga och testa komplexa algoritmer för bildbehandling och videobearbetning i realtid. Videogränssnittet är förverkligat på RTL-nivå (Xilinx, 2012) och kan konfigureras med en MicroBlaze-processor för olika videoupplösningar. Denna version tar in videodata från en kamera med VGA-ingång med en upplösning på 1024x768 vid 60 Hz. Videodata buffras upp i minnet och genom en flexibel arkitektur kan användaren utföra realtidshantering på en enskild bildruta eller flera bildrutor vilken/vilka senare visas upp på skärmen via en DVI-utgång.

Behandlingsalternativen är:

- Visa färgvideo utan bearbetning
- Utföra realtidsbearbetning på en enda ram för RGB-videodata
- Visa utdata och/eller utföra flerbildsbehandling och visa utdata

Här används High-Level Synthesis (HLS)-design konstruktionsflöde där användaren kan utforma bildbehandlings- och videobearbetningsprogram i C, omvandla dessa till maskinvara med Symphony C-kompilator och sedan implementera och testa dessa enkelt med hjälp av IVPP.

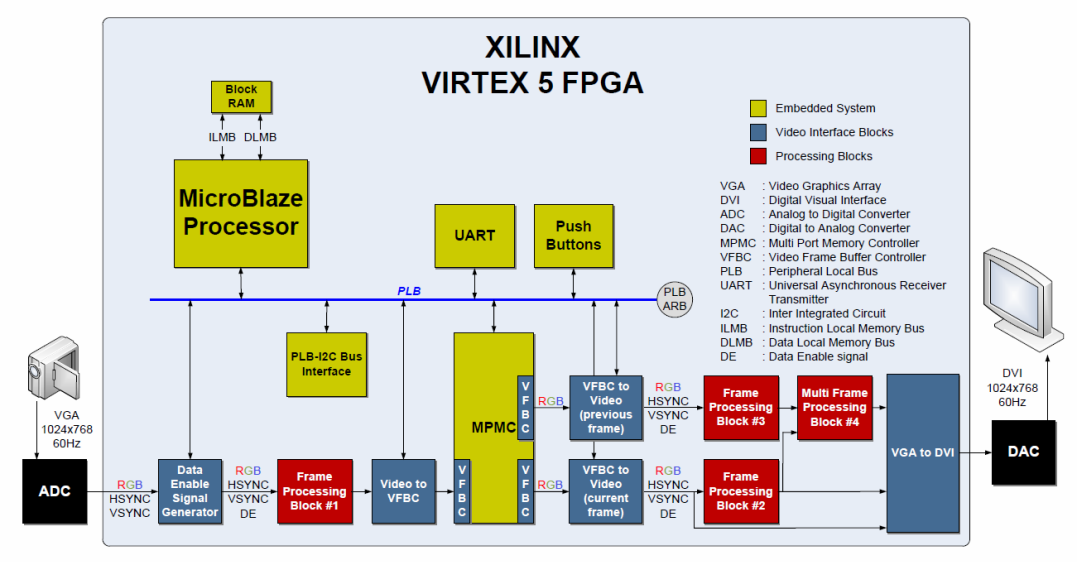


Fig. 4. Översikt över bildbehandlings- och videobearbetningsplattformen. (Desmouliers, Oruklu, Aslan, Saniie, & Vallina, 2012)

En FPGA-baserad implementering av en videobearbetningsplattform visas i en konferensartikel (Desmouliers, Aslan, Oruklu, & Saniie, 2009). Systemet är uppbyggt i en FPGA (Xilinx Virtex II Pro). Här tillämpas *HW/SW Co-Design* vid uppbyggandet av hårdvara och mjukvara och man realiserar komplexa algoritmer för realtidsbildbehandling och -videobearbetning. Artikeln presenterar ett upplägg för VPP och man diskuterar de arkitektoniska byggstenarna och FPGA-syntesresultaten. Varje maskinvara (ett anpassat acceleratorblock) och programvarukomponent exekveras av en inbyggd CPU-kärna i ett flexibelt och modulärt inbyggt system. I en studie görs en rörelsedetekteringsalgoritm i realtid och man visar genomförbarheten av en dylik plattform. Man kan koppla maskinvaruacceleratorer in till systemet med önskade tillståndsmaskiner. VPP kan vara en robust lösning för ett brett utbud av multimedia applikationer men även för sändande och genomströmmande video-, videokodnings- eller avkodnings-, övervaknings-, detekterings- och identifieringstillämpningar. För VPP används utvecklingskortet Virtex-II Pro (Xilinx, 2014a) med FPGA, två PowerPC-processorer och DDR

SDRAM DIMM-modul, som kan inneha 2 GB RAM. En expansionskontakt är tillgänglig för att förse utvecklingskortet med en VDEC1-videodekoder.

Terasic Technologies presenterar i sin användarmanual för tPad en intressant lösning för ett inbyggt system. (tPAD, 2010) Med användning av ett tPad Embedded Evaluation Kit erbjuds en omfattande designmiljö för inbyggda system. TPad erbjuder en integrerad plattform, som omfattar verktyg för design, referensdesign för utveckling av inbyggd programvara och hårdvaruplattform med brett utbud av program. TPad finns förkonfigurerad med FPGA-hårdvarureferensdesign, som inkluderar flera färdig-att-köra mjukvaruapplikationer, som en utvecklare behöver för att snabbt utveckla och bygga upp komplexa inbyggda system. Denna lösning ger en inbyggd allt-i-allo lösning med robust plattformsbas för multimediaprogram i FPGA.

Kretstillämpning av hierarkisk flerbildsrörelseberäkning för högupplöst (HD) rörelseestimering, s.k. Multi Frame Hierarchical Motion Estimation (MFHME)-kretslösningar finns även som FPGA-konstruktioner och implementeringar. (Ho, Klepko, Ninh, & Wang, 2011) Målet för denna tillämpning är en högkvalitets- och rörelsekompenserad FPGA-accelerator med videobildhastighet, som kräver många rörelsebilder (MF) och exakta rörelsebanor. För att uppnå låg beräkningskomplexitet, har kretsen en hierarkisk struktur och en förberäknad uppslagstabell för att beräkna kvadrerade skillnader på pixel. Resultatet av genomförandet visar att kretsen kan stöda bildhastighetsaccelerering av högupplöst High Definition (HD)-video av 1080p-format med 30-60 bilder per sekund med en klockfrekvens på 55 MHz. I tillämpningen används pipeline och återanvändning av data för att åstadkomma en krets med hög genomströmningshastighet. MFHME-kretsen är lämplig för HD och högprestanda videoapplikationer i realtid.

Desmouliers, Aslan, Oruklu, Saniie och Martinez (2010) beskriver en FPGA bildbehandlings- och videobearbetningsplattform (IVPP) skapad med *HW/SW Co-Design* och PICO-teknik. Ett PICO-verktyg från Synfora (BDTi, 2010) används, som accepterar C eller C++ källkod. Källkoden kompileras till en hårdvaruimplementering i RTL-format för FPGA- eller ASIC-implementering. Hårdvara och mjukvara utgör en plattform, som är genomförd i en FPGA (Xilinx Virtex-5). Med finns en MicroBlaze-processor, som

möjliggör olika videoupplösningar. Flera videotillämpningar av program visas bl.a. rörelsedetektor och objektssökning med IVPP för realtidsvideobearbetning.

2.2 Lösningar med FPGA

Det finns även en lösning för FPGA-tillämpning för hög upplösning. (Bowen & Bouganis, 2008). Här kombineras en uppsättning av överlappande bilder med låg upplösning för att till slut skapa en enda bild med högupplösning. Lösningen baserar sig på ett realtidssystem, som bygger på en vägd medelvärdesalgoritm kombinerad med en snabb flerbildssuperupplösnings- och resolutionsalgoritm. Algoritmens krav skalas linjärt mot målbildskvalitén vilket gör algoritmen, enligt författarna, idealiskt för olika realtidsapplikationer som t.ex. HDTV. Simuleringsresultaten visar en hastighetsökning med tre gånger under optimerad programvaruimplementering med en försumbar förlust för bildkvalitén. Arkitekturen fordrar pipeline och har utformats och byggts för att kunna mata ut 61 bilder/s med en bildstorlek på 1280x720 bildpunkter.

I ett examensarbete (Nelson, 2000) implementeras bildbehandlingsalgoritmer där FPGA-teknik används. Författaren konstaterar att FPGA-teknik har blivit ett betydande mål vid genomförandet av algoritmer, som passar videotillämpningar och bildbehandling. Läsaren hänvisas till referenser som (Chou, Mohanakrishnan, & Evans, 1993) och (Benedetti & Perona, 1998). Examensarbetet visar att FPGA-teknik är idealisk för höghastighets- och fönstersystem-algoritmer. En nackdel med presenterad teknik är bl.a. storleken på algoritmerna (se kapitel IV). Om ett RAM utanför (FPGA-)kretsen används för FIFO kan de syntetiserade mönsterstorlekarna minskas betydligt. Designen som presenteras utnyttjar parallellism med FPGA-teknik.

2.3 Lösningar med FPGA och DSP

I kategorin FPGA med DSP finns en lösning där FPGA används som en co-processor till en DSP. (Junaid & Ravindrann, 2007) Detta arrangemang och denna lösning ökar summahastigheten av databehandlingen och minskar strömförbrukningen i systemet. I

lösningen exekveras FPGA parallellt med DSP. En implementering med en ASIC för en filteralgoritm skulle medföra talrika MAC för att alla steg i filtret skall kunna behandlas parallellt. Men i och med att FPGA erbjuder en flexibel arkitektur kan alla MAC-operationer exekveras parallellt. Den programmerbara logiken kombineras lätt för allmänt ändamål med en DSP. I detta projekt används en FPGA (Spartan 2E) med ett gränssnitt mot Texas Instruments TMS320C6711 DSP. Hastigheten förbättras drygt 450 gånger jämfört med en konventionell DSP genom bildkomprimering med hjälp av DCT-teknik.

Gorgon (1997) beskriver en konstruktion av en styrenhet använd för styrning av en Retina bildbehandlingsplattform. Retina-plattformen är gjord för bildinsamling, bearbetning och analys. Modulen inkluderar en Video ADC, Virtex FPGA enhet, Motorola DSP och ett PCI Master-gränssnitt, som möjliggör exekvering av alla operationer i realtid. FPGA-kontrollenheten sköter data, som går mellan de olika enheterna och minnena, arbetsturgivningen mellan dessa och uppgifter till och från kringutrustning. FPGA-kontrollenheten använder sig av registerkonfiguration, som har information om prioriteringar och semaforer styr arbetsfördelningen mellan de enskilda modulerna. Med flexibla integrerade utvecklingsmiljöer (IDE) snabbas bildbehandlingsalgoritmernas konstruktion och genomförande upp. Xilinx och Motorolas DSP är primära komponenter i Retina-plattformen (Gorgon & Pryzybylo, 2001).

Gorgon (2012) beskriver faktorer, som man skall notera vid val av bildbehandlingssystem då man överväger fördelar och nackdelar med varje teknik. Dessa är ordinarie processor, digital signalprocessor, grafisk processor, ASIC och FPGA. I artikeln tas upp olika standarder för videoöverföring och forsknings- och utvecklingstendenser inom FPGA-baserad bildbehandling presenteras kortfattat. Tidigare DSP-baserade signalbildbehandlingssystem och digitala bildbehandlingssystem med Harvard-arkitektur (ARM, 2008) och tilläggs-MAC har varit attraktiva för beräkningar av digitala filter och transformeringar. Men i.o.m. att behov för parallella beräkningar finns är DSP inte längre ett alternativ. Superdatortillverkarna uppmärksammar mera möjligheten med FPGA då FPGA-acceleratorerna strukturellt är anpassade till superdatorstandarder och kan göra omfattande beräkningar och analys för bildbehandling. På grund av FPGA:s större buss-

kapacitet uppnås en hög överföringshastighet. FPGA-acceleratorer kan användas i system med betydande bearbetningsprestanda. Realtidsbehandling med hög datahastighet på bildström utan förlust av pixel kräver stor bearbetningsprestanda med särskilda algoritmer. FPGA fullgör detta på pixelströmmar i realtid under dataöverföringen. I artikeln presenteras en metod att beräkna bearbetningsprestanda vid parallella bildbehandlingsoperationer. En analys gjordes av hastigheten och effektiviteten vid pipeline och man kan påvisa att effektiviteten (E) är 1 genomförd i FPGA-arkitektur, parallellt och med linjär hastighetsökning.

2.4 Jämförelse mellan andra motsvarande system och vårt system

Här görs en jämförelse mellan de genomgångna systemen och framförs vad vår bildbehandlingsplattform erbjuder, som de kartlagda systemen inte erbjuder. De fördelar bildbehandlingsplattformen erbjuder gås igenom med undantag av de egenskaper utvecklingskortet och lösningen ”tPad Embedded Evaluation Kit DE2-115 with LCD Touch Panel and Camera” erbjuder. (tPAD, 2010) Denna tas upp sist i jämförelsen**.**

Om man ser på de övriga systemens lösningar är dessa enbart lokala bildbehandlingslösningar, s.k. korta lösningar av typen en-till-en-punkts koppling eller endast behandlingssystem av envägs genomströmmande bilddata. Denna lösning är, inkluderat själva behandlingsystemet, även en serverlösning. I och med detta inkluderar denna lösning allt vad en sedvanlig server erbjuder. Servern, kompletterad med bildbehandling, arbetar emot hela Internet eller lokal nod för inkommande data. Servern bildbehandlar detta och skickar sedan data tillbaka ut i Internet-rymden eller till lokal nod. Detta system erbjuder datatransmission med protokollen TCP och UDP i realtid av nyttodata och bildbehandlad data var som helst på Internet.

I och med att bildbehandlingsplattformens hårdvara och mjukvara använder Alteras bibliotek för hårdvara och mjukvara kan alla dessa användas inkluderat alla hjälpmedel för debuggning, simulering och kompilering av C- och Assembler-filer, färdiga funktioner, procedurer och mjukvarupaket men även VHDL- och Verilog-filer, färdiga hel- eller delinstansieringar eller hårdvarupaket.

Om man undersöker de övriga systemen finns inget omnämnt i artiklarna om att dessa har ett operativsystem för mjukvaran. Detta system erbjuder Micriums µC/OS-II, som är ett realtidsoperativsystem som möjliggör att olika mjukvara kan köras samtidigt. Operativsystemet erbjuder även signalering mellan olika taskar, semafor- och händelsehantering o.s.v. Ytterligare kan olika taskar sättas i olika prioritet för att kunna ge förtur åt vissa taskar. Vidare om man undersöker de övriga systemen finns inget omnämnt om trafiken utåt utan denna sker endast i en änd-till-änd lösning eller är genomgående (utan protokoll).

Detta system erbjuder TCP- och UDP-trafik på Ethernet-buss och en teoretisk takhastighet på 1 Gbps eller ca 118 Mpixel/s av 8-bitars pixeldata. Såvida storlek på HW eller exekvering av SW inte begränsar eller inga parametrar eller nya data måste mellanändras eller -laddas under beräkningarna kan man beräkna 200 Mpixel/s. Man kan dock nå högre beräkningshastigheter internt. Använder man pipeline kan man under *full pipeline*-läge t.ex. linjärt beräkna 400 Mpixel/s (med 50 MHz på HW-klockpulsen) såvida man använder 32-bitars databuss in och ut från bildbehandlingsenheten och i denna byggt upp en 4x4 bytes pipeline.

Angående hur långa funktionerna får vara har inte omnämnts i artiklarna. Denna lösning erbjuder en modell, som omfattar premissen att man kan t.ex. beräkna hur många bildbehandlingsoperationer som helst såvida inte storlek på HW begränsar. Inget omnämns heller om ett byte av bildbehandlingsmetod under själva bildbehandlingen. Denna lösning erbjuder byte av bildbehandlingsmetod med en precision på en pixels (8-bitars data) eller fyra pixels (32-bitars data i en kvartett) mellan olika bildbehandlingsmetoder. Man kan ändra och styra bildbehandlingsmetod via insignaler eller kommandon till bildbehandlingsenheten med kommando via JTAG-bussen eller på basis av inkommande data. Ytterligare kan man avläsa status av bildbehandlingsprocessen från bildbehandlingsenheten. Inget finns omnämnt om DMA-kanaler, Timer eller (G)PIO vilka erbjuds i denna lösning, som ger snabba oberoende överföringsfunktioner, precisa övervakningsmöjligheter och möjlighet till t.ex. mätning respektive HW-felsökning av funktioner i systemet.

En intressant lösning finns i artikeln, rörande Retina-plattformen, där man kan skapa hårdvara med hjälp av C-script-källkod. Dock måste man komma ihåg att detta sätt att bygga hårdvara endast är anpassad för sin egen omgivning. En sådan lösning kan inte användas med andra plattformar. Lösning med IPS erbjuder användning med VHDL och Verilog vilka resulterar i att hårdvaran kan användas i många omgivningar men även med tanke på mjukvaran i plattformen, som har fördelen att denna använder Hardware Abstraction Layer (HAL) och Application Program Interface (API). Därmed är denna oberoende av förändringar i den underliggande hårdvaran.

Angående lösningen ”tPad Embedded Evaluation Kit DE2-115 with LCD Touch Panel and Camera” (tPAD, 2010) står denna närmast vad denna plattform kan erbjuda. Men p.g.a. dess mindre kapacitet hos FPGA kan nämnda kort inte komma upp i den storlek på hårdvara, som denna lösning erbjuder. Går vi ytterligare ett steg längre fram så framkommer att Terasics ALTERA DE3-utvecklingskort kan byggas upp till en flerkortslösning. Med flera kort sammankopplade kommer all service denna lösning erbjuder att kunna utökas, kompletteras eller t.o.m. göras dynamisk.

3 HÅRD- OCH MJUKVARA

Ett inbyggt system, även kallat inbäddat eller embedded system på svenska, är ett mindre avklätt datorsystem, som ofta har en självständig roll med eller utan medlemskap i ett större system. Det inbyggda systemet har ett begränsat eller ett fåtal speciella funktioner tillägnat det speciella system det betjänar. I detta kapitel genomgås hård- och mjukvara, som är anknutna till forskningen.

3.1 FPGA-plattformen som ett inbyggt system

Ett inbyggt system kan vara en del av en apparat eller maskin inkluderat hårdvarukopplade eller styrande funktioner i det unika systemet. Bildbehandlingsplattformen klassificeras som ett inbyggt system i.o.m. att denna har en speciell uppgift i det att den utför bildbehandling emot Internet, lokala nätverk (LAN) eller i en änd-till-änd koppling. I IPS ligger mjukvaran och hårdvaran dold i jämförelse med ordinarie system, t.ex. i en PC.

3.1.1 Ett inbyggt systems kännetecken, egenskaper och uppgifter

Bildbehandlingsplattformen utför speciella riktade uppgifter och är inte ett system för allmänt ändamål med många uppgifter vilket gör att denna har ett inbyggt systems kännetecken och egenskaper. IPS har realtidsprestanda, är förenklad och är en fokuserad lösning till en låg kostnad i.o.m. att inga stora minnesskivor, monitorer m.m. behövs. Dessa egenskaper kännetecknar även ett inbyggt system. I jämförelse med en PC finns IPS-mjukvaran lagrad i ett program.

IPS har en specifik roll och uppgift och tillämpas i en speciell omgivning. Plattformen tar emot och sänder data, ansvarar för kommunikation på flera I/O-gränssnitt, fullföljer bildbehandling, svarar för avancerad systemtrafik över en 1 Gbps Ethernet-buss, tolkar och kvitterar kommandon och hanterar tid samt sköter TCP- och UDP-trafiken, signal- och socket-hantering, taskfördelning genom ett operativsystem m.m.

Beroende på vilka skydd, som sätts in i t.ex. kommunikationen kan plattformen skyddas och isoleras från något yttre. Även här kan plattformen klassificeras som ett inbyggt system. I en änd-till-änd koppling utesluts plattformen från något yttre och därmed är den tillförlitlig.

3.1.2 Användargränssnitt och tillförlitlighet

Inbyggda system har ofta allt från inget användargränssnitt alls till komplexa grafiska användargränssnitt. I bildbehandlingsplattformen skall finnas i huvudsak två MMI i värddatorn. Dels MMI emot Ethernet-bussen och dels JTAG-bussen. Olika tekniker används för att återstarta ett system om ett oväntat yttre fel genererar ett mjukvarufel. Ett exempel på en lösning till en omstart när ett yttre fel genereras är en vakthund styrd av en timer, som omstartar systemet om inte programvaran regelbundet meddelar vakthunden med ett speciellt data. I bildbehandlingsplattformen hålls systemet vaket bl.a. via Ethernet-bussen, som vaktar att kommunikationen till bussen är uppkopplad. Vidare kompletteringar kan göras via implementering av dylika vakter, som uppdateras via en timer-avbrottsrutin.

3.2 Ett FPGA-systems ambitioner

Generellt inkluderar ett FPGA-system en konstruktion av en ny hårdvarulösning med HDL-språk och/eller fördefinierade HW-biblioteksblock, nya eller/och kompletterande IP-kärnor/HW-block inkluderat tester, ändringar och anpassningar. Av en mjukvarulösning skapas en ny mjukvara inkluderad nya applikations-, funktions-, driv- och avbrottsrutiner med ett eller flera operativsystem. Dessa implementeras till en ny SoPC-/SoC-lösning i en FPGA. Alla delar optimeras för att snabbas upp i den nya lösningen. I och med att valet av teknik kommer detta att påverka utgångsresultatet och flera tekniker måste undersökas ur flera synvinklar. På så sätt får en FPGA-plattform för bildbehandling en optimal lösning och är en flexibel plattform för framtida forskning och utveckling.

I dag erbjuder FPGA-kretsen hårdvaru- och mjukvaruarkitektur i samma krets, inklusive snabb och smidig utvecklings- och testomgivning. Med FPGA-tekniken fokuserar man direkt på mjukvaru- och hårdvarulösningen och genomförandet av dessa i en och samma krets. Med denna teknik kan en ny lösning återgenereras under relativt kort tid i jämförelse med sedvanlig teknik som t.ex. med ASIC. Man skapar ett programmerbart system i en programmerbar krets (SoPC). Simulering kan utföras direkt med HW-kompilator eller separat mjukvara. Därmed vinner man tid i skapandet av hårdvarudelen. FPGA-kretsen har större kapacitet och inrymmer i dag hårdvaru- och mjukvaruresurser, som dylika lösningar behöver. FPGA erbjuder hårdvarukärnor och mjukvarukomponenter (softcore) för instansieringar. En typisk IP-kärna är Nios II-processorn, som används för exekvering av mjukvara medan NicheStack (även kallad NicheStack TCP/IP Stack) är en typisk mjukvarukomponent, som används för TCP-protokollhantering. FPGA-lösningen erbjuder även multiprocessor-instansieringar i en och samma krets.

Lösningar med ASIC är mycket kostsamma och kan inte beaktas i aktuella system då en dylik lösning är statisk och inte en test- och utvecklingsvänlig lösning för forskning. En bottenplatta måste ha dynamisk karaktär då nya system ständigt skapas. FPGA-lösningen har en låg utvecklingskostnad i förhållande till en ASIC-lösning. (Xilinx, 2015) ASIC är en tidskrävande lösning. Därför vinner en FPGA-lösning gentemot en ASIC-lösning i denna tillämpning. Inte heller CPLD är intressanta då dessa har en restriktivare struktur av programmerbara matriser och erbjuder ett relativt litet antal klocktypsregister.

3.3 FPGA-kretsen

En Field Programmable Gate Array (FPGA) är en integrerad krets, som konstruktören kan omprogrammera under utveckling. (Embedded Micro, 2013) En FPGA-hårdvarukonfiguration beskrivs med ett hårdvarubeskrivande språk (HDL), liknande det som används för en ASIC. Genom omprogrammerbarhet av logiska grindar och RAM i FPGA skapas smidigt ett hårdvaru- och mjukvarusystem.

FPGA används för system där möjlighet att uppdatera funktionaliteten är viktig. (Wiśniewski, 2009). I FPGA med sina programmerbara logikblock och konfigurerbara interkopplingar tillåts block att vara sammankopplade i olika konfigurationer. Logikblock kan konfigureras att utföra komplexa beräkningsfunktioner eller enklare logik. (Wiśniewski, 2009) FPGA-kretsen drivs av det faktum att den kombinerar de bästa delarna från ASIC, SW och processorbaserade system.

3.3.1 Intresset för FPGA och storlek på marknaden

Enligt Xilinx har marknaden ständigt ökat för FPGA-kretsar. År 1985 framställdes första kommersiella FPGA av Xilinx (Xilinx, 1997), år 1994 låg marknaden på $385 miljoner (Xilinx, 1997) och man uppskattar den ligger på $2,75 miljarder år 2010 (McGrath, 2006). År 1987 var antalet logiska element i Xilinx FPGA ca 9.000 stycken (Xilinx, 1997), år 1992 ca 600.000 stycken (Wayback Machine, 1996) och tidigt år 2000 flera miljoner. (Maxfield, 2004). På grund av det stigande intresset för FPGA ökade antalet startade projekt och år 2005 noterades 80.000 stycken och år 2008 90.000 stycken. (McGrath, 2006)

3.3.2 Utveckling med FPGA

Nya strategier växer fram inom FPGA-teknologin. Nu kombineras t.o.m. logiska block och sammankopplingar av traditionella FPGA med inbyggda mikroprocessorer, tillhörande kringutrustning och gränssnitt mot komplexa system-i-krets eller SoC-lösningar. (Xilinx, 2014b) SoC kan skapas för ASIC- eller FPGA-tillämpningar.

År 2010 introducerade Xilinx den första SoC-kretsen med bl.a. mikrokontrollerimplementeringar av 32-bitars processorer, minne och I/O i FPGA för att göra FPGA lättare att använda för konstruktörerna. (McConnel, 2010) Xilinx:s SoC FPGA Zynq-7000 har en dubbelkärnig ARM Cortex-A9 MPCore-processor, vardera med mellanminne (cache), valbart RAM- eller ROM-minne och I/O-gränssnitt. (Nass, 2010)

Den höga graden av integration bidrar till minskad energiförbrukning och värmeavgivning. En FPGA med en separat fast del för CPU har lägre kostnad och är ett mindre sy-

stem med högre tillförlitlighet då de flesta fel i modern elektronik uppstår på PCB i anslutningarna mellan kretsarna. (Altera, 2015a)

3.4 Ett utvecklingskort

Ett typiskt FPGA-utvecklingskort är ett bord innehållande hårdvarugränssnitt emot t.ex. USB-kommunikationskanaler, minne och kanske kontakter till ett eller flera externa kort genom vilka man kan kommunicera utåt. Ett bra utvecklingskort skall innehålla en mera komplett hårdvaruuppsättning av många möjliga komponenter, som man vill använda sig av i en plattform. Med ett utvecklingskort bygger man upp ett prototypsystem. Ett utvecklingskort med hårdvara kan användas för test och utveckling. Ett typexempel på ett utvecklingskort är Terasics ALTERA DE3-utvecklingskort. (Altera, 2015b)

3.5 Viktiga verktyg, hjälpmedel och komponenter vid konstruktion

I detta kapitel med underkapitel ges en överblick över viktiga verktyg, hjälpmedel och komponenter vid konstruktion, utveckling och uppbyggnad av en bildbehandlingsplattform med ett Nios II-processorsystem.

Denna överblick (se även bilaga 2) ges för orientering i konstruktionsarbetets olika faser och beroenden, som innefattar viktiga hörnstenar och som representerar en form av flödesdiagram i utvecklingen av en bildbehandlingsplattform. Genomgång startas överst till höger i bilaga 2, med Qsys-verktyg och omkringliggande komponenter för att fortsätta nedåt och avslutas med IPS längst ned, som representeras av en hårdvaru- och mjukvaruimplementering i Stratix III FPGA.

3.5.1 Qsys-verktyg för koppling till Avalon-buss

Konstruktionsarbetet av hårdvaran när ett Nios II-processorsystem skapas startas med Qsys-fasen och -verktyget, som är Alteras systemintegreringsverktyg och en delkomponent av Quartus II-verktyget. Med detta sparas betydande tid vid FPGA-

konstruktionsprocessen av hårdvarukonstruktioner med Qsys-komponenter (Qsys-komponent används synonymt med IP- eller HW-kärna). Qsys genererar automatiskt intern sammankopplingslogik för konstruktionen med IP- och HW-kärnor. Se kapitel 4.7.2 för tillvägagångssätt vid uppbyggnad och användningen med Qsys-verktyget och t.ex. kapitel 5.4.5 för fördjupning i Qsys angående enheten för bildbehandling.

Med Qsys-verktyget sammanställs och -kopplas IP- och HW-kärnor som t.ex. Nios II-processor-, Timer-, DMA-, PIO-, bildbehandlings-, bildbehandlingsmetodvalskärnor m.fl. Dessa kommunicerar internt via Avalon-bussen.

3.5.2 Avalon-buss för samkoppling mellan hårdvarukärnor

Avalon-bussens gränssnittsanvändning (se bilaga 1 och inom de till vänster och höger varande blå avgränsande linjerna med punktkopplingar) förenklar betydligt systemdesign av en plattform med Qsys-komponenter genom att man följer en standard vid anslutning av dessa till varandra. Avalon-gränssnittet har olika familjer av gränssnitt, som definierar aktuellt gränssnitt. Dessa familjer av gränssnitt är lämpliga för höghastighetsströmmande data, vid läsning och skrivning av register, minne, styrning av off-chip-enheter m.m. Dessa standardgränssnitt finns utformade i Qsys-verktyget.

Bildbehandlingsplattformen använder dessa standardiserade gränssnitt i sammankopplingen av sina Qsys-komponenter. Genom dessa standardgränssnitt ökas kompatibilitet i konstruktionen för framtiden. Avalon-bussen innehåller bl.a. data-, adress- och kontrollbuss. (Altera, 2014a)

Jämfört med en traditionell buss i ett processorbaserat system, där endast en bussledande tillåts, erbjuder Avalon-bussen Qsys-komponenter möjlighet till byte att vara bussledande och använda ett slavarbetsturgivningssystem där tur ges åt flera Qsys-komponenter att kommunicera eller vara bussledande och arbeta samtidigt i bakgrunden.

3.5.3 Nios II-processor för exekvering av mjukvara

För att exekvera mjukvara behöver bildbehandlingsplattformen en processor. Denna plattform använder en Nios II-processor (Nios II/f). Denna har 32-bitars inbyggt systems processorarkitektur och är speciellt skapad för Alteras FPGA-familjer. I och med detta blir plattformen passande för många inbyggda systemprogram, allt från DSP till systemkontroll. I plattformen har Nios II-processorn den styrandes mera än den bearbetandes roll. Nios II-processorn är jämförbar med MicroBlaze, som är en konkurrerande CPU för FPGA från Xilinx. Till Nios II-processorns hjälp finns ett bibliotek av kringutrustning och gränssnitt, som är tillgängliga royalty-fritt i Alteras FPGA. (Altera, 2014b) och (Altera, 2014c)

3.5.4 DMA-kärnor för dataöverföring

I en bildbehandlingsplattform där stora mängder av data överförs, är Direct Memory Access (DMA) IP-kärnor nödvändiga. Dessa överför data till eller från t.ex. en yttre enhet från eller till RAM *utan* hjälp av CPU. Därmed är CPU fri att fullfölja andra uppgifter under DMA-enhetens arbete. Vissa data behöver bara t.ex. överflyttas från ett ställe till ett annat och inte bearbetas eller behandlas. I dessa situationer sparar DMA tid åt andra enheter då den kan sköta överföringen av data. I denna avhandling används DMA 0 synonymt med den IP-kärna, som överför data från Ethernet-kretsen/-bussen till RAM och DMA 1 synonymt med den IP-kärna, som överför data till Ethernet-kretsen/-bussen från RAM.

3.5.5 Timer-kärna för operativsystem

I plattformen skall finnas delar, som har koppling till tid och är sammankopplade med mjukvaran. Därmed behövs en timer och en timer-avbrottsrutin. En timer behövs för att stega operativsystemet framåt men även för andra tidsrelaterade processer.

I en bildbehandlingsplattform används en s.k. *halvautomatisk* timer. Den består av ett laddnings- och räknarregister, en skrivsignal, klockinsignal och en utsignal. Programvaran laddar vid uppstart laddningsregistret med ett periodvärde. Det samma värdet förs även till räknarregistret. Fördelen med den halvautomatiska timern är att systemet får en

fast tid och period mellan timer-avbrotten, vilket ett operativsystem ofta behöver. Nackdelen med denna typ av timer är att systemet inte fritt kan välja tid för timer-avbrotten utan måste då först omladda laddningsregistret vid tidsbyte.

3.5.6 Parallel input/output- och General Purpose I/O-gränssnitt

För att läsa från eller skriva till yttre hårdvaruenheter skall finnas flera Parallel input/output (PIO). Dessa kan sammankoppla ut- eller ingångar direkt till eller från inre eller yttre enskilda signaler eller breda bussar.

För praktisk test och uppföljning av hårdvaran med t.ex. oscilloskop måste också finnas General Purpose I/O (GPIO)-anslutningar implementerade i plattformens hårdvara. GPIO har ett mycket bredare användningsområde än PIO. Alla signaler inuti FPGA kan kopplas till och från en GPIO-anslutning.

3.5.7 Ethernet- och JTAG-buss för kommunikation

Ethernet är en familj av datornätverk för lokala kommunikationsnätverk, Local Area Network (LAN). System för kommunikation via Ethernet delar upp en dataström i kortare stycken s.k. ramar. Varje ram innehåller käll- och slutadress, data, felkontroll av data m.m. för att skadad data kan identifieras och ev. återsändas. Sedan år 1980 har Ethernet bibehållit en hög grad av kompatibilitet. Uppbyggnaden av Ethernets MAC-adress och Ethernet-ram har påverkat uppbyggnaden av andra nätverksprotokoll. Ethernet-tekniken utvecklas ständigt för att möta nya krav på bandbredd på marknaden.

I och med behovet av snabb överföring av bilddata i bildbehandlingsplattformen behövs ett 1 Gbps Ethernet-gränssnitt. Dock har Terasics ALTERA DE3-utvecklingskort inte något sådant gränssnitt inbyggt utan ett sådant skall sammankopplas till utvecklingskortet via ett externt Ethernet HSMC-NET-dotterkort (Terasic, 2013). Dotterkortet, kopplat till moderkortet, innehåller två 1 Gbps Ethernet sändtagare med ett High Speed Mezzanine Connector (HSMC)-gränssnitt emot FPGA. För mera information om Ethernetbussen och sammankopplingen mellan Ethernet-bussens dotterkort och moderkortet hänvisas läsaren till kapitel 5.2. För fördjupande läsning om Ethernet hänvisas läsaren

till Telecom News Now (2011), Cisco, Juniper, HP drive Ethernet switch market in Q4 (2011) och filmen The History of Ethernet (2006).

En JTAG-buss (via Eclipse-verktyget) skall finnas och används för att t.ex. verifiera RAM, kontrollera register i processorn, stega fram i programmets mjukvaruinstruktioner, följa upp var mjukvara exekverats, sätta brytpunkt i SW m.m. Bildbehandlingsplattformen kommunicerar utåt via en JTAG Qsys-komponent i hårdvaran och vidare till värddatorn via ett USB-gränssnitt. Vid start utnyttjas JTAG-bussen först för laddning av hårdvaran och sedan mjukvaran.

3.5.8 Quartus II-verktyg för kompilering av hårdvara

Quartus II (v.11.0) är ett verktyg för konstruktion av programmerbara logiska enheter. Här används detta för kompilering av hårdvara, hårdvarubeskrivning, visuell visning av logisk uppbyggnad av hårdvara, simulering av hårdvara med ModelSim m.m. På Quartus II-verktygets nivå möts hela kedjan av HW-konstruktioner ända från Qsys-nivå där färdiga Qsys-komponenter implementerats i systemet med andra kompletterande HW-kärnor för att till slut bygga upp den slutliga hårdvarukonstruktionen.

3.5.9 Eclipse- och Nios II Software Build Tools-verktyg

Vid utveckling av mjukvaran skall *Nios II Software Build Tools* (SBT) för Eclipse användas. SBT är en uppsättning av plug-in (insticksprogram som ger tilläggsegenskaper) i Eclipse-ramverket *Eclipse C/C++ Development Toolkit* (CDT). SBT ger en utvecklingsomgivning, som fungerar likadant för alla typer av Nios II-processorer i inbyggda system. SBT innehåller en C/C++-kompilator, som är baserad på en GNU-verktygskedja. GNU-verktygskedjan är en övertäckande benämning på en samling *programmeringsverktyg*. Dessa verktyg används på olika sätt för utveckling av applikationer med operativsystem. Eclipse kan användas vid bl.a. körning, felsökning och profilering av program i Nios II-omgivning. (Altera, 2014d).

3.5.10 Program- och dataminne

I plattformen behövs dels ett inre minne (on-chip RAM) i FPGA men även ett externt minne (off-chip RAM) i ett yttre minneskort. On-chip-minnets storlek konfigureras till 128 kB och off-chip-minnets storlek till 256 MB. Var de olika typerna av data- och programsegment finns placerade, i det inre eller yttre minnet, är inte entydigt p.g.a. att detta styrs av kompilatorerna, hur kompilatorerna är konfigurerade men även andra faktorer, som t.ex. om data- och instruktionscacheminne finns konfigurerade för processorn.

Huvudregeln är dock att programminnet till största delen finns i det yttre minnet men även till någon del i minnet i FPGA p.g.a. vilka typer av programsegment som används. Programminnet består av konstanter, statiska data och mjukvara såsom operativsystemet, dess taskar, applikationsprogram, drivrutiner, avbrottsrutiner m.m. som Nios II-processorn exekverar. Mjukvaran kan även konfigureras till ROM och därmed behöver inte systemet återladdas efter ett strömbortfall.

Dataminnet finns i huvudsak placerat i on-chip RAM i FPGA men även till någon del i yttre minnet p.g.a. vilka typer av data-segment som används. Dataminnet innehåller och handhar variabler, buffertar, databaser, processorns behov av stack, heap m.m. Det behövs även ett dedikerat minne för DMA-kanalerna. Detta används som databuffertar för respektive kanal vid överföring av data. Läsaren hänvisas till kapitel 5.6 för mera information om systemets mjukvara och bilaga 3 och .map-, linker.h-, linker.x- och system.h-filen om systemets adresskonfiguration.

3.5.11 µC/OS-II-operativsystem

Operativsystemet skall fördela processorns resurser mellan olika mjukvaruprocesser (taskar), som exekveras och är därför viktigt. I IPS används Micriums µC/OS-II operativsystem. (Altera, 2011a) Operativsystemet är flyttbart till olika omgivningar, är realtidsbaserat och har en flerprocessexekverande kärna. Det erbjuder bl.a. tjänster som:

- Processexekvering och -hantering
- Tidshantering
- Minneshantering
- Intertask-hantering

Operativsystemet µC/OS-II:s kärna arbetar ovanpå HAL, Board Support Package (BSP) och Nios II-processorn. BSP är en implementation av *specifik stödmjukvara* för ett givet system, som överensstämmer med ett givet operativsystem. I och med denna arkitektur har mjukvaran med OS för Nios II-processorn fördelar som:

- Mjukvaran är flyttbar till andra Nios II-hårdvarusystem
- Mjukvaran är resistent mot förändringar i underliggande HW
- Mjukvaran kommer åt alla tjänster, som HAL erbjuder via API
- Avbrottsrutinerna är lätta att arbeta med och sätta in via API

Operativsystemet är prioritetsbaserat och har en förhållandevis liten realtidskärna. Det kan hantera upp till 64 processer (taskar). 56 processer är tillgängliga för programmeraren. Prioriteten för en task är satt så att ju lägre prioritetsvärde denna har desto högre prioritet har en task. För mera information om µC/OS-II hänvisas läsaren till (Labrosse, 2002a) och Micriums hemsida (Labrosse, 2016). Altera distribuerar endast µC/OS-II i Nios II Embedded Design Suite (EDS) och utvecklingssyfte. Micrium erbjuder fri användarlicens för universitet och studenter. (Altera, 2011b)

3.5.12 Användning av HAL och API

Mjukvarubiblioteket HAL innehåller en uppsättning av funktioner, som används för att initiera och få tillgång till varje typ av hårdvaruenhet från mjukvaran. Här används delar ur HAL vid uppbyggnaden av den slutliga mjukvaran och bildar mjukvarugränssnittet API. Mjukvarugränssnittet API är en abstraktion, som möjliggör att programmeraren kan använda operationer (skapa/radera taskar o.s.v.) på operativsystemnivå medan mjukvaran i sig fortfarande är *portabel* över en mängd olika hårdvaruuppsättningar.

De grundläggande delarna av HAL-arkitekturen ger följande tjänster och fördelar:

- Integrering med ANSI C-standardbibliotek och tillgång till C-standardbiblioteksfunktioner
- Drivrutiner, som ger tillgång till alla HW-enheter i systemet

- Systeminitiering, som utför initiering av *task* för processorn och exekveringsmiljö innan `main()`-proceduren exekveras
- Varje *enhet* instansieras och initieras i systemet innan `main()`-proceduren exekveras

3.5.13 Ethernets användargränssnitt

Kommunikationen skall ske via ett 1 Gbps Ethernet-gränssnitt med ett IPS testsystemprogram, som exekveras i en värddator och Windows-miljö. Användargränssnittet används som en client/server-terminal för protokoll som TCP och UDP och kan samtidigt buffra och hantera TCP- och UDP-data med många faciliteter. Det skall även finnas möjlighet att följa upp testresultaten med lagring av data i Excel-filer.

3.6 Parallellism i bildbehandlingsplattformen

Vid parallellberäkning och -exekvering utförs många operationer samtidigt. Detta fungerar genom principen att stora problem delas upp i flera mindre arbetsuppgifter, som sedan kan beräknas samtidigt, således parallellt. (Gottlieb & Almasi, 1989)

3.6.1 Beroendeskap inom parallellism

Parallellism kan i många fall användas för att nå en högre exekveringshastighet eller genomströmning. Å andra sidan medför parallellism olika utmaningar som måste tas hänsyn till. Beräkningar med parallellism medför bl.a. vissa beroendeskap, som är fundamentalt grundläggande delar vid genomförandet av parallella algoritmer. Dessa är att:

- hela förloppet kan *inte* köras snabbare än det längsta förloppet i kedjan (den s.k. kritiska linjen),
- beräkningar, som är beroende av tidigare beräkningar i kedjan kan *inte* inledas förrän de tidigare beräkningarna har slutförts och

- man måste anpassa antalet steg/beräkningar rätt så att *inga långa* steg finns i mitten av beräkningskedjan och därmed belastar de andra stegen p.g.a. detta/de långa beräkningssteg/en. (Ailawadi, 2009) och (Gottlieb & Almasi, 1989)

3.6.2 Olika typer av parallellism

Den första möjligheten till hårdvaruparallellism är vid *pipeline-beräkningar* där olika beräkningskretsgrupper kopplas till *samma hårdvaruklockpuls*. Genom att bygga upp ett passande nät av logiska kretsgrupper kan komplexa beräkningar göras på ett fåtal klockcykler. Processorn måste skriva och läsa synkront med pipeline-kedjans beräkningskretsgrupper i FPGA.

Den andra möjligheten till parallellism är *via pipeline-steg i processorn*. Processorn själv sköter genom sin HW-uppbyggnad om uppdelningen av instruktionernas olika pipeline-faser vid exekveringen av dessa. I och med att Nios II/f används i IPS finns tillgång till en pipeline-arkitektur i sex steg.

Den tredje möjligheten till parallellism är *via Avalon-bussen*. Här måste kopplingarna till HW-enheten eller enheterna vara kopplade på rätt sätt till Avalon-bussen så att flera enheter samtidigt kan fungera som master i helheten av systemet. En DMA kan vara kopplad som en master fastän även processorn är master men gentemot andra enheter i systemet. Före och efter dataöverföring samspelar processorn och DMA-kanalen om ett gemensamt minne där dessa utbyter data med varandra.

Den fjärde möjligheten till parallellism är när *HW-enheter delar en gemensam HW-klockpuls*. Denna parallellism fungerar i princip på samma sätt, som den förstnämnda men med den skillnaden att här går inte gränsen vid logiska operationer utan kopplingen till systemet kan vara direkt till och från en eller flera HW-enheter. Processorn måste dock skriva/läsa synkront med HW-grupperna i FPGA för att få in/ut rätt resultat.

I mjukvaran i bildbehandlingsplattformen finns ingen sann typ av parallellism. Alla mjukvaruprocesser exekveras endast synbarligen parallellt. I verkligheten får alla taskar (SW-processer) eller avbrott en liten processorkörtid vardera. Varje task eller avbrott

exekveras av processorn under kort tid varefter processorn snabbt byter till nästa task eller avbrott och exekverar denna/detta.

3.7 Pipeline

Pipeline är en uppsättning av *olika dataprocesselement* anslutna i serie, där utresultatet från en bearbetning av data är ett inelement till nästa bearbetningsprocess. Elementen i en pipeline utförs parallellt ovanpå varandra, sett ur tidsaxelns synvinkel, eller i en tidsskivad mode där då (del-)behandlat data kan sättas undan i mellanlagringsbuffertar mellan de olika elementen, som har delbehandlats.

3.7.1 Olika typer av pipeline

En *linjär pipeline* är en serie av bearbetningsstadier, som är ordnade att *linjärt utföra en bestämd funktion under en dataström*. De grundläggande användningsområdena för linjär pipeline är instruktionsexekvering, aritmetisk beräkning och minnesåtkomst. (Godse & Godse, 2006, s. 57) Den linjära pipeline-tekniken har dock en begränsning då den har beroendeskap av t.ex. instruktionernas ordning.

En *icke-linjär pipeline* (även kallad dynamisk pipeline) är en serie av bearbetningsstadier, som kan konfigureras att *utföra olika funktioner vid olika tidpunkter*. I en dynamisk pipeline finns framåtmatande eller bakåtmatande anslutningar i bearbetningen, som skickar data till och från olika delar i pipeline vid olika tillfällen. (Godse & Godse, 2006, s. 331) Vid en dylik pipeline skrivs data in varje gång eller allt eftersom till pipeline. Med dynamisk pipeline kan beroendeskap av instruktionernas ordning övervinnas. (Prabhu, 2009). För fördjupande läsning i dynamisk pipeline hänvisas läsaren till (Cooke, 2003), (Bharat & Sumit, 2009, ss. 58–), (Gaspar, 2013), (Laurenti, Djafarian, & Catan, 2002), (Stallings, 2012) och (Tsai, Chen, & Tseng, 2015).

3.7.2 Fördelar och nackdelar med pipeline

En fördel med pipeline är inte att den minskar tiden för en viss tidpunkts processgenomförande av data utan att genomströmningen av data ökas i systemet vid bearbetning av

alla åtgärder. En *hög (lång) pipeline* leder till en *ökning av latens* (tid mellan start för bearbetning av data tills hela bearbetningskedjan har genomfört alla bearbetningssteg i pipeline). Ju *högre (längre)* pipeline är desto *mera kan bearbetas* parallellt p.g.a. att flera bearbetningssteg samtidigt kan fullföljas på data.

En *nackdel* med pipeline är att den vanligen *kräver mera hårdvaruresurser* (HW-enheter för delbearbetning och minne) än med ett system, som bearbetar all data seriellt. Detta beror på hur mycket den seriella databearbetningen kan återanvända resursen i delbearbetningsskedet i steget före. (Laudon, Golla, & Grohoski, 2009, ss. 205–231) Om man betraktar en instruktionspipeline så kan den i vissa fall t.o.m. öka tiden för en instruktion att avslutas p.g.a. upp- och nedkörningsfasen av pipeline. Vid t.ex. uppkörningsfasen av pipeline tar det en viss tid att uppnå full genomströmning och fullskalig pipeline-exekvering.

Vid konstruktion av pipeline bör beaktas att utmatningen från en pipeline av färdigbearbetat data bör göras enligt den *långsammaste bearbetningsfasens hastighet*. En annan viktig faktor är att tillhandahålla *tillräckligt stora buffertar* mellan de olika pipelinefaserna om bearbetningstiderna kan *variera, dataelement kan skapas eller gå förlorade* längs pipeline. Detta görs för att undvika pipeline-köer. (Quinn, 2003) och (ZIPcores, 2013). Se även (Urhan & Franklin, 2001) och (Diego & Cerd, 2008, ss. 728–753).

3.7.3 Pipeline i bildbehandlingsplattformen

Emedan bildbehandlingsplattformen med sin FPGA har ett större antal hårdvaruelement och -celler kan den med pipeline och lämpliga metoder med multiplikation i hårdvaruenheter uppnå 400 M-beräkningar/s (se kapitel 2.4), en hög videobearbetningsfrekvens och långa FIR-filter. (Mirzaei, Hosangadi, & Kastner, 2006) Läsaren hänvisas till (Godse & Godse, 2006, ss. 321-331) för fördjupning i pipeline-konceptet.

4 HÅRD- OCH MJUKVARAN I PLATTFORMEN

I detta kapitel genomgås konstruktions- och implementeringsmetoder såsom HW/SW Co-Design och logistik och deras betydelse i en FPGA-helhetslösning. Vidare genomgås gränssnitt mellan och rollerna och platserna hos mjukvaran och hårdvaran i konstruktionskedjan och deras verktyg samt HAL-bibliotek och API-abstraktion.

4.1 Traditionell FPGA-konstruktion

Utmaningarna med FPGA-teknik i det förflutna har varit att endast de kunnigaste ingenjörerna med djup kunskap i hårdvarukonstruktion har kunnat använda lågnivå-FPGA-verktyg. Men genom ökning av verktyg för högnivå syntes- (HLS-)konstruktion har situationen radikalt ändrat utvecklingen med FPGA-teknik.

Hårdvarubeskrivningsspråk (HDL), som VHDL och Verilog, har utvecklats till primära beskrivningsspråk för hårdvaran och används för att utforma algoritmer i FPGA-kretsen. Genom syntax hos HDL kan även mappning eller anslutningar av externa eller interna I/O-signaler till andra block fullföljas, som slutligen fullgör helhetsfunktionen hos en algoritm.

Jämför man HDL med det sedvanliga seriella tänkandet i samband med ordinarie mjukvaruprogrammering kommer parallelliteten med vid användning av HDL. Nyckeln till förstående av HDL-programmering, alltså HW-algoritmer, ligger i att dessa fungerar parallellt och samtidigt. Här måste man fokusera på *alla* HW-blocks funktioner *grupperade efter drivande klock- och ingångssignaler* för att senare vidga vyn på samkonstruktioner och -reaktioner med de i övre hierarkin varande HW-blocken. Sist dras slutsatser av helfunktionerna.

4.2 Exempel på högnivå synteskonstruktion

Uppkomsten av grafisk högnivå syntes- (HLS-)konstruktionsverktyg, t.ex. National Instruments LabVIEW FPGA, har avlägsnat några av hindren för traditionell HDL-konstruktionsprocess. Programmeringsmiljön i LabVIEW FPGA är lämpad för FPGA-programmering då den representerar parallellism och dataflöde och kan användas för att integrera befintliga VHDL IP-kärnor i en LabVIEW FPGA-konstruktion. (National Instruments, 2012) Ingen direkt kunskap i lågnivå av HDL-språk behövs.

4.3 Co-Design

HW/SW Co-Design är synonymt med att systemet på systemnivå konstrueras *hårdvaru- och mjukvarumålmedvetet, samtidigt* och att *synergin* mellan hårdvara och mjukvara *utnyttjas* (Micheli & Gupta, 1997).

Inbyggda system för realtidsapplikationer har ofta operativa tidsfrister från en händelse till systemsvar, således att systemet svarar inom en viss tid, en s.k. tidsgräns. Inbyggda system byggs ofta upp och genomförs som blandade mjukvaru- och hårdvarusystem. I allmänhet används mjukvaran för funktion och flexibilitet medan hårdvaran används för prestanda. Vid traditionell konstruktionsmetod av inbyggda system skapas, specificeras och konstrueras HW och SW var för sig. Problem med denna metod är att:

- Den saknar en enhetlig HW/SW-representation vilket leder till svårigheter att kontrollera hela systemet och man passerar den på förhand definierade HW/SW-uppdelningen, som i sin tur leder till omfattande efterkonstruktioner och kan påverka hela systemets tid för färdigställande.
- En på förhand definierad HW/SW-uppdelning leder till *suboptimala* konstruktioner.
- Bristen på ett väldefinierat flöde gör att en specifikation har revideringssvårigheter, påverkar tid till marknaden-faktorn och när produkten färdigställs.

För att övervinna dessa problem måste det finnas en metod för specifikation, automatisk syntes och validering av dessa underklasser av inbyggda system. En konstruktion måste göras i en enhetlig ram, med en enhetlig HW/SW-representation för att inte påverka genomförandet av HW eller SW. *En* (1) modell måste bibehållas genom *en* (1) hel konstruktionsprocess, vilken syftar till att bevara ramegenskaperna hos konstruktionen. (Pederson, 2015)

Co-Design löser ovannämnda problem genom att man redan i planeringsskedet av ett system tar i beaktan och använder denna metod. Därmed undviker man problematik med att systemet saknar enhetlighet, som leder till svårigheter att kontrollera hela systemet. På grund av konstruktionspremissen hos metoden om synergi mellan HW och SW och att trygga HW/SW-representation utåt och inte erfara ödesdigra resultat för konstruktionen bör dagens konstruktörer ha erfarenhet av eller kunskap i både hårdvaru- och mjukvarukonstruktion eller vara både hårdvaru- och mjukvarukonstruktörer.

4.4 Datalogistik och signalering

Vid närmandet till hårdvaran och mjukvaran i bildbehandlingsplattformen vinns t.ex. tid genom att planera alla delar logistiskt rätt. Detta medför att alla delar i systemet får sina service-, betjänings- och informationsflöden i rätt tid och ordning för att tillfredsställa alla parter i rätt tid och ordning.

Mjukvara idkar samverkan med hårdvara genom olika kanaler såsom mjukvarustyrning och -kontroll och mjukvarukvitteringar medan hårdvara idkar samverkan med mjukvara på liknande sätt. Ett exempel på synergi och *HW-signalering* till SW i Co-Design är när Timer-kärnan har uppnått sin räknargräns och ett timer-avbrott aktiveras. Denna avbryter Nios II/f-processorns exekvering av ordinarie mjukvara och processorn startar att exekvera Timer-kärnans avbrottsrutin.

4.5 Mjuk- och hårdvarans roller i bildbehandlingsplattformen

Mjukvarans funktioner kan delas in i tre grupper. I den första gruppen har den styrandes roll, vilket är en summaprofil av (styr-)funktionaliteter hos plattformen. I den andra gruppen *betjänar* den *hårdvaran*, som t.ex. sker efter att en DMA IP-kärna har slutfört en dataöverföring. Då sänder IP-kärnan en avbrottsförfrågan till Nios II/f -processorn, som går till avbrottsrutinen för DMA-kanalen. I den tredje gruppen *betjänar* den *sig själv***,** som t.ex. när operativsystemet byter task. Vid byte av task uppdaterar mjukvaran den då aktuella taskens räknare och övriga status i taskens databas varefter operativsystemet laddar nästa i tur stående tasks räknare och status från den nya taskens databas.

Hårdvarans funktioner kan också delas in i tre grupper. I den första gruppen *fullföljer* den *de tunga arbetsprocesserna* i systemet, vilket är en summaprofil av (HW-)funktionaliteter hos plattformen. I den andra gruppen *betjänar* den *mjukvaran*, som t.ex. när hårdvaran ersätter en for-loop i SW för överföring av data med en DMA IP-kärnas överföring av data från ett till ett annat ställe i minnesrymden. I den tredje gruppen *betjänar* den *sig själv*, som t.ex. när *hårdvaran* i DMA IP-kärnan under överföringsprocessen själv *uppdaterar sina pekar- eller räknarregister.*

4.6 Konstruktion av FPGA-helhetslösning med Co-Design

Utvecklingen av hårdvaran och mjukvaran sker i arbetsdator, som innehar Quartus II-(v.11.0) (inkluderat Qsys-) och Eclipse-verktygets programvaror.

Hårdvaran och mjukvaran i bildbehandlingsplattformen utvecklas enligt samma modell som ett Co-Design-konstruktionsflöde i figur 5. I figuren omfattar lådorna Ⓐ-Ⓕ olika verktygsaktiviteter i utvecklingskedjan. Med hjälp av HW- och SW-utvecklingsverktygen byggs grundpelarna upp. I värddatorn definieras HW- och SW-funktioner med programmeringsspråk eller grafiska gränssnitt i konfigurationsfiler, program och databaser.

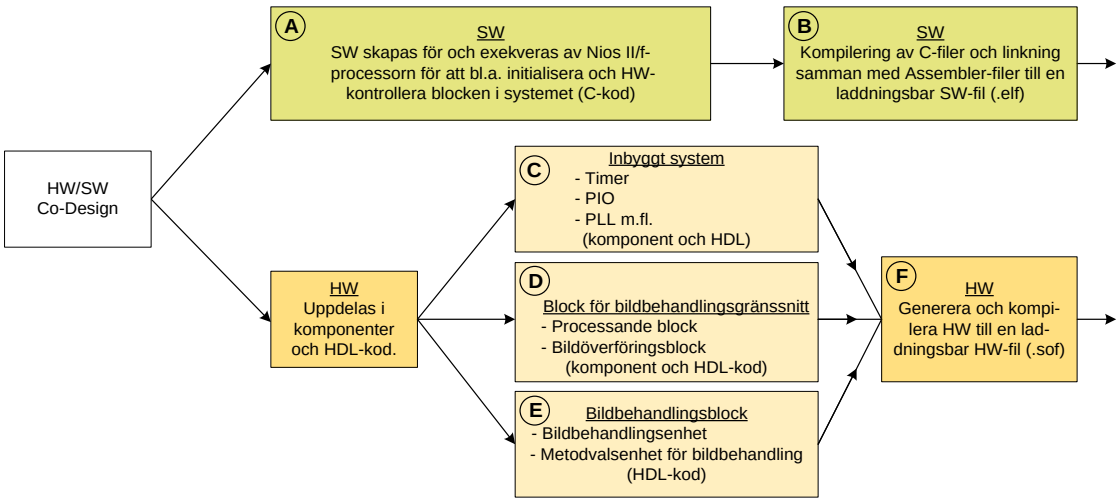


Fig. 5. Co-Design-konstruktionsflöde och faser (Desmouliers, Aslan, Oruklu, Saniie, & Martinez, 2010) (tillämpat på plattformen av författaren).

Co-Design-metodens premisser har en central roll vid utveckling av mållösningen för HW och SW och valet av uppbyggnadsprocess av HW och SW med respektive utvecklingsverktyg.

4.7 Verktyg för uppbyggnad av hårdvaran i FPGA

Utvecklingsverktygen av hårdvaran är Alteras Quartus II och Qsys. HDL-programmeringsspråken, i samarbete med Quartus II- (v. 11.0) och Qsys-verktyget, används vid skapandet av HW. Qsys-verktyget används när man skapar HW via ett grafiskt gränssnitt emot färdiga IP-kärnor.

4.7.1 HDL-källkod

Sedvanliga HDL-moduler (VHDL, Verilog o.s.v.) fullföljer hårdvarufunktioner. Dessa visas i figur 5 med fokus i lådorna Ⓒ, Ⓓ och Ⓔ. Med hjälp av HW- eller IP-kärnor bildas den slutliga lösningen som har basen av komponenter från systemets bibliotek av IP-kärnor.

4.7.2 Qsys

Verktyget Qsys förenklar konstruktionsprocessen och den blir snabbare hos system i FPGA med bl.a. systemintegration av Nios II-processorsystem samman med IP- och HW-kärnor. Dessa kärnor finns oftast i Qsys-komponentbiblioteket och är fördefinierade komplexa HW-funktioner och -kretsar, som är testade och optimerade. Exempel på IP-kärnor är DMA-, Timer- och PIO-komponenten. Av dessa skapas och sammanställs ett HW-system via grafiska gränssnittet i Qsys-verktyget.

Qsys-verktyget sparar betydande tid vid konstruktionsprocessen av ett FPGA-system genom att detta automatiskt genererar intern kopplingslogik för anslutningen av IP-kärnor till Avalon-interkopplingsbussen vid skapande av system. Qsys-verktygets placering i HW-konstruktionskedjan visas i figur 5 med fokus på *komponentkoden/-kärnorna* i lådorna Ⓒ, Ⓓ och Ⓔ och *generering* av databasen i låda Ⓕ. Egna HW-kärnor skapas och uppbyggs i HDL-källkod eller med konstruktionsverktyg för kretslayout. (Altera, 2014e) Ett typexempel är bildbehandlingsenheten. Efter Qsys-fasen genereras en databas med alla komponenter som använts under denna fas.

Verktyget Qsys drivs med en FPGA-optimerad nätverk-i-krets (NoC) teknik och är efterföljande SOPC Builder-verktyget. (Altera, 2016)

4.7.3 Quartus II

Verktyget Quartus II analyserar, kompilerar och sammanställer en helhet av en HDL-konstruktion, inkluderat Qsys-komponenter och HW-kärnor, till ett slutsystem. Detta kan göras efter att man har skapat alla HDL-filer antingen manuellt eller med konstruktionsverktyget Qsys. Quartus II-verktygets process består av flera delsteg som *design* (konstruktion), *syntes* (sammanställning och översättning), *nedladdning* och *testning*. Quartus II-verktygets arbete fokuseras i låda Ⓕ i figur 5. (Altera, 2014f)

I syntessteget, när HW-konstruktionen har skapats med HDL och verifierats så matas denna till ett sammanställningsverktyg, som tar logiken igenom flera komplexa steg. I syntessteget översätts konstruktionen till ett lågnivåspråk för FPGA för att optimera lo-

gikimplementeringen. Här kartläggs den tekniska konstruktionen för genomförande i FPGA-målkretsen med beaktande av logiska resurser som LE, ALM och andra logiska block eller celltyper.

För vald FPGA-målkrets och utvecklingskort genereras automatiskt även en på teknisk nivå kartlagd nätlista för FPGA-kretsens stiftnummer och signalnamn, det s.k. stiftsplaneringssteget, emot skapad HW-konfiguration och symboler i HDL-källkoden. Resultatet är en konfigurationsfil (en bitström), som innehåller information om hur de olika signalerna och blocken skall sammankopplas i FPGA-målkretsen.

Efter att konstruktion och översättning är slutförda är den binära fil (.sof), som skapats av Quartus II-verktyget klar för *nedladdning* och konfigurerar mål-FPGA t.ex. via det seriella gränssnittet (JTAG). Sist visar *teststeget* om konfigurationen utfallit väl. För vidare läsning om alla faser och steg hänvisas läsaren till (Altera, 2014g), (Altera, 2015c) och (Altera, 2014h).

4.8 Verktyg för uppbyggnad av mjukvaran i FPGA

Som utvecklingsverktyg för mjukvaran används Eclipse-programvaran (för Nios II-processorsystem (v. 11.0)). Nyttan med detta är att den erbjuder en snabb och smidig utveckling av SW. I Alteras tillämpningar sköter Eclipse-verktyget kompilering och sammanställning av alla SW-filer till en slutlig konfigurationsfil (.elf), som sist laddas ned till Terasics ALTERA DE3-utvecklingskort.

I IPS finns C- och Assembler-baserad mjukvara, som innehåller bl.a. operativsystem, taskar, driv-, avbrotts- och kommunikationsrutiner, applikationsprogram, kommandotolkar m.m. I denna avhandling behandlas endast den C-baserade mjukvaran (förutom vid Reset-situation).

4.8.1 C- och Assembler

Mjukvaran har kodats i C- och Assembler-språket och exekveras i Nios II/f-processorn. Det finns olika C-funktioner och -procedurer för att implementera ett operativsystem,

operativsystemsrutiner via HAL API, olika typer av driv- och avbrottsrutiner för t.ex. DMA och timer, rutiner för PIO m.fl. I figur 5 och låda Ⓐ visar C-källkodens plats i SW-kedjans uppbyggnad. Via SW styrs SW och HW-funktioner via HW-block, skrivningar och läsningar till och från minnesrymden, olika task, avbrottsrutiner m.m.

4.8.2 Eclipse-verktyget

Med Eclipse-verktyget har man överblick över mjukvaran. Verktyget kompilerar, linkar och sammanställer hela SW. I figur 5 och låda Ⓑ finns dess placering i utvecklingskedjan. Via Eclipse kan man felsöka koden i målkortet. Verktyget erbjuder bättre felsökningsmöjligheter än t.ex. Altera Monitor-verktyget då Eclipse ger bättre möjlighet för felsökning på både C- och Assembler-nivå, avläsning av och skrivning till alla delar av minnet, yttre enheter, inre hårdvaruenheter, uppföljning av målprocessorns register m.m.

4.8.3 Mjukvarubiblioteket HAL och mjukvarugränssnittet API

Mjukvarubiblioteket HAL har många drivrutiner, som används emot olika hårdvara i systemet. I mjukvaran används delar ur HAL, som ett mjukvarugränssnittet (API) emot aktuell hårdvara, som SW utnyttjar vid skrivning till eller läsning av aktuell konfigurerad HW. Vid kompilering, när API byggs upp för aktuell HW, skapar Eclipse-verktyget ett utdrag på basis av systeminformationen i SOPC-filen (.sopcinfo). I figur 6 visas olika lager i ett HAL-baserat system, från applikationsprogram, genom HAL API till verklig HW i systemet.

HAL-arrangemanget med drivrutiner för medvarande HW delar upp och ger en tydlig skillnad på SW-applikationen och själva drivrutinen för en HW-enhet och förenklar sättet att skriva drivrutiner för ny HW och kringutrustning. Denna drivrutinsabstraktion främjar återanvändandet av SW-kod, som är *resistent mot förändringar* i underliggande HW.

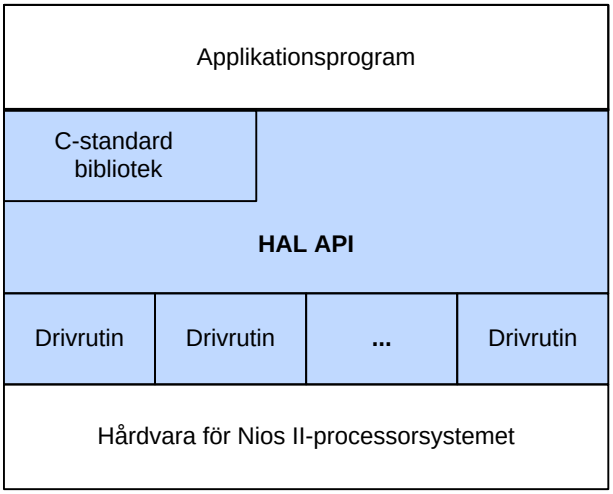


Fig. 6. Ett HAL API-system har flera lager, som gör att mjukvaran är *resistent mot förändringar* i den underliggande hårdvaran för Nios II processorsystemet.

För fördjupande läsning hänvisas läsaren till (Altera, 2010a) och (Altera, 2011c). Tillgång till respektive HW- eller IP-kärna i systemet fås genom skrivning eller läsning i minnesrymden vilket kan jämföras med användning av en mappad minnesarea. Angående HW- eller IP-kärnornas adresser hänvisas läsaren till bilagorna 1 och 3.

5 IMPLEMENTERING

I detta kapitel behandlas och tas upp implementeringen i bildbehandlingsplattformen. Helhetssystemet i figur 2, på sidan 23, kan i princip indelas i två huvuddelar. Dels en bildbehandlingsenhet (bildbehandlingsplattformen) i mitten, som består av en mjukvara och hårdvara och dels en värddator till höger. I värddatorn exekveras ett terminalprogram, IPS testsystem och Eclipse-verktyget, under Windows 7, med Ethernet-gränssnitt. Värddatorn och IPS kommunicerar med varandra på en 1 Gbps Ethernet-buss via en Ethernet-kabel och på en JTAG-buss via en USB-kabel. Via värddatorn och JTAG-MMI i Eclipse-verktyget och Ethernet-MMI kan plattformen kontrolleras och testas. Genom dessa kan man styra funktioner, ge kommandon, sända och ta emot data och följa upp sänt och mottaget data på Ethernet- och JTAG-bussen i plattformen.

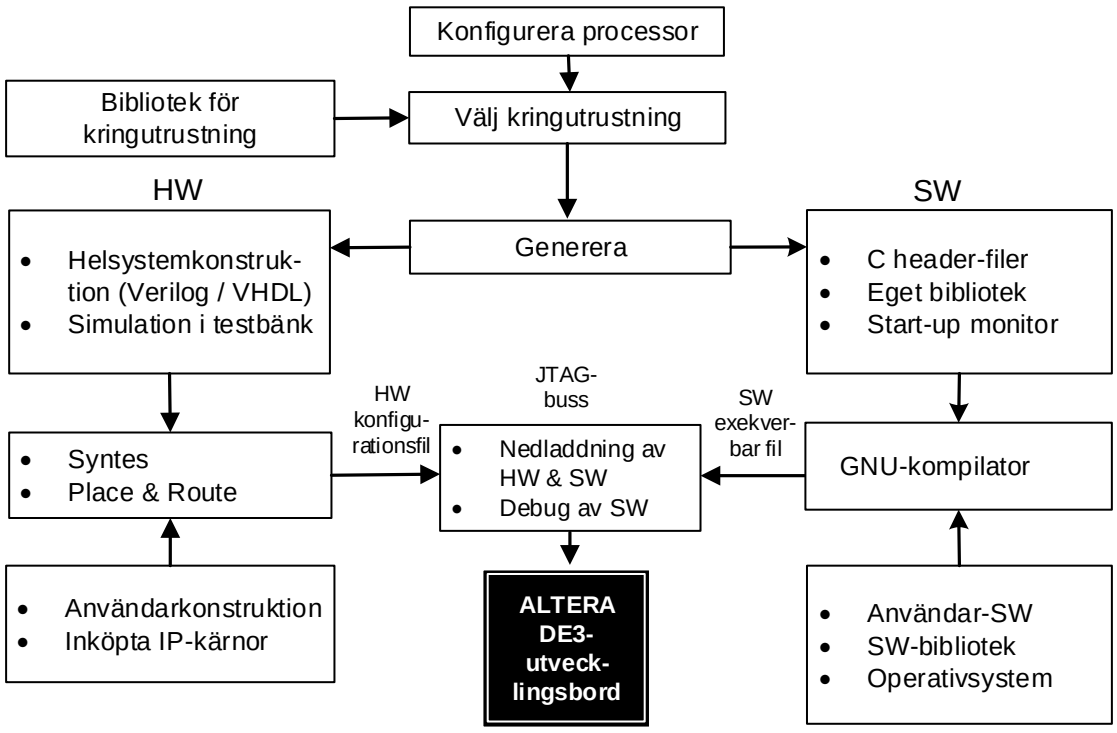


Fig. 7. Vägen vid skapande av databaserna för hårdvara och mjukvara. Anpassad från (Altera, 2008) av författaren.

I figur 7 presenteras det generella tillvägagångssättet vid skapandet av ett Nios II-processor hårdvaru- och mjukvarusystem, som genomgår flera faser.

5.1 Utvecklingskortet

Utvecklingskort för hårdvaran i bildbehandlingsplattformen är Terasics ALTERA DE3-utvecklingskort med Stratix III FPGA-krets (figur 8).

Via externkort eller programmering har utvecklingskortet olika typer av kommunikationsbussar, gränssnitt emot två 1 Gbps Ethernet-kanaler, höghastighets OTG USB-gränssnitt, USB-gränssnitt emot JTAG-buss för hantering av mjukvara och debug av den samma, temperatursensor och uttag för SD-kort. Det finns gränssnitt för olika typer av mätinstrument för mätningar i realtid och sammankopplingar till yttre monitorer, I^2C-bussar för kommunikation med gränssnitt, databussar emot externa gränssnitt m.m. Det finns även ett lokalt enkelt MMI med sensorer och olika brytartyper och monitorer.

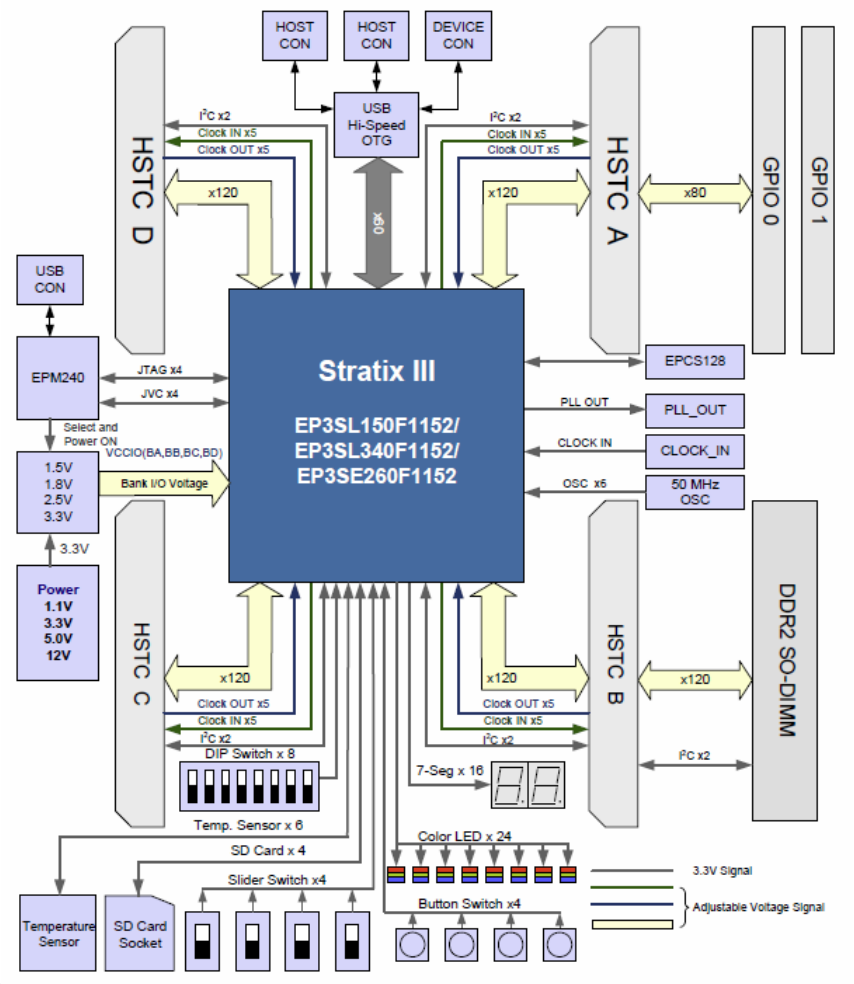


Fig. 8. Terasics ALTERA DE3-utvecklingskortets blockdiagram. (Terasic, 2009)

Två eller flera utvecklingskort kan sammankopplas och bilda ett eller flera skilda självständiga system utåt och köras som en eller flera skilda självständiga enhet(er) utåt.

Utvecklingskortet har omfattande utvecklingsverktyg vilka innehåller många enheter för olika kringutrustning och för samkonstruktion med tilläggskort för utbyggnad till stora system. Kortet har premisser för bildbehandling då det har gränssnitt för externt storminne med ända upp till 1 GB (med vald typ av minneskort). Läsaren hänvisas till (Terasic, 2009) för information om uppbyggnaden av utvecklingskortet och placeringen av de olika komponenterna.

Utvecklingskortet har en FPGA (EP3SL150F1152), som har stort logikminne (142.500 stycken likvärdiga LE) och Trimatrix SRAM-minne med 355 stycken M9K-, 16 stycken M144K- och 2.850 stycken MLAB-block. Den innehåller sammanlagt 5.499 inbäddat RAM-kb eller 6.390 RAM-kb då MLAB-block inräknas. Ytterligare finns 384 stycken multiplikatorer med 18 x 18 bredd och 8 stycken phase-locked-loop (PLL) enheter. (Altera, 2010b) För ytterligare information om Stratix III FPGA se (Altera, 2015d).

5.2 Moder-, dotter- och minneskortet

Moderkortet, Terasics ALTERA DE3-utvecklingskort, och dotterkortet, HSMC-NET med två Ethernet-gränssnitt, sätts samman via HSTC C-kontakten (figur 1 och figur 8) på moderkortet. Via dotterkortet kan kommunikation ske på Ethernet-bussen. Med hjälp av mjukvara och hårdvara för Media Access Control (MAC)-gränssnittet konfigureras Ethernet-kretsarna och sänds och tas emot data från och till kortet respektive.

På dotterkortet (Terasic, 2010) finns Ethernet-kretsar, kristaller, EEPROM och LED, som indikerar trafikstatus på Ethernet-bussen. EEPROM-kretsen kan konfigureras via ett I^2C-gränssnitt, mjukvara och hårdvara. Via dotterkortet och ett RJ-45-gränssnitt kopplas bildbehandlingsplattformen till Ethernet-bussen. Ethernet-krets och -gränssnitt som används för Ethernet-trafik på dotterkortet är kanal 1. Resulterande bildbehandlat data sänds via dotterkortet över Ethernet-bussen i skurar på ca 118 Mpixel/s. Läsarens hänvisas till (Nobel, 2011) för fördjupning i information om 1 Gbps-bussens verkliga

dataöverföring i ett nätverk. Det yttre minneskortet (1 GB), sammansatt med moderkortet, kan ta hand om stora datamängder i systemet.

5.3 Hårdvarublocken

I figur 9 visas de viktigaste blocken i bildbehandlingsplattformens hårdvara.

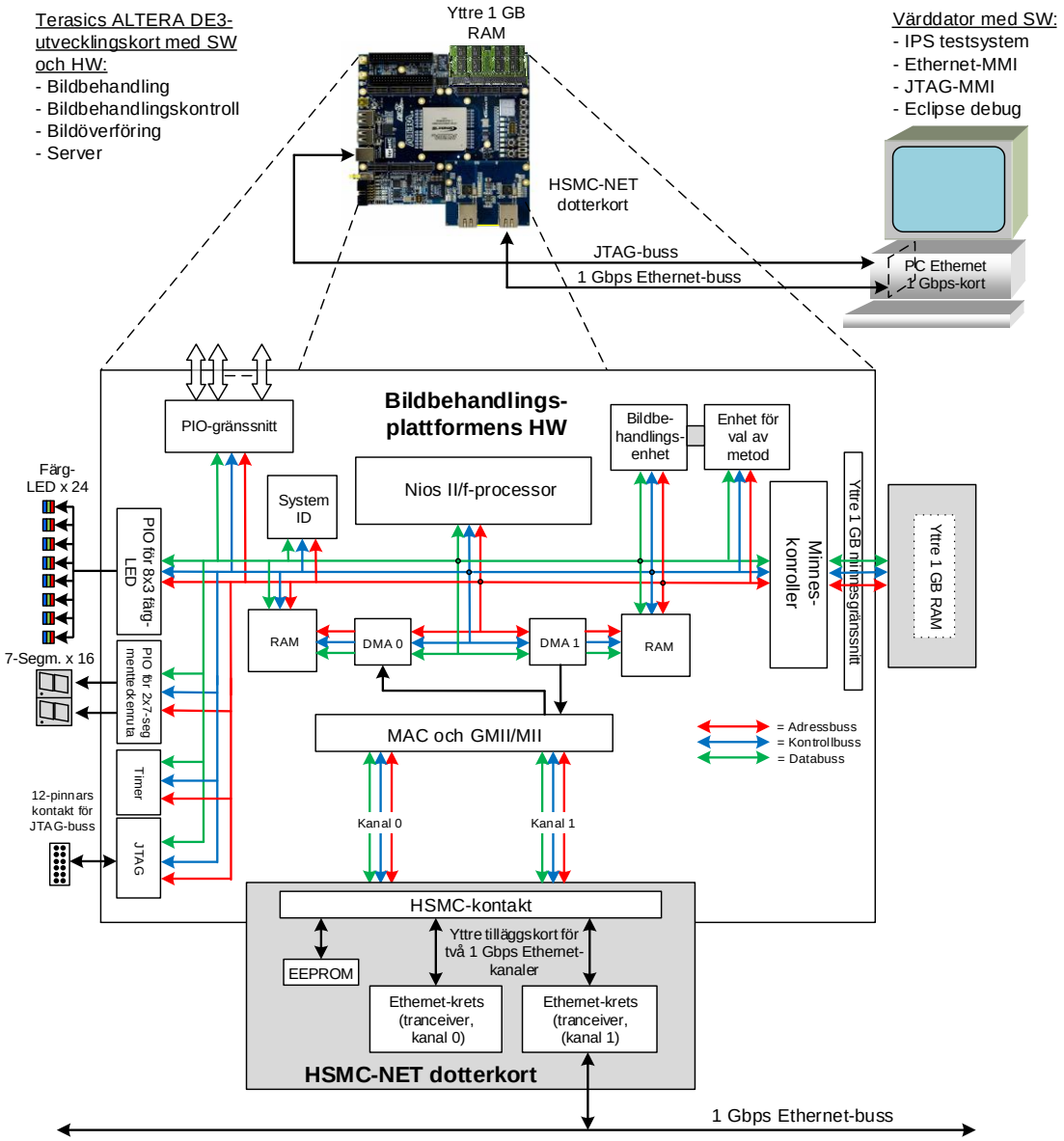


Fig. 9. Överblick över plattformens viktigaste hårdvarublock, sammankopplingar och uppbyggnad med undermoduler och kontakt till omgivningen.

Vidare i figur 9 visas hårdvarans uppbyggnad, undermoduler, yttre minnes- och tilläggskort för kommunikation till Ethernet- och JTAG-bussen samt sammankoppling mellan utvecklingskort och värddator. Hårdvaran i systemet fullföljer det tunga arbetet av och med data, till och ifrån Ethernet- och JTAG-gränssnittet och sköter kommunikation, busstrafik, beräkningar, kontroll och synkroniseringar av skeenden och händelser i, med och mellan alla IP- och HW-kärnor i systemet. Delar av mjukvaran ersätts med hårdvara. Läsaren hänvisas till bilaga 1 för beskrivning av hårdvarans uppbyggnad och IP- och HW-kärnor i IPS emot Avalon-bussen i Qsys-verktygets grafiska gränssnitt och bilaga 3 för hårdvarans adresskarta.

5.4 Funktioner implementerade i hårdvaran

I hårdvaran finns bl.a. IP-kärnor (-enheter) som DMA, PIO, JTAG, Nios II/f, Timer, GMII/MII, MAC och HW-kärnor för bildbehandling, val av bildbehandlingsmetod, bildbehandlingsgränssnitt samt några bildbehandlingsmetoder. I HW finns kretsgrupper skrivna i VHDL, Verilog eller skapade med Qsys. Många av dessa ersätter eller accelererar mjukvara. Från värddatorn sänds styr- eller bildbehandlingskommandon via Ethernet-bussen eller kommandon för att välja bildbehandlingsmetod via JTAG-bussen. Bildbehandlingsplattformen svarar på kommandon och sänder tillbaka bilddata behandlad antingen i TCP- eller UDP-format till värddatorn, som presenterar denna.

I figur 10 visas HW-enheter och deras funktionsmässiga sammankopplingar till varandra i hårdvaran i plattformen. Använd Ethernet-krets (-gränssnitt), som via sina bussar är kopplade från utvecklingskortets gränssnitt för dotterkortet, är vidarekopplad till GMII/MII. Aktuell Ethernet-krets buffrar inkommande eller avgående data från eller till Ethernet-bussen respektive. GMII/MII sköter bl.a. om möjlighet av val mellan 4- eller 8-bitars bredd på databussen till och från dotterkortet och via MAC med namnet `tse_mac` (Triple Speed Ethernet MegaCore) möjliggörs tre hastigheter (10, 100 och 1.000 Mbps) på Ethernet-bussen. DMA 0- och DMA 1-kärnan överför buffrat ankommet eller avgående data till eller från RAM respektive.

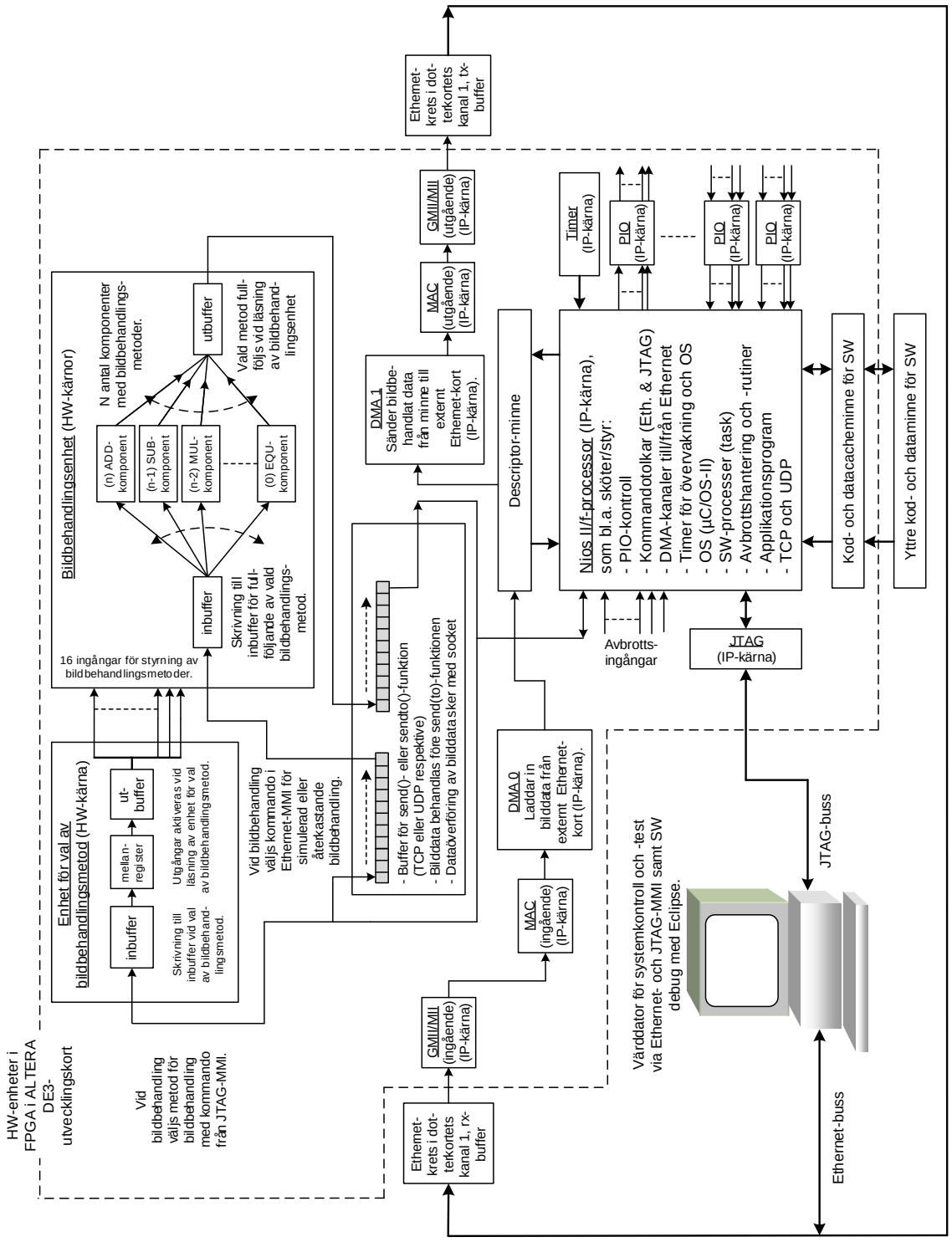


Fig. 10. En logisk bild och överblick över bildbehandlingssystemets funktionella sammankopplingar, HW-enheter och komponenter med värddator.

Nios II/f-processorn styr systemet och exekverar operativsystem, drivrutiner, avbrottsrutiner och applikationer. I (on-chip och off-chip) RAM finns program- och dataminnet, programvariabler, stack m.m. Timern sköter om att operativsystemet drivs framåt synkroniserat med tiden och övervakar skeenden som taskväxlingar och tids- och taskfördelningar.

Det finns separata HW-kärnor för bildbehandling och val av bildbehandlingsmetod (figur 10). JTAG-bussen sköter kommunikationen till och från värddatorn varifrån man bl.a. kan göra felsökning och följa upp förlopp i mjukvaran. Via PIO sköts bl.a. utvecklingskortets lokala MMI (brytar-, LED-, 7-segmentteckenrutans- och dotterkortets EEPROM-hantering).

RTL-schema över systemet skapas med hjälp av Quartus II-verktyget. Detta har många undernivåer. Genom schemat kan olika delar i systemet sökas på alla nivåer (kärnornas block kan dock vara krypterade). Läsaren hänvisas till bilaga 4 för att stifta bekantskap med RTL-schemat över den översta nivån av helhetsuppbyggnaden av hårdvaran i Stratix III FPGA och dotterkortets inre delar.

5.4.1 DMA

I hårdvaran används två DMA (Direct Memory Access) IP-kärnor. Den ena, med namnet `sgdma_rx` (se bilaga 1), är den DMA-kanal som sköter dataöverföringen från Ethernet-kretsen på dotterkortet till RAM. Den andra, med namnet `sgdma_tx` (se bilaga 1), är den DMA-kanal som sköter dataöverföringen från RAM till Ethernet-kretsen på dotterkortet. Viss del av RAM (descriptor-minnet) är tilldelat för DMA-kanalernas användning. (Altera, 2015e)

Som ett exempel tas här upp DMA IP-kärnan `sgdma_rx`. Denna har fyra ingångssignaler mot Avalon-bussen. Till exempel ingångssignalen `reset` används för att nollställa DMA IP-kärnan och signalen `in` används för att synkronisera DMA IP-kärnans access från MAC IP-kärnan med namnet `tse_mac`.

5.4.2 Nios II/f

I hårdvaran finns instansierad en Nios II/f-processor. Denna finns tillgänglig i Qsys-biblioteket. Processorn inkluderar många (vissa alternativa eller valfria) enheter (figur 11) för att kunna betjäna sin omgivning på olika sätt. Nios II/f-processorn har en mjukvarukärna med RISC-arkitektur och gjord helt i programmerbara FPGA-logik- och -minnes-block. Användaren kan själv utöka Nios II/f-processorns grundläggande funktionalitet och instruktionsuppsättning genom att skapa egna instruktioner anpassade för utökad kringutrustning eller för att öka hastigheten på tidskritiska mjukvarualgoritmer.

Några exempel på signaler till Avalon-bussen från Nios II/f-kärnan, `cpu` (se bilaga 1), är `reset_n`, som används för att nollställa processorn, `clk`, som används för att exekvera och synkronisera signaler vid skrivning till och läsning från IP- och HW-kärnor och processorns ingångssignaler `IRQ 0-3`, som är avbrottsignaler och kopplade till olika IP-kärnor för betjäning av deras avbrottsförfrågan. (Altera, 2014i)

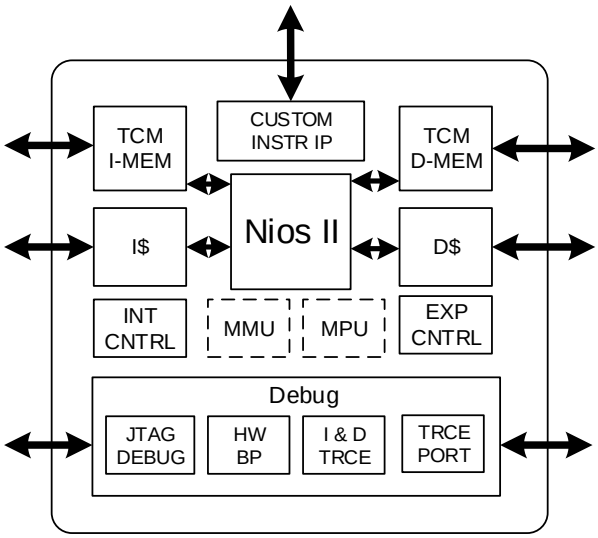


Fig. 11. Nios II/f-processorns inre enheter såsom Debug (dubuggning), INT CNTRL (avbrottshantering), MMU och MPU (minneshantering och -skydd), EXP CNTRL (undantagshantering), I$ (instruktionscache), D$ (datacache), TCM I-MEM (snabbt instruktionsminne utan cache), TCM D-MEM (snabbt dataminne utan cache), CUSTOM INSTR IP (anpassade instruktioner) och Nios II (exekvering av mjukvara). Anpassad från (Altera, 2014j) av författaren.

Anpassade instruktioner kan användas i samband med t.ex. optimering av mjukvara och slingor för digital signalbehandling (DSP). Det är här signalen `custom_instruction_master` kommer in. Signalen är f.n. inte kopplad till någon enhet/kärna men finns till för framtida bruk. Läsaren hänvisas vidare till (Altera, 2007a) angående specifikationer av de olika gränssnitten mellan Avalon-bussen och Nios II/f-processorn och (Altera, 2014j) och (Altera, 2015f) för fördjupning i processorns funktioner.

5.4.3 Timer

I hårdvaran finns tillgänglig en Timer IP-kärna. Läsaren hänvisas till kapitlen 3.5.5 och 5.6.8 för uppgifter hos timern. IP-kärnan har tre ingångssignaler mot Avalon-bussen, t.ex. `reset` för att nollställa utgångarna och sig själv, `clk` för att synkronisera kontrollbussens signaler vid skrivning till och läsning från IP-kärnan och `s1`, som innehåller adress-, data- och kontrollbuss. Läsaren hänvisas till (Altera, 2007a) angående specifikationer av de olika gränssnitten mellan Avalon-bussen och Timer-kärna.

5.4.4 PIO

I hårdvaran finns tillgängliga åtta PIO IP-kärnor. Dessa finns tillgängliga i Qsys-biblioteket. Utgångssignalerna från vissa PIO sätts enligt skrivna ingångsdata till respektive PIO och ingångssignalerna till vissa PIO sätts enligt avlästa yttre ingångsdata till respektive PIO.

Ett exempel på PIO är IP-kärnan `pio_led` (se bilaga 1), som har tre ingångar mot Avalon-bussen. Dels `reset` för att nollställa utgångssignalerna, dels `clk` för att synkronisera kontrollbussens signaler vid skrivning till och läsning av kärnan och dels `s1`, som innehåller adress-, data- och kontrollbuss. Till utgången från denna PIO, med signalnamn `pio_led_external_connection` innehållande 24 utsignaler, är kopplade LED för att indikera systemets inställning. Läsaren hänvisas till (Altera, 2007a) angående specifikationer av de olika gränssnitten mellan Avalon-bussen och PIO.

5.4.5 Enheten för bildbehandling

I hårdvaran finns en hårdvarukärna för bildbehandling. HW-kärnan, ett gemensamt gränssnitt emot bildbehandlingsmetoderna (`reg16_avalon_interface_0`), kan skrivas och läsas med olika bredd på databussen (n.v. version har 16-bitars databuss) (se figur 12 och bilaga 1). Till HW-kärnan kan kopplas ett fritt antal valfria konstruerade beräknande komponenter (funktioner, även kallade bildbehandlingsacceleratorer), som bara begränsas av FPGA-kretsens antal HW-celler. HW-kärnan har fyra ingångar mot Avalon-bussen. Dels `reset_sink`, för att nollställa kärnan, dels `clk_sink` för att synkronisera kontrollbussens signaler vid skrivning till och läsning av kärnan, dels `avalon_slave`, som innehåller adress-, data- och kontrollbuss och dels `image_process_controll`, som f.n. innehåller en 16-bitars databuss för val av bildbehandlingsmetod.

Beroende på konstruktion av HW-kärnan finns möjlighet att välja $(2^{16}–1)$ olika eller 16 samtidiga enskilda bildbehandlingsmetoder. Genom att öka antalet ingångar för val av bildbehandlingsmetoder kan antalet möjliga metoder ökas.

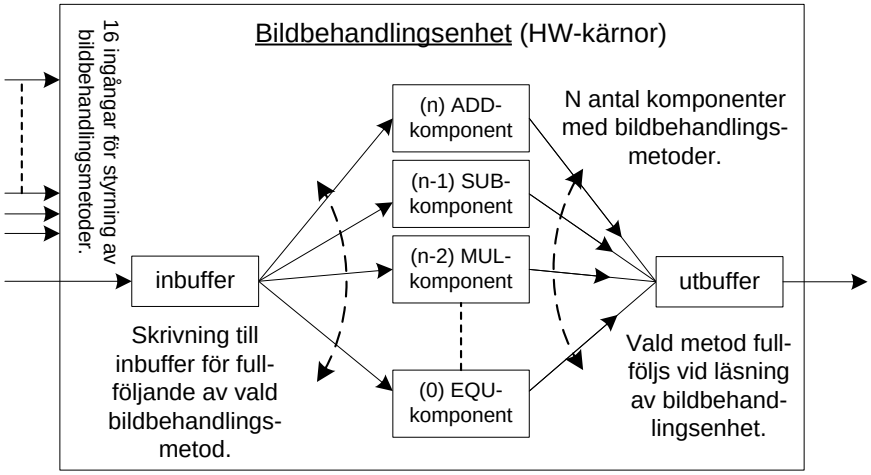


Fig. 12. Genom inskrivning av data till vald komponent (HW-kärna) fullföljs vald beräkningsmetod vid läsning av bildbehandlingsenheten.

Kärnan har en externkopplad utgång, `image_process_status`, som är reserverad för framtida bruk. Läsaren hänvisas till kapitel 5.4.4 och PIO:s specifikationer av gränssnitt mot Avalon-bussen.

5.4.6 Enheten för val av bildbehandlingsmetod

I hårdvaran finns en hårdvarukärna för att välja bildbehandlingsmetod. Databussen hos HW-kärnan (figur 13) kan skapas med olika bredd (n.v. version har 16-bitars databuss) och har tre ingångar mot Avalon-bussen: `reset_sink`, `clk_sink` och `avalon_slave` (som innehåller adress-, data- och kontrollbussen). Kärnan har en utgång (f.n. 16-bitars databuss), som är kopplad till bildbehandlingsenhetens ingång.

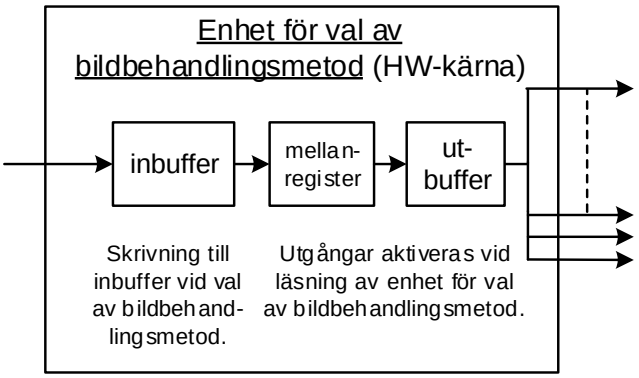


Fig. 13. HW-kärnan (se bilaga 1) för val av bildbehandlingsmetod med in- och utgångar.

När val av metod skall fullföljas sker detta i två steg. Först skrivs valdata in till adressen för enheten för val av behandlingsmetod. När man sedan läser adressen för enheten ställs utgångarna enligt det valdata, som just skrevs in. Eftersom utgångsignalerna är kopplade till enheten för bildbehandling kommer denna nu att börja behandla data enligt den nya inställda bildbehandlingsmetoden. Valet kan ändras med en pixels resolution. En färgpixel behöver dock tre byte (B) av data vid användning av RGB-protokoll.

5.4.7 Pipeline vid bildbehandling

I hårdvarukärnan för bildbehandling finns möjligheter för användning av *linjär pipeline* eller *dynamisk pipeline*. Läsaren hänvisas till kapitel 3.7.1 för deras funktion.

Vid linjär pipeline skrivs data in varje gång till pipeline. Vid t.ex. 32-bitars pipeline databuss kan fyra pixel (vardera med 8-bitars bredd) bilddata skrivas in samtidigt. Vid läsning av data från pipeline bearbetas fyra pixel i pipeline-kedjan samtidigt med vald be-

räkningsmetod. En pipeline-cykel består av en skrivning och en läsning. Plattformen använder en bytebaserad linjär pipeline med endast den första beräkningens pipeline aktiv.

I figur 14 visas ett exempel på pipeline-exekvering med beräkning av sju datakvartetter och upp- och nedkörning av pipeline. När t.ex. full pipeline är aktiv skrivs samtidigt datakvartett tre (D3) in till aktuellt mellanregister för första beräkningens pipeline, D2, som genomgått första beräkning (B1) in till aktuellt mellanregister för andra beräkningens pipeline och D1, som genomgått andra beräkning (B2) in till aktuellt mellanregister för tredje beräkningens pipeline.

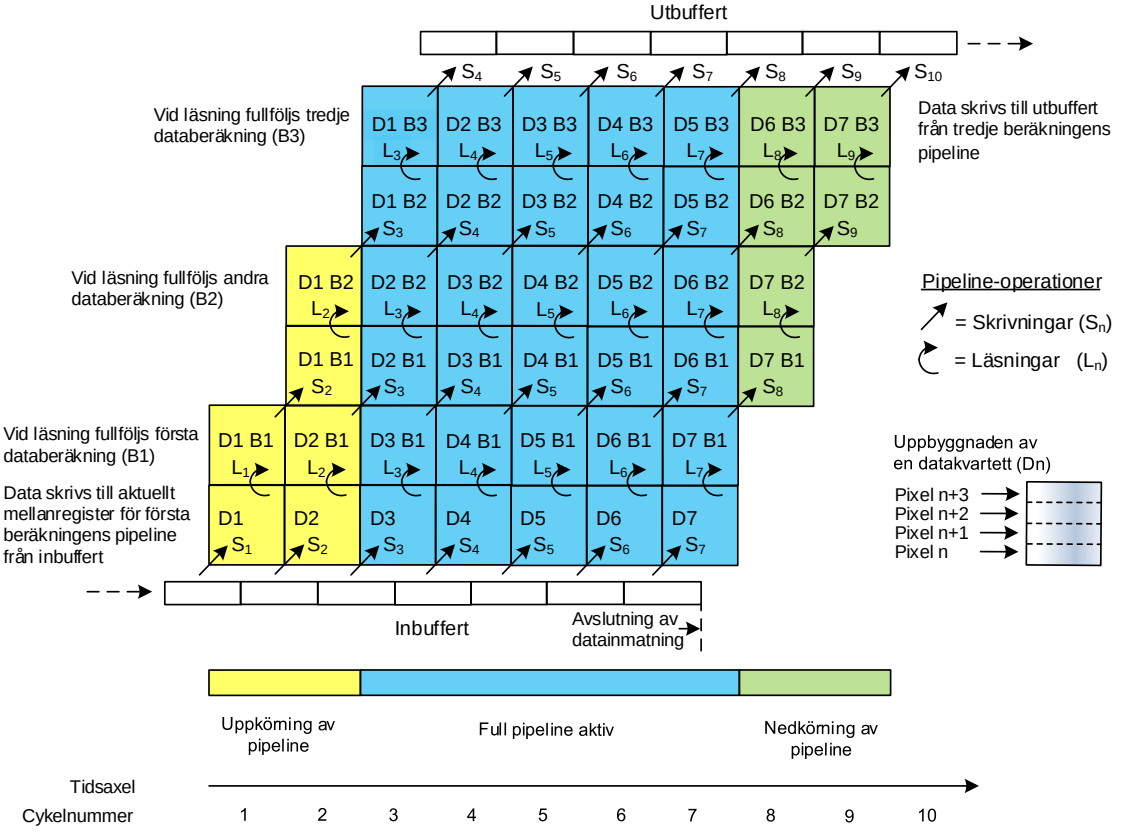


Fig. 14. Pipeline-exekvering där tre beräkningar på fyra pixel kan göras samtidigt när full pipeline är aktiv.

Vid läsning fullföljs samtidigt första beräkning (B1) på kvartett D3, andra beräkning (B2) på kvartett D2 och tredje beräkning (B3) på kvartett D1. Nu har kvartett D1 genomgått alla tre beräkningar och kan skrivas till utbufferten. Detta förlopp fortsätter på

samma sätt med skrivning av datakvartett fyra (D4) in till aktuellt mellanregister för första beräkningens pipeline o.s.v. tills inga nya indata skrivs till ett mellanregister för första beräkningens pipeline. Efter detta följer nedkörningsfasen av pipeline-exekveringen under två cykler. Man kan bygga valfri höjd på pipeline och därmed göra valda beräkningar på valfritt antal data samtidigt. Den begränsande faktorn är resurser av HW-celler i FPGA. För information om principfunktion hos en dynamisk pipeline hänvisas läsaren till kapitlen 3.7.1 och 3.7.2.

5.5 Introduktion till mjukvaran

Mjukvaran för valt utvecklingskort kan skapas dels genom Nios II Software Build Tools (SBT), som skapar en ny bottendatabas för mjukvara eller dels utnyttja en befintlig bottendatabas för mjukvara för ett typiskt projekt, som modifieras enligt systemets behov. Det sista alternativet väljs, dock begränsar det flexibiliteten och utvecklingsmöjligheterna i någon mån. Läsaren hänvisas till (Altera, 2014k), (Altera, 2010a) och (Altera, 2014l) för fördjupning i uppbyggandet av en helt ny SW-bottendatabas.

Eftersom hårdvaran är huvudintresset i denna avhandling väljs delar från mjukvaran hos applikationen Simple Socket Server med mjukvarukomponenten NicheStack TCP/IP Stack och operativsystemet µC/OS-II, som en lösning för en SW-bottendatabas. Den nya mjukvarulösningen presenterar en tillämpning av en Telnetserver, som smidigt implementerar styr- och bildbehandlingskommandon med TCP- och UDP-transmission av data in/ut på en 1 Gbps Ethernet-buss. Denna helhetslösning är anpassad till Terasics ALTERA DE3-utvecklingskort med Alteras Stratix III FPGA, en hårdvaru- och optimerad mjukvarulösning, intern Avalon-buss och extern Ethernet- och JTAG-buss.

5.6 Mjukvarufunktioner

Man kan grovt dela in mjukvaran i fyra delar som *system start-up, operativsystem, bildbehandling* och *avbrottsrutiner*. Går man djupare in i respektive del (och på vilken nivå genomgången av mjukvaran även görs) så inkluderar system start-up-delen System reset

och Start-up och Systeminitialisering. Operativsystemdelen inkluderar µC/OS-II RTOS och Grundtaskar. Bildbehandlingsdelen omfattar en Ethernet-task samt en JTAG-task. Avslutningsvis inkluderar delen för avbrottsrutiner timer- och DMA-avbrott. Avbrottsrutin för JTAG-enheten tas inte upp i avhandlingen.

5.6.1 Mjukvarans uppbyggnad

Helheten av mjukvaran är en C/C++- och Assembler-programvara för ett Nios II/f-processorsystem.

Denna inkluderar en 1 Gbps Ethernet Server för bildbehandling med socket-hantering via kanal 1 på ett HSMC-NET-dotterkort, som är kopplat till en Ethernet-buss. Dotterkortet är kopplat till ett Terasics ALTERA DE3-utvecklingskort, som har en JTAG-buss och är kopplad till en värddator.

Principfunktionen hos mjukvaran i bildbehandlingsplattformen är att man från värddatorn på Ethernet-bussen via Ethernet-MMI sänder TCP-styr- eller -bildbehandlingskommandon eller styr- eller bildbehandlingsvalskommandon på JTAG-bussen via JTAG-MMI till IPS. IPS svarar på respektive buss till respektive MMI genom kvittering på sänt kommando. Vid Ethernet-MMI TCP-styr- eller -bildbehandlingskommandon sänder IPS tillbaka behandlat bilddata antingen i TCP- eller UDP-format, som värddatorn presenterar.

Mjukvaran använder sig av NicheStack TCP/IP Stack och µC/OS-II i multi-thread-miljö. NicheStack TCP/IP Stack ger stöd för avbrottsbaserad drivrutinmiljö för hantering av Ethernet-gränssnitten på dotterkortet. (Altera, 2011e, s. 19) Kommunikation sker med TCP-, UDP- eller ASCII-kommunikation mellan IPS (som server) och värddator (som client). I figur 15 visas mjukvaran, som innehåller processorns initiering, det inbyggda systemets initiering, operativsystemets initiering, tre avbrottsrutiner och fyra användartaskar.

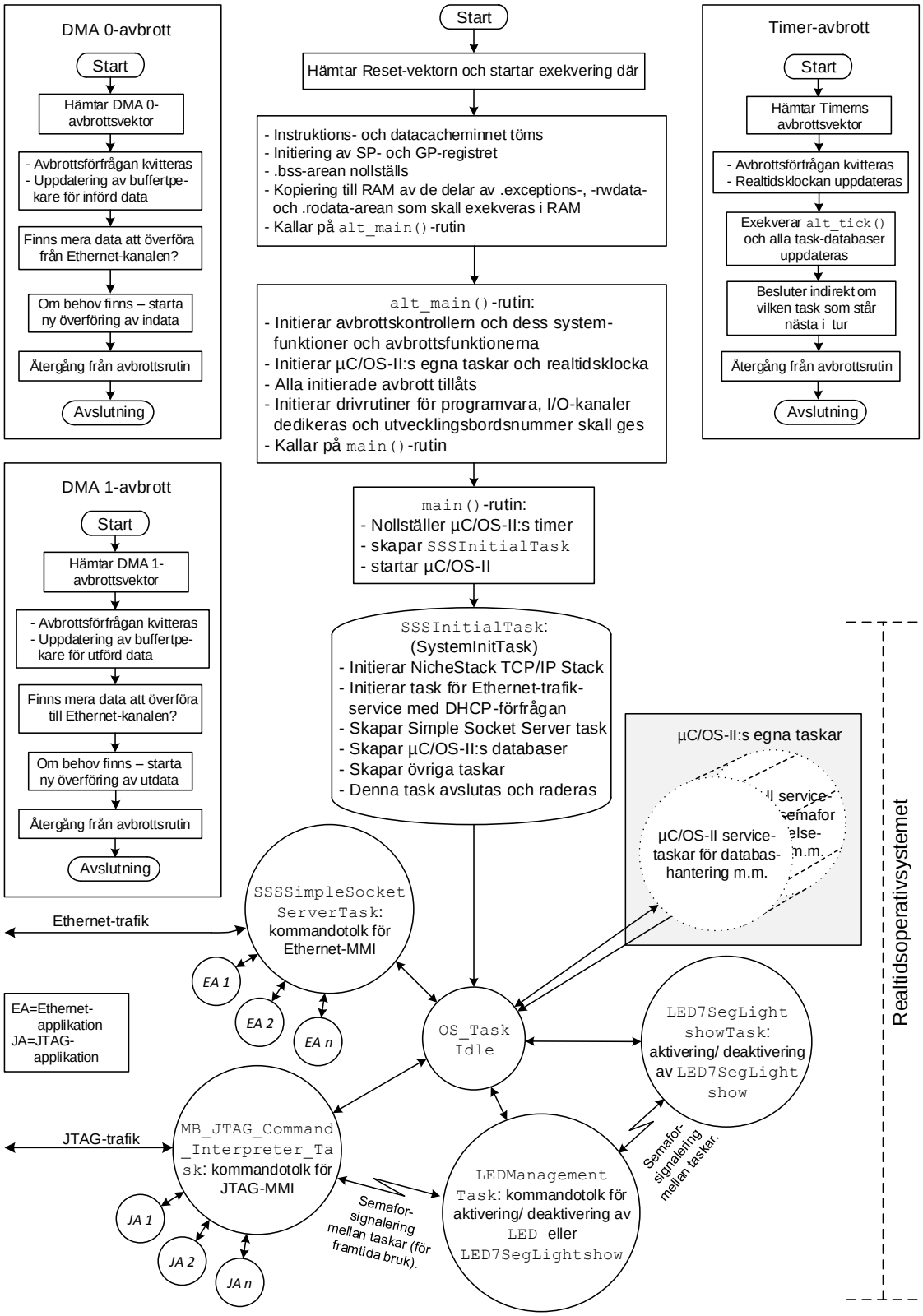


Fig. 15. Logisk bild på blockuppbyggnaden hos mjukvaran med start från reset-läge, operativsystemet med taskar, kommunikationsbussar och avbrottsrutiner.

Vidare finns en 0-task (`OS_Taskidle`) inkluderad i µC/OS-II och operativsystemets egna taskar för databas-, task-, kö- och semaforhantering m.m., som inte berörs i avhandlingen i detalj. Om ingen annan task finns att exekvera så exekverar processorn 0-tasken. Angående tasken SystemInitTask hänvisas läsaren till kapitel 5.6.2. För mera information om DMA 0-, DMA 1- och timer-avbrott hänvisas läsaren till kapitlen 5.6.7 och 5.6.8.

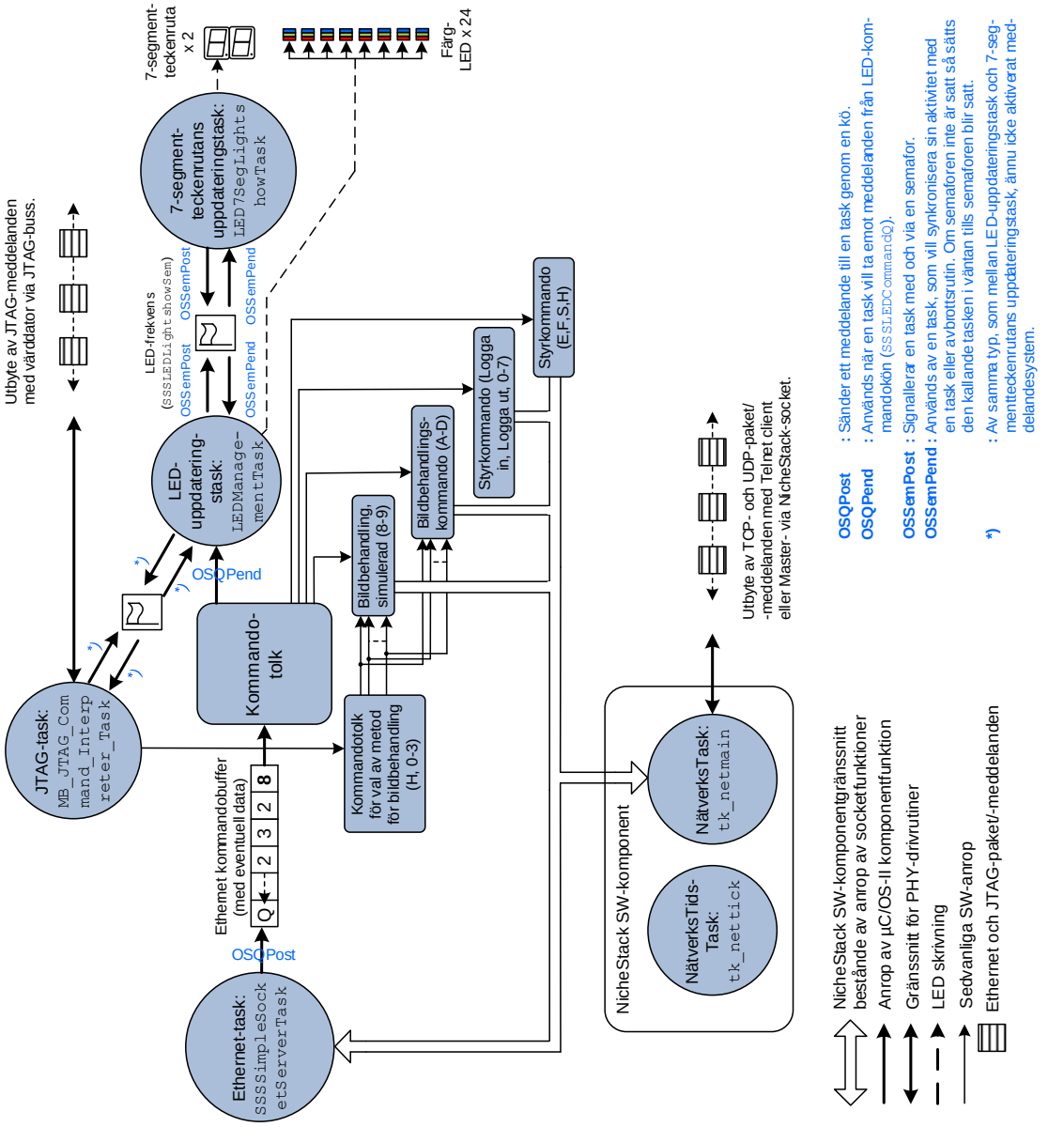


Fig. 16. Kommunikations- och helhetsstrukturen i mjukvara och systemfunktioner efter start och initiering. Anpassad från (Altera, 2011e, s. 19) och kompletterad av författaren.

I figur 16 visas helsystemfunktionen efter initiering och start av mjukvaran med tanke på taskarna och hanteringen av Ethernet- och JTAG-meddelanden i IPS. (Altera, 2011d), (Altera, 2007b) och (Altera, 2007c)

En Ethernet-task finns, som inkluderar en kommandotolk för styr- och bildbehandlingskommandon på Ethernet-bussen. Dessa kommandon, som ges från värddatorns Ethernet-MMI är antingen styr- eller bildbehandlingskommandon. Vid dessa styrs kommando med eventuell data till Ethernet-taskens kommandotolk för antingen simulerad bildbehandling, återkastande bildbehandling eller återkastande bildbehandling med prestandatest. För mera information se kapitlen 5.6.5.

En JTAG-task finns, som inkluderar en kommandotolk för styrkommandon och för val av bildbehandlingsmetod på JTAG-bussen. Vid styr- eller bildbehandlingsvalskommandon givna från värddatorns JTAG-MMI skickas kommandona via JTAG-bussen till JTAG-tasken för tolkning av kommando. För mera information se kapitel 5.6.6. För hantering av utvecklingskortets lokala MMI finns en LED-uppdateringstask, som sköter trafiken och signaleringen till och från 7-segmentteckenrutans uppdateringstask och JTAG-task.

5.6.2 System reset och start-up

Efter att mjukvaran laddats ned till programminnet går processorn vid system reset-läge in i reset-avbrottsrutinen där processorn startar att exekvera initieringskoden. Stack- och globala pekarregistret initieras, ett minnessegment i RAM nollställs och några minnessegment kopieras till RAM. Slutligen kallas `alt_main()`-rutinen på. (Altera, 2014m) För mera information om hur Nios II SBT allokerar och bygger upp minnet och minnesareorna och angående reset-läge och start-up (uppstartning) av systemet hänvisas läsaren till kapitel 12.1.5 i (Chu, 2012).

I `alt_main()`-funktionen skapas en dynamisk miljö med fullständig kontroll över startsekvensen, anrop av HAL-funktioner och initiering av systemet före start av ordinarie applikation. Vidare initieras hårdvarans avbrottskontrollenhet, dess systemfunktioner och avbrottsrutiner, operativsystemets egna taskar och realtidsklocka och alla avbrott

tillåts. Efter detta initieras mjukvarans drivrutiner och C-standard-I/O-kanalerna dedikeras. Till slut efterfrågas utvecklingskortets nummer (unikt nummer för varje utvecklingskort), som kvitteras via JTAG-MMI och `main()`-rutinen kallas på.

I `main()`-rutinen nollställs först operativsystemets timer, SystemInitTask skapas och operativsystemet startas. När operativsystemet startats söker det efter task med högsta prioritet (SystemInitTask). I denna startas exekvering på task-nivå. För vidare information om denna task hänvisas läsaren till 5.6.4. Innan plattformen är klar för användning exekveras en DHCP-sökningssekvens. Här söker NicheStack TCP/IP Stack efter IP-adress till en DHCP-server. Om inget svar erhålls, uppstår DHCP-timeout och IPS tilldelas en statisk IP-adress. Nu är IPS och Ethernet-bussen klar för användning och lyssnar på inkopplingsförfrågan (*Logga in*) från en client på den initierade IP-adressen och port 30 (vilka meddelas i JTAG-MMI).

För mera information om funktionerna ovan hänvisas läsaren till `alt_sys_init.c`-filen och kapitlen ”Developing Device Drivers for the Hardware Abstraction Layer” och ”Nios II Software Build Tools” i (Altera, 2010a). För socket- och nätverksprogrammering hänvisas läsaren till (Stevens W. R., 1988) och (Donahoo & Calvert, 2009) och för information om minnesuppbyggnaden hänvisas läsaren till (Altera, 2010a).

5.6.3 Operativsystem

I mjukvaran används operativsystemet µC/OS-II. Varje task tilldelas en tidsdel och en cirkulär hanteringsordning. (Arpaci-Dusseau & Arpaci-Dusseau, 2014) och (Kleinrock, 1964)

I figur 17 visas olika delar av mjukvaran, som kan indelas i applikationsprogram, processoroberoende mjukvara med µC/OS-II, applikationsspecifik mjukvara med µC/OS-II och µC/OS-II portad mjukvara med specifikt vald processor- och Timer-enhet.

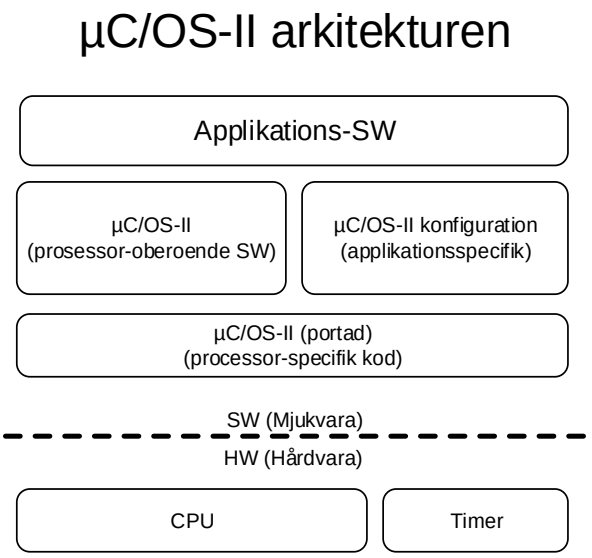


Fig. 17. Uppbyggnaden av operativsystemet och mjukvaran och deras relationer. Anpassad från (Kremer, 2009) och modifierad av författaren.

Vid task-byte sparar operativsystemskärnan den aktuella taskens status (bl.a. CPU-register) i dess databas.

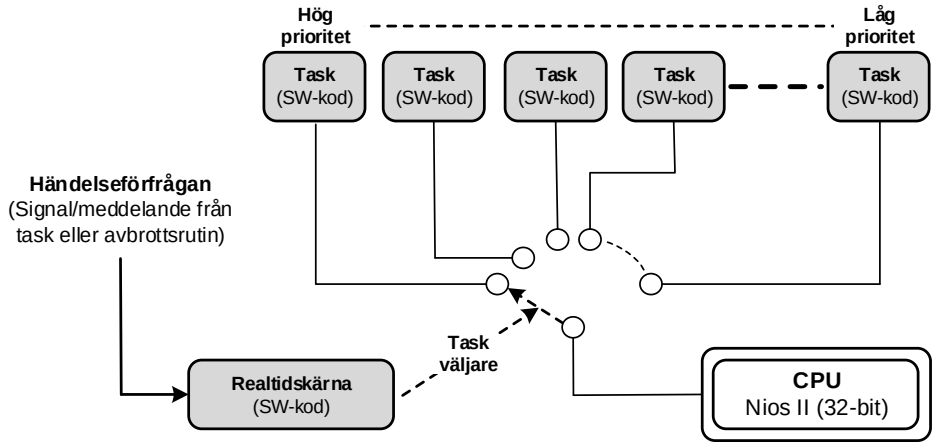


Fig. 18. Sambandet mellan timer-avbrott och taskbyte. Operativsystemet väljer den task, som står högre i prioritet framom task med lägre prioritet. Anpassad från (Labrosse, 2014) och modifierad av författaren.

Beroende på premisser angående tid och prioritet (figur 18) byter OS-kärnan task genom att ladda nästa tasks status från dess databas och fortgår exekvera denna nya task där den senast avbröts.

En task kan stå i lägena *Inaktiv*, *Klar*, *Körs*, *Väntar* och *Avbrottsrutin körs*. Mjukvaran körs synkroniserad med operativsystemet och avbrottsrutinerna. Styrfunktioner till operativsystemet kan t.ex. ges via taskar, applikationsprogram och avbrottsrutiner eller från operativsystemet själv. För mer information om µC/OS-II hänvisas läsaren till (Labrosse, 2002b, ss. 405–532) och (Cools, 2010).

5.6.4 Grundtaskar

Det finns fem grundtaskar i bildbehandlingsplattformen. Den första grundtasken, SystemInitTask, initierar NicheStack TCP/IP Stack-mjukvarukomponentens och operativsystemets strukturer och taskar. Den andra grundtasken, NätverksTask, sköter hanteringen av nättrafiken. Den tredje grundtasken, NätverksTidsTask, sköter tidshanteringen av nätverksstacken. Den fjärde grundtasken, LED-uppdateringstasken, läser data från LED-kommandokön och tolkar dessa. Den femte grundtasken, 7-segmentteckenrutans uppdateringstask, sköter uppdatering och visning (show) på den högra 7-segmentteckenrutan.

Första grundtasken (SystemInitTask) (figur 15) instansierar och initierar µC/OS-II-operativsystemets källor och strukturer. Tasken måste ha högsta prioritet. Läsaren hänvisas till (Altera, 2010a) för mera information angående orsak. Här initieras och startas NätverksTask för NicheStack TCP/IP Stack-servicen och NätverksTidsTask för Ethernet-trafikservice med DHCP-förfrågan. Efter detta skapas Ethernet- och JTAG-tasken, operativsystemets databaser och övriga grund- och bildbehandlingstaskar initieras. Slutligen avslutar och raderar tasken sig själv. För mera information om NätverksTask och NätverksTidsTask hänvisas läsaren till andra och tredje grundtasken respektive. För ytterligare information angående innehållet i SystemInitTask hänvisas läsaren till figur 15 och (Altera, 2011e, ss. 1–19).

Andra grundtasken (NätverksTask) (p.g.a. utrymmesbrist inte med i figur 15), initierar nätverksstacken med nätverksgränssnitt. Efter att nättrafiken startats sätts denna task i en loop för hantering av mottagna datapaket. Vid inkommen data på Ethernet-bussen startas DMA 0-kanalens dataöverföring genom initiering av dess register med parametrar för längd av data, slutadress i dedikerat minne m.m. DMA-kanalens överföring startas genom skrivning till dess kontrollregister. Då ett datapaket mottagits i mottagarkön `rcvdq` (i RAM) väcks den sovande taskens loop som börjar hantera datapaket i kön.

Tredje grundtasken (NätverksTidsTask) (p.g.a. utrymmesbrist inte med i figur 15), initierar NicheStack TCP/IP Stack för användning med RTOS. Tasken används för tidshanteringen av nätverksstacken. Fjärde grundtasken (LED-uppdateringstasken) (figur 15) läser data från LED-kommandokön och tolkar dessa. Efter tolkningen sköter tasken om (beroende på kommando) övervakning och uppdatering av växling hos åtta blå LED mellan aktivt eller inaktivt läge, signalering via semafor åt 7-segmentteckenrutans uppdateringstask om aktiv eller inaktiv visning av show i 7-segmentteckenrutan eller signalering via semafor med JTAG-task.

Sista grundtasken (7-segmentteckenrutans uppdateringstask) (figur 15) styr och sköter uppdatering och visning av ljusshow på den högra 7-segmentteckenrutan. När visningen är aktiv väljer den ett slumpmässigt mönster och uppdaterar detta på teckenrutan. Tasken hanterar signalering mellan sig och LED-uppdateringstasken via en semafor. Vid uppdateringen av visningen på teckenrutan kontrolleras om det finns en förfrågan via semaforen. Om detta är fallet läses denna och växlar sen mellan aktiv eller inaktiv visning på teckenrutan.

5.6.5 Ethernet-task

Tasken sköter inläsning och tolkning av kommandon från Ethernet-bussen och exekvering av de applikationer, som finns kopplade till kommandona och bildbehandlingen. Vid kommandon givna via Ethernet-MMI i värddatorn får NicheStack TCP/IP Stack i plattformen ett Ethernet-paket, som innehåller ett styr- eller bildbehandlingskommando. NicheStack TCP/IP Stack processar de inkommande Ethernet-paketen med TCP. Ethernet-taskens kommandotolk kallas nu på från tasken och extraherar aktuellt kommando

och eventuella data i bufferten. Beroende på kommando och efterföljande data styrs man till aktuell procedur, funktion eller task.

Vid styrkommandot *Logga in* sker en inloggning på plattformen med handskakning och det fastställs parametrar och regler för kommunikationskanal som upprättas. Efter att bildbehandlingsplattformen och värddatorn etablerat kontakt på deras respektive IP-adresser kan TCP-trafik utbytas. Ifråga om UDP-trafik kan även trafik utbytas efter att IPS och värddatorn etablerat kontakt på deras respektive IP-adress och IPS skapat en UDP-socket, som satts att lyssna på port 10.000, som även värddatorn använder. Vid styrkommandot *Logga ut* sker en utloggning från plattformen och det skapas en informationstext, som skickas tillbaka på Ethernet-bussen och till Ethernet-MMI. Efter detta stängs den upprättade Ethernet-förbindelsen mellan värddator och plattform.

Vid styrkommandona *0–7* styrs data till LED-uppdateringstasken. Kommandots ASCII-kod omvandlas till ett binärt värde, som syftar på en unik blå LED i utvecklingskortet. Detta skrivs direkt till adressen för aktuella PIO och LED. Vid skrivning sätts denna LED och PIO-utgång varannan gång aktiv eller inaktiv. Se kapitel 5.6.4 för mera information. Vid styrkommandot *E* skapas en UDP-socket i IPS, som startar att lyssna på port 10.000 och UDP-meddelanden på Ethernet-bussen. Vissa grundtaskar blockeras även i samband med detta kommando. Vid styrkommandot *F* avbryts detta lyssnande på Ethernet-bussen och de blockerade grundtaskarna startas åter. Vid styrkommandot *S* styrs data till LED-uppdateringstasken, som skickar en signal via en semafor till 7-segmentteckenrutans uppdateringstask. Denna antingen startar eller avslutar visning på högra 7-segmentteckenrutan. Se kapitel 5.6.4 för mera information. Vid styrkommandot *H* styrs data till mjukvara, som skapar en hjälptext för användaren och skickar denna tillbaka till värddatorn på Ethernet-bussen och Ethernet-MMI.

Det finns två huvudkategorier av bildbehandling i IPS. Vid den första typen *simuleras* bildbehandling, som aktiveras via kommandona *8* och *9* givna via Ethernet-MMI från värddatorn. Vid dessa kommandon skapas i IPS en *känd teststräng*. Denna kopieras först in till utströngen varefter samma teststräng igen kopieras till utströngen men nu gående genom bildbehandlingsacceleratorn, som är aktiverad (enligt vald beräknande

metod). Efter detta sänds hela utsträngen med respektive protokoll till värddatorn med hjälp av `send()`- eller `sendto()`-funktionen i HAL API. (Stevens & Rudoff, 2004) Vid bildbehandlingskommando *8* sänds data tillbaka på Ethernet-bussen och Ethernet-MMI som ett UDP-meddelande och vid bildbehandlingskommando *9* sänds data tillbaka på Ethernet-bussen och Ethernet-MMI som ett TCP-meddelande.

Vid den andra typen av bildbehandlingskommandon, *återkastande bildbehandling*, sänds kommandon med (via Ethernet-MMI) *fritt givna* eller *av testprogrammet valda* data på Ethernet-bussen från värddatorn. Vid bildbehandlingskommandona *A* och *B* extraheras inkommande data efter kommando i IPS. Data efter kommando styrs genom bildbehandlingsenheten, som är inställd inaktiv under första halvan av utgående dataströmmen och aktiv (enligt vald beräknande metod) under andra halvan av utgående dataströmmen. Efter detta sänds hela utsträngen med respektive protokoll till värddatorn med hjälp av `send()`- eller `sendto()`-funktionen i HAL API. (Stevens & Rudoff, 2004) Vid bildbehandlingskommando *A* sänds data tillbaka till Ethernet-MMI omvandlat, som ett TCP-meddelande och vid bildbehandlingskommando *B*, som ett UDP-meddelande.

Bildbehandlingen av data är en beräknande funktion och representeras av en komponent konstruerad i VHDL eller annat HDL-språk. En komponent instansieras mot bildbehandlingsacceleratorns gränssnitt. Mjukvaran kommunicerar emot detta gränssnitt genom Avalon-bussen på gränssnittets adress i minnesrymnden. Läsaren hänvisas till figur 10 för att se koppling mellan HW-kärna för bildbehandling och HW-kärna för val av bildbehandlingsmetod. Beräkning på data sker via två buffertar, dels den vänstra inbufferten till och dels den högra utbufferten från bildbehandlingsacceleratorn. Data som skall behandlas samlas i inbufferten. Behandlingen startas genom att man först läser bilddata från inbufferten och skriver in denna till acceleratorn på dess adress. Behandlingen av data slutförs när man läser ut data från acceleratorn på dess adress och skriver in denna till utbufferten. Runt gränssnittet finns insatta HW-kärnor med VHDL-komponenter (`COMPONENT`) eller motsvarande HDL-syntax, som utför fritt valda enkla

eller komplexa beräkningar på genomströmmande data. Se kapitel 5.6.6 vid byte av metod.

Vid *C(3)*-, *C(15)*-, *D(3)*- och *D(15)*-kommandot aktiveras en återkastande bildbehandling med prestandatest. Vid dessa kommandon sänds *av testprogrammet vald data* på Ethernet-bussen från värddatorn. För att använda dessa måste *E*-kommandot först aktiveras. Vid kommandona bygger värddatorn upp och sänder *allt längre och längre strängar* av data (från 1 t.o.m. 1458 nyttodata) i grupper på 3 stycken lika längd (vid *C(3)*- och *D(3)*-kommandot) eller i grupper på 15 stycken lika längd (vid *C(15)*- och *D(15)*-kommandot) efter varandra varande strängar. Vid dessa extraherar IPS data efter kommando och styr denna genom bildbehandlingsenheten, som är inställd aktiv för utgående dataström. Vid *C(3)*- och *C(15)*-kommandot sänder värddatorn strängar med ”2” till IPS och data sänds tillbaka på Ethernet-bussen som TCP-meddelanden. Vid *D(3)*- och *D(15)*-kommandot sänder värddatorn strängar med ”4” till IPS och data sänds tillbaka på Ethernet-bussen som UDP-meddelanden. Man avslutar testerna med F-kommandot.

Man måste vara inloggad för att använda dessa kommandon (förutom vid kommandona *E* och *F*). Vid styrkommandona (*Logga in*, *Logga ut*, *0–7*, *E*, *F*, *H* och *S*) och bildbehandlingskommandona (*8, 9, A och B*) kvitteras givet kommando tillbaka till Ethernet-MMI i värddatorn. Bildbehandlingskommandon med prestandatest (*C* och *D*) kvitteras även men är osynliga i Ethernet-MMI. I Ethernet-MMI finns räknarregister för sända och mottagna meddelanden. Då sista meddelande anlänt skrivs antalet sända och mottagna meddelanden samt längd på nyttodata i sista meddelandet ut. Räknarregistren adderas ständigt. Man kan även spara undan testresulten i Excel-filer (med filnamnen ”TCP test results.xls” och ”UDP test results.xls” som standard) från respektive utfört test genom att aktivera kontrollen till höger om respektive kontroll för start av snabbtest i Ethernet-MMI.

5.6.6 JTAG-task

Denna task sköter om inläsning av och innehar en kommandotolk för kommandon, som ges eller tas emot på JTAG-bussen via JTAG-MMI i värddatorn. Efter tolkning av ett kommando kallas på och exekveras de applikationer, som startats med hjälp av kommandot. Tasken sänder med jämna mellanrum texten ”Ge kommando:” till värddatorns JTAG-MMI. Denna task har en semafor reserverad för framtida bruk. Semaforen avlyssnas och hanteras f.n. endast på primär nivå i LED-uppdateringstasken, som JTAG-task då kan kommunicera med.

Styrkommandon och kommandon från JTAG-MMI ges i värddatorn och skickas seriellt via JTAG-bussen till IPS. De innehåller styrkommando eller kommandon för bildbehandlingsmetod med eventuellt data. JTAG-bussen har ett ASCII-kommunikationsprotokoll. I kommandotolken extraheras inkommande kommando och eventuellt data från bussens inbuffert. Beroende på kommando och data styrs man till aktuell procedur och funktion i mjukvaran eller aktiverar olika beräkningsfunktioner i bildbehandlingsenheten. Det finns f.n. styrkommandot *H* och kommandona *0–3* för val av bildbehandlingsmetod. Kommandon givna för val av bildbehandlingsmetod gäller både vid simulerande och återkastande bildbehandlingskommandon givna från Ethernet-MMI. Funktion via JTAG-MMI, som kallas på och aktiveras kvitterar givet kommando till JTAG-MMI. Efter kvittens getts skrivs det givna kommandots binära representation in till HW-kärnan för val av bildbehandlingsmetod på dess adress. Själva bytet av bildbehandlingsmetod sker vid läsning av samma kärna.

Den valda matematiska bildbehandlingsmetoden är i kraft tills en ny godkänd matematisk metod ges eller ström bryts till IPS. Valet ”ingen matematisk (beräknande) bildbehandlingsmetod” (*0*) är standard som behandlingsmetod vid start av systemet. Vid kommandon för bildbehandlingsmetod kan beräknande bildbehandlingsmetod vara ”ingen matematisk bildbehandlingsmetod” (*0*), ”addition, som matematisk bildbehandlingsmetod” (*1*) (vilket betyder en addition med 1 pixel per pixel), ”subtraktion, som matematisk bildbehandlingsmetod” (*2*) (vilket betyder en subtraktion med 1 pixel per pixel) och ”multiplikation med två, som matematisk bildbehandlingsmetod” (*3*) (vilket betyder en multiplikation med 2 pixel per pixel). Vid styrkommandot *H* skapas en hjälp-

text för användaren. Hjälptexten skickas tillbaka på JTAG-bussen i ASCII-format till JTAG-MMI.

5.6.7 DMA

Det finns två DMA IP-kärnor i IPS. Dessa finns tillgängliga i Qsys-biblioteket. Den ena av två IP-kärnorna är DMA 0 `(sgdma_rx)` och den andra är DMA 1 `(sgdma_tx)`. DMA 0 används när data överförs från Ethernet-kretsen (kanal 1) på dotterkortet till RAM och DMA 1 när data överförs från RAM i IPS till Ethernet-kretsen (kanal 1) på dotterkortet. Genom att vissa RAM-areor är dedikerade för in- och utgående data så kan överföringar fullföljas utan att processorn stör överföringarna. (Altera, 2015e) DMA 0 har en utgående avbrottsignal kopplad till Nios II/f-processorn (se bilaga 1). Den aktiveras när IP-kärnan har överfört programmerad kommunikations- och bilddata från Ethernet-kretsen. DMA 1 har en utgående avbrottsignal kopplad till Nios II/f-processorn (se bilaga 1), som aktiveras när DMA 1 har överfört programmerad kommunikations- och bilddata till Ethernet-kretsen. När respektive avbrottssignal aktiveras går processorn in i respektive avbrottsrutin. Läsaren hänvisas till figur 15 (block för DMA 0- och DMA 1-avbrott) för fördjupning i funktionen hos avbrottsrutinerna.

I respektive avbrottsrutin uppdateras de respektive kanalernas buffertpekare för överförd data och avbrottsförfrågan från respektive IP-kärna kvitteras. I DMA 0:s avbrottsrutin kontrolleras sedan Ethernet-kretsens statusregister om ytterligare data finns att överföra från Ethernet-kretsen till RAM. Om så är fallet startas en ny DMA-överföring av data till RAM. Efter detta signalerar mjukvaran till kommandotolken att starta tolkning av inkommet kommando och eventuellt data. I DMA 1:s avbrottsrutin kontrolleras buffertpekarna om det finns ytterligare data i utbufferten att överföra till Ethernet-kretsen. Om så är fallet startas en ny DMA-överföring av data från RAM. Om ingen ny respektive DMA-överföring behövs går processorn ur respektive avbrottsrutin. För fördjupning i DMA-kärnorna hänvisas läsaren till (Altera, 2015e)

5.6.8 Timer

I IPS finns en Timer IP-kärna. Denna finns tillgänglig i Qsys-biblioteket. Vid timer-avbrott går processorn till timer-avbrottsrutinen, som bearbetar data vid bestämda tidpunkter. Timer-avbrottsrutinen stegar bl.a. fram operativsystemets process- och task-klockor, register samt alarm men håller även ordning på andra realtidskopplingar, hos systemet. Läsaren hänvisas till figur 15 (block för timer-avbrott) för fördjupning i funktionen hos timer-avbrottsrutinen. Vid timer-avbrottet kallas `alt_avalon_timer_sc_irq()`-funktionen på, som först kvitterar avbrottsförfrågan och kallar sedan på funktionen `alt_tick()`. Denna signalerar åt systemet att ett timer-tick har inträffat.

Funktionen `alt_tick()` stegar fram operativsystemets tick-klocka och uppdaterar alla alarmregister i OS. Läsaren hänvisas till `alt_tick.c`-filen för ytterligare detaljer. I och med att hela databasen för alla taskar med respektive realtidsklockor stegas framåt och uppdateras görs här även beslut om vilken task som står i tur att exekveras m.m. Om annan task står i tur att exekveras byter operativsystemet nästa tasks status till klar för exekvering före processorn lämnar `alt_tick()`-funktionen och timer-avbrotts-rutinen. Varje task står i en oändlig loop. Timer-avbrott i µC/OS-II behandlar även task, som t.ex. har suspenderats (blockerats) av någon anledning m.m. (Holenderski, 2014)

6 TESTMETOD OCH -RESULTAT

I detta kapitel genomgås och visas prestandatester och testresultat med bildbehandlingsplattformen. Det visas hur testsystemet är sammankopplat under testerna mellan plattform och värddator och hur det egenutvecklade IPS testsystem-programmet är uppbyggt. Avslutningsvis analyseras resultaten, plattformens resurseffektivitet och påverkan av olika optimeringar.

6.1 Ambitioner och förväntningar

Det finns olika typer av tester man kan utsätta ett system för. Dessa är belastningstest (systemets belastningstest), stresstest (systemets robusthet vid överbelastning), uthållighetstest (systemets uthållighet vid belastning), peaktest (systemets reaktion för plötslig ändring i belastning), konfigureringstest (systemets reaktion vid upprepning av återkonfigurering) och isoleringstest (systemets reaktion vid upprepning av systemstart). (Molyneaux, 2014)

Med IPS-plattformen fullföljs belastningstester. Ambitionen med testerna är att visa att plattformen kan ta emot nyttodata i ett TCP-meddelande från en enhet i Ethernet-rymden, omvandla denna och sända tillbaka denna antingen som ett TCP- eller UDP-meddelande i realtid. Vidare visas vilka funktions- och prestandakriterier systemet uppfyller och var gränsen går för dess maximigenomströmning. Med testerna undersöks, mäts, valideras och kontrolleras attribut i systemets kvalité och att programmet fungerar korrekt.

6.2 Testutrustning

Innan testerna kan utföras behövs en PC med operativsystemet Windows 7 och Ethernet-gränssnitt, Terasics ALTERA DE3-utvecklingskort med externt Ethernet-kort och plattformens mjukvara. Vidare behövs plattformens och PC:s IP-adress, Eclipse- (v.

11.0) och Alteras Quartus-programvaran (v. 11.0), Microsofts Visual Studio (v. 2013) med IPS testsystem-programmet och en USB-kabel. Avslutningsvis behövs:

- två rakkopplade Ethernet-kablar och en 1 Gb switch (fall 1) eller
- en korskopplad Ethernet-kabel (fall 2)

I fall 1 kopplas den ena Ethernet-kabeln fast mellan IPS-plattformen och 1 Gb switch:en och den andra Ethernet-kabeln fast mellan PC och 1 Gb switch:en. I fall 2 kopplas den korskopplade Ethernet-kabeln fast mellan plattformen och PC. USB-kabeln kopplas fast mellan PC:s USB-gränssnitt och plattformens JTAG-gränssnitt.

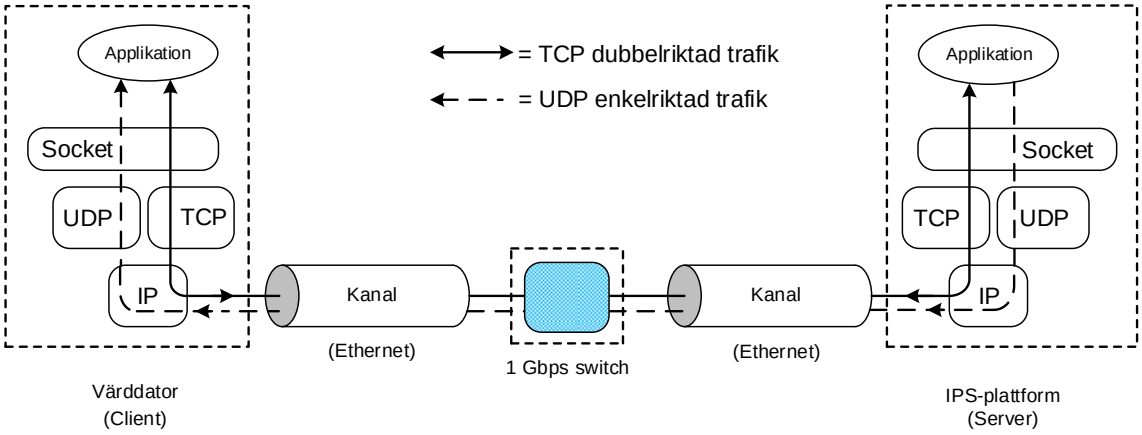


Fig. 19. Sammankopplingen vid TCP- och UDP-tester på bl.a. kommunikationsnivå i fall 1 (med switch i streckad linje) och fall 2 utan switch.

I figur 19 visas sammankopplingen som en samlad bild av olika delkomponenter angående funktioner och kommunikation under de olika testmetoderna, som används i testsystemet i plattformen. Figuren visar även hur testmeddelanden rör sig på Ethernetbussen och de olika protokoll-, socket- och applikationsnivåerna mellan värddator och plattform vid TCP-tester (via den heldragna linjen) och UDP-tester (via den streckade linjen). Vid testerna är PC och IPS-plattformen sammankopplade enligt figur 2 på sidan 23.

6.3 Testmetrik

För att beräkna den verkliga datagenomströmningen, efter att data sänts iväg, mäts den tid det tar tills nyttodata kommer tillbaka omvandlat i TCP- eller UDP-meddelandenas dataområde. Genomströmningen visar omvandlat data i relation till använd tid efter att data sänts iväg genom alla delar innehållande överföring, kommunikation, kommando-tolkning, databehandling, tillbakasändning och erhållande av svar via aktuell socket. Värddatorn väntar på svar från plattformen innan den sänder nästa teststräng (se pseudokod i figur 20).

Genomströmningen per meddelande är:

$$G_{str} = \frac{L_m \times 8 \times 10^9}{(t_m - t_s)},$$

Där G_{str} är genomströmning $(\mathrm{b/s})$, t_s är tid när testdata sänts från värddatorn (ns), t_m är tid när responsdata mottagits till värddatorn (ns), L_m är antal nyttodata i det sända meddelandet (B) och 8×10^9 är konstant för att omvandla genomströmning från $\mathrm{B/ns}$, till $\mathrm{b/s}$.

Om man räknar med vad som ytterligare omger data blir genomströmningen betydligt högre p.g.a. de olika lagren. Läsaren hänvisas till (Nobel, 2011) för fördjupning i 1 Gbps-trafikens uppbyggnad. Om alla lager tas med i ett TCP-meddelande blir summan 1538 B/ram med 1460 B data i datalagret. Sänder man t.ex. 1 B nyttodata blir summan minst 1538–1460+3 B (där sista term innehåller 1 B kommando, 1 B nyttodata och 1 B sluttecken) vilket leder till att det i verkligheten sänds 81 B på 1 Gbps Ethernet-bussen.

6.4 Testprogram

Starten av test med plattformen sker via Ethernet-bussens Man Machine Interface (MMI) i IPS testsystem-programmet. Detta exekveras i PC och är en mjukvara, som är

gjord för verifiering av funktioner i plattformen och för att man ska kunna testa att plattformen fungerar rätt.

6.4.1 Testprocedur

Vid kommandon för plattformens prestandatest aktiveras en återkastande bildbehandling. Värddatorn bygger upp och sänder ut växande längd (eller sjunkande längd, med små justeringar i IPS testsystem-programmet) på teststrängar av data i grupper på tre stycken lika långa strängar (låg belastning) efter varandra eller i grupper på 15 stycken lika långa strängar (hög belastning) efter varandra. Se vidare kapitel 5.6.5 angående testprocedurerna. Se tabell 1 för vilka specifika prestandatester som genomgås.

```
For (Test_sträng_längd=1 to 1458) {
  Fyll Teststräng med kommando+”2”:or+<lf> // om kommandot är c eller
  Fyll Teststräng med kommando+”4”:or+<lf> // om kommandot är d
  For (Antal_lika_strängar=1 to 3)         // om låg belastning eller
  For (Antal_lika_strängar=1 to 15){       // om hög belastning
    Teststräng sänds från IPS och realtidsklocka startas och noteras
    När svar fåtts från IPS stoppas realtidsklockan och noteras
    per post (/post) sparas nu i RAM-array:
      Postnummer, Meddelandetyp (TCP/UDP), Meddelandelängd (B)
      Sändningstid(ns), Mottagningstid (ns)
      Differenstid beräknas (ns) (mottagningstid-sändningstid)
      Genomströmning beräknas (bps) för testmeddelande }}
```

Fig. 20. Pseudo-kod över principfunktionen hos testsystemet (<lf> i teststrängen representerar slutttecknet, vilket är ASCII-koden för line feed, 10).

Alla tester består i princip av två for-loopar (se figur 20): en yttre for-loop, som sköter om *längden* av TCP-testmeddelanden, som sänds till plattformen och en inre for-loop, som sköter om *antalet* upprepningar av TCP-testmeddelanden, som sänds till plattformen. Vid start, i den inre for-loopen, konstruerar värddatorn ett testmeddelande och sänder detta till plattformen via Ethernet-bussen. Nu startar värddatorn en realtidsklocka, som noteras och sätts att lyssna på antingen ett TCP- eller UDP-svarsmeddelande från plattformen. När svarsmeddelande mottagits stoppas realtidsklockan som noteras och nu skrivs svarsmeddelandets post-nummer, typ, realtidsklock-

ans start- och sluttid, beräknad mellanskillnad mellan dessa, antalet sända data och beräknad genomströmning in i en RAM-array. Efter detta startas en ny process med konstruktion och sändning av nästa testmeddelande. När hela processen fullföljts meddelas användaren om detta och bereds att kunna lagra erhållna svars- och beräkningsvärden från databasen i RAM till en Excel-fil.

6.4.2 Erhållna testdata

Vid ett test informeras plattformen om vilken typ av operation, som skall utföras genom att teststrängen har ett specifikt kommando först i teststrängen och sist ett sluttecken (<lf>).

Efter att ett test genomförts finns all mätdata samlat i RAM. Detta kan överföras till en Excel-fil. *Per test* lagras först i Excel-filen datum och tid när detta test startats, rubrik för hela testet och rubriker för respektive parameters kolumn. Vid sparande av data i Excel-filen kommer sedan att *från varje* sändning och mottagning av teststräng från testet att samlas in postnummer, meddelandetyp (TCP eller UDP), meddelandelängd (B), sändningstid (ns) (efter att testmeddelandet sänts från värddatorn), mottagningstid (ns) (efter att testmeddelande mottagits till värddatorn), differenstid (ns) mellan sändnings- och mottagningstid och genomströmning för testmeddelandet (bps).

6.4.3 Utmaningar och lösningar vid prestandatester

I ett system där genomströmning testas måste all annan aktivitet, som kan störa processen, avbrytas. Endast de absolut nödvändigaste delarna får tillåtas och använda processorns kraft under testprocessen.

I plattformen kan avbrottsrutiner oväntat aktiveras och störa testerna. Under dessa tester tillåts därför endast timer- och DMA-avbrottet. Timer-avbrottet tillåts därför att det sköter drivningen av operativsystemet och taskarna, som sköter Ethernet-kommunikationen med hjälp av NicheStack-mjukvarukomponenten. DMA-avbrotten tillåts därför att de sköter om överföringen till och från Ethernet-kretsen på tilläggskortet. Före testerna, vid start av lyssnande på UDP-meddelanden, blockeras onödiga taskar och utskrivningar, som annars förbrukar processorns och operativsystemets krafter.

Klockan i värddatorns operativsystem är bas för mätningen av tid mellan sänt testmeddelande och mottaget responsmeddelande med omvandlat data. Timern i detta operativsystem har en resolution på 500 ns. (Microsoft, 2016) Under testprocessen måste en skrivning av erhållna data till en RAM-struktur ske med så få medlemmar som möjligt så att skrivningen av data kan göras på så kort tid som möjligt (då beräkning av plats i strukturen kan ta tid). Först efter att hela testprocessen har genomförts kan data i RAM-strukturen läsas och sparas i en Excel-fil. RAM-buffertarna läses med hjälp av två forloopar för att man lätt ska kunna ändra ordningsföljden vid överföring av data. Orsaken till att data hanteras emot en Excel-fil först efter testprocessen avslutats beror på att skrivning till skivminne är en mycket långsam process.

Allmänna tester har gjorts med plattformen ansluten till övriga servrar, Ethernet-enheter och värddator via switch. Dessa externa källor förbrukar dock även Ethernet-bussen då de håller sig uppdaterade med plattformens status o.s.v. Testerna utförda och redovisade i avhandlingen sker när plattformen endast är ansluten till värddatorn via en switch. Vid inloggning (handshake) byter värddatorn och plattformen information med varandra. Värddatorns MSS-värde är 1460 medan plattformens MSS-värde är 1458. Erhållande av tid från Ethernet-ramarna har övervägts. Men detta är ju fel medan denna tid inte berättar något om när data nått inbufferten i testprogrammet för att kunna kontrolleras eller hanteras. Denna metod ger ännu ingen relevant information om datans innehåll.

6.5 Utförda tester och deras parametrar

Tester utfördes med två protokoll, dels TCP och dels UDP. Alla kommandon som sänds från värddatorn sänds med TCP. I detta system där Ethernet-ramens MTU är 1500 är maxgränsen för sänd data i TCP-meddelandet 1460 i värddatorn. För mer information hänvisas läsaren till (IETF, 1981) och (Nobel, 2011). Därför sätts även den övre gränsen för längd på teststrängarna till detta värde. Testmeddelandena inkluderar 1 B kommando, maximalt 1458 B nyttodata (att omvandla) och 1 B sluttecken (<lf>). Andra egenskaper vid testerna är antal upprepningar av sändning av teststrängar för att kunna se om det finns likheter mellan testerna och för att kunna se om det finns likheter mellan

testerna över hela skalan av datalängder på teststrängar. Vid testerna skapas både stigande och sjunkande längd på teststrängarna för att kunna se om det finns likheter vid stigande och sjunkande längd på teststrängarna.

Tabell 1. Specifikt utvalda ordinarie tester med IPS testsystem-programmet, som fullföljts med plattformen.

Testnummer	Meddelandetyp	Datalängd	Variationstyp	N upprepningar
1	TCP	1–1458	Stigande	3 (låg belastning)
2	TCP	10–1	Sjunkande	3 (låg belastning)
3	TCP	1449–1458	Stigande	3 (låg belastning)
4	TCP	1458–1	Sjunkande	15 (hög belastning)
5	UDP	1458–1	Sjunkande	3 (låg belastning)
6	UDP	1–10	Stigande	3 (låg belastning)
7	UDP	1449–1458	Stigande	3 (låg belastning)
8	UDP	1–1458	Stigande	15 (hög belastning)

Sammanlagt utfördes 80 ordinarie tester och fem optimeringstester. Från åtta grupper med 10 tester i vardera (exklusive optimeringstesterna) valdes ut representativa kurvor av TCP- och UDP-tester med stigande eller sjunkande längd på teststrängarna och tre eller 15 upprepningar i testerna.

Alla tester har genomgåtts. Det kan konstateras att alla testkurvor visar lika uppbyggnad och trend oberoende om tester gjorts med TCP eller UDP eller med stigande eller sjunkande längd på teststrängarna. Därför redovisas endast några testresultat med hög eller låg belastning, stigande eller sjunkande längd på teststrängarna och båda protokollen. Tabell 1 visar alla specifikt utvalda ordinarie tester, som redovisas i avhandlingen.

6.5.1 Låg belastning: TCP

I figur 21 visas test 1 (se tabell 1) med stigande längd på nyttodata i teststrängarna fr.o.m. 1 B t.o.m. 1458 B vid ett TCP-test med låg belastning. Figuren visar genom-

strömningen över hela skalan av längder på teststrängar sända till och mottagna från plattformen. I figuren visas genomströmningens maximivärden med blå streckad kurva, medelvärden med röd heldragen kurva och minimivärden med grön streckad kurva utav tre repetitioner.

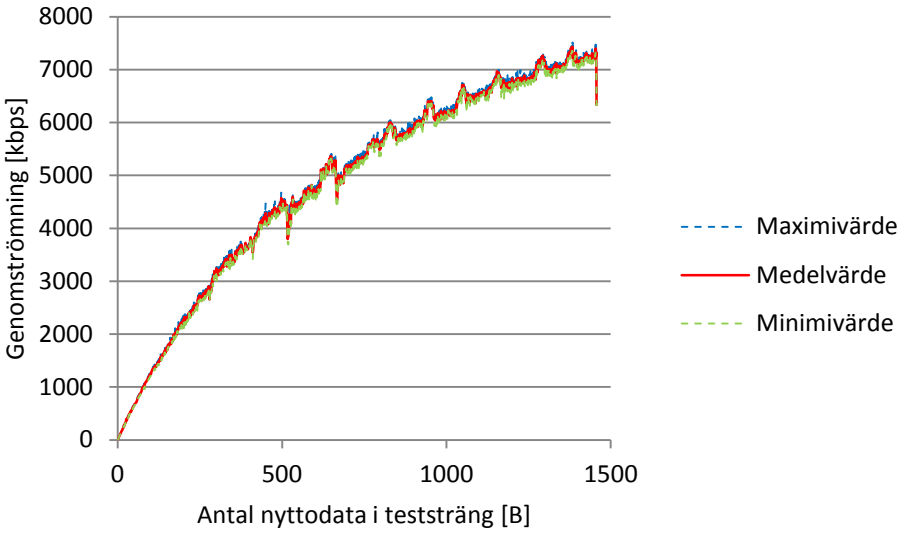


Fig. 21. Resultat från test 1 av genomströmning över hela området för nyttodatalängderna (1–1458) B och N=3.

Mellan maximi- och minimivärdelinjen finns ständigt en skillnad vilket betyder att responstiderna under testet varierar vid samma längd på testdata. Maximi- och minimivärdena vid varje längd på teststrängen innesluter responstiderna för respektive grupp av tre lika långa sända och mottagna TCP-testmeddelanden.

I och med att ett TCP-test med låg belastning fullföljs så sänds och tas emot tre stycken teststrängar per ny teststrängslängd. Detta leder till att DMA-avbrotten har ett lågt antal då de är tre stycken för både in- och utförsel av data till och från RAM respektive innan värddatorn tar sig tid att skapa en ny teststrängslängd. Ethernet-task, NätverksTask och NätverksTidsTask belastar processorn lågt med tre gångers processhantering av både mottagna och sända meddelanden. Taskarna sköter tolkning av kommandon från Ethernet-MMI, styrning av bildbehandling, hantering och omvandling av inkommande och utgående TCP-meddelanden samt tidshantering och övervakning av meddelanden för aktuella protokoll. I och med att DMA-avbrotten inträffar sällan och taskarna belastar

processorn lågt kommer heller inga stora antal trapp- eller trappnivåväxlingar eller stora antal eller djupa nedåtgående pikar att framträda på genomströmningslinjen. Den ses ha en stabilare gång.

6.5.2 Olinjäritet i genomströmningskurvan

I systemet under testerna finns ett timer-avbrott som aktiveras och påverkar genomströmningen. Tiden för exekvering av avbrottet är dock kort och om det syns så är det som tvära nedåtgående pikar på genomströmningslinjen. Taskarna omnämnda i kapitel 5.6.1 exekveras ständigt ensamma eller i olika grupper och syns som trappor med olika nivåer på genomströmningslinjen. Taskarna väcks baserade på prioriteter, tider eller händelser i t.ex. socket under testerna.

Det finns många faktorer som sänker genomströmning och resulterar i att denna planar ut. Varje protokollager som används kostar en del i tid att behandla. Dessa protokoll fordras för att visa väg för och skydda data. Ytterligare faktorer kan t.ex. vara att mottagande nätverksenhet är upptagen eller händelser i eller på applikationsnivå. Även busskollisioner resulterar i förlängning av svarstiderna om omsändning måste fullföljas. Möjligheter till busskollisioner ökar ju längre data man sänder och resulterar i förlängning av svarstiderna. Det kan även uppstå stridigheter om vem som skall använda bussen om den är ledig för sändning och systemet har kontroll. Det finns även fördröjningar på 5 ns/m i kabeln och dessa resulterar i att ett *kort* meddelandes sista bit anländer *tidigare* till mottagande nod än ett *långt* meddelandes sista bit.

Ju längre data som sänds desto längre tid går åt för sändning och mottagning i både värddator och plattform. Ju längre teststrängar som sänds och tas emot ju mera belastas processorn p.g.a. signalering med OSQPost-funktionen där meddelanden överförs via en mellanbuffert och anländer till mottagande (Ethernet-task) eller sändande task (NätverksTask) via en *kö* (se figur 16) via operativsystemet. Vid sändning av längre och längre teststrängar ökar tiden för beräkning av CRC-summan både före sändning och efter mottagning av data i både värddator och plattform. I plattformen används HAL API-funktioner som har overhead i stacken då dessa anropas. Beroende på hur medsänd data refereras till kan anropen förlänga tiden att processa denna.

Ju längre teststrängarna är desto längre tid tar (från yttersta till innersta nivå i systemet) dataöverföring av indata och dataöverföring av utdata på Ethernet-bussen, DMA-överföring från Ethernet-krets till RAM och DMA-överföring från RAM till Ethernet-krets, TCP-stackhantering av indata och TCP- eller UDP-stackhantering av utdata med NicheStack-mjukvarukomponenten, inhantering av data med socket vid mottagning samt uthantering av data med socket vid sändning och signalering till NicheStack-mjukvarukomponenten och signalering till tolkande task och tolkning av meddelande.

Gemensamma faktorer som ständigt ligger i bakgrunden och primärt påverkar ovannämnda delar är processorns hastighet, operativsystemet och dess meddelandehantering (via kö), DMA-kanalernas överföringshastighet och ev. undantag i timer-servicen och -avbrottet. Sekundärt påverkar även dessa faktorer genomströmningshastigheten hos systemet, förorsakar köbildning i systemet och resulterar i allt längre och längre svarstider vid längre teststrängar och en olinjär genomströmningskurva i relation till längden på testdata.

6.5.3 Låg belastning: TCP (övriga utvalda tester)

I figur 22 visas test 2 (se tabell 1) med sjunkande längd på nyttodata i teststrängarna vid ett TCP-test med låg belastning. I figuren visas att genomströmningen är ganska linjär ända till teststrängens minimivärde på 1 B.

Mellan maximi- och minimivärdelinjerna finns ständigt en skillnad vilket betyder att responstiderna under testet varierar vid samma längd på testdata. Maximi- och minimivärdelinjerna möter inte varandra men är närmast varandra i slutet av testet. Maximi- och minimivärdena vid varje längd på teststrängen innesluter responstiderna för respektive grupp av tre lika långa sända och mottagna TCP-testmeddelanden.

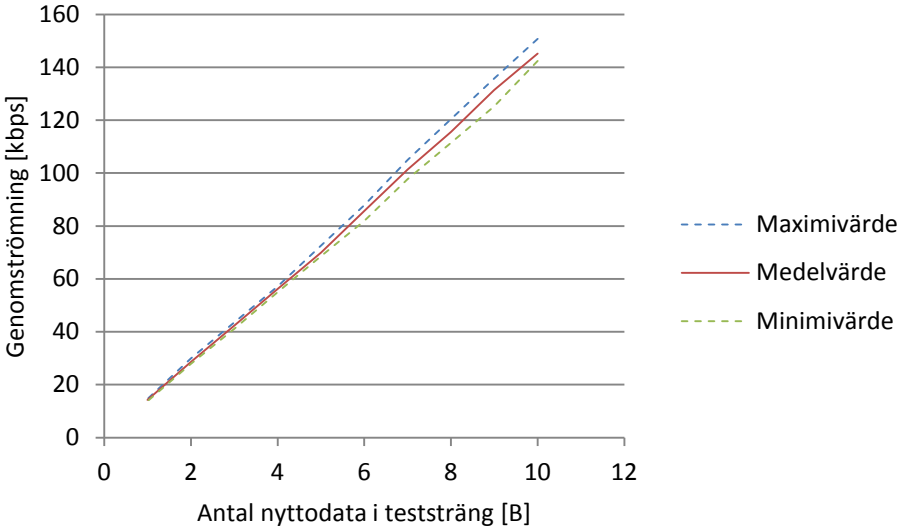


Fig. 22. Resultat från test 2 av genomströmning i början av området för nyttodata-längderna (10–1) B och N=3.

I figur 23 visas test 3 (se tabell 1**)** med en stigande längd på nyttodata i teststrängarna vid ett TCP-test med låg belastning.

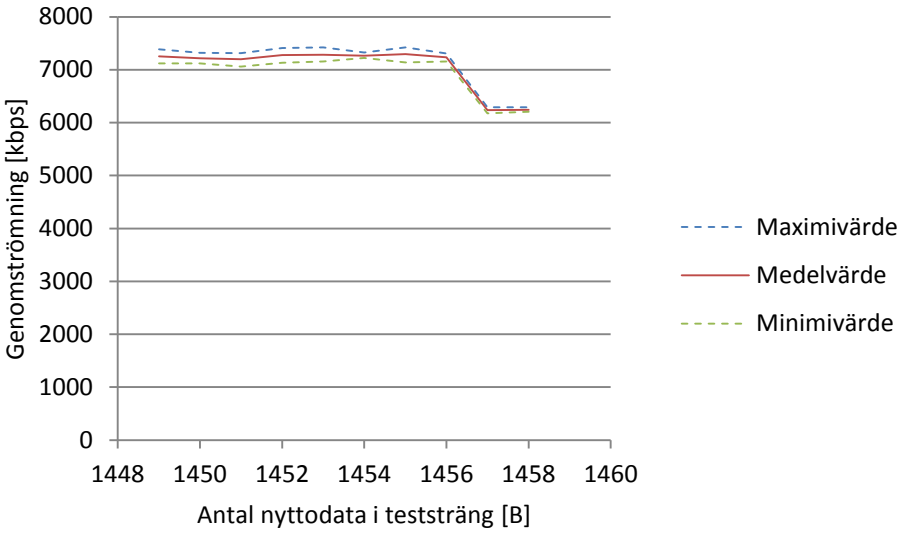


Fig. 23. Resultat från test 3 av genomströmning i slutet av området för nyttodatalängderna (1449–1458) B och N=3.

I figur 23 visas att genomströmningen fortsätter ända till teststrängens maximivärde på 1458 B[1]. För mera information om funktioner hänvisas läsaren till kapitel 5.6.5.

6.5.4 Hög belastning: TCP

I figur 24 visas test 4 (se tabell 1) med sjunkande längd på nyttodata i teststrängarna vid ett TCP-test med hög belastning. Figuren visar genomströmningen över hela skalan av längder på teststrängar sända till och mottagna från plattformen. I figuren visas genomströmningens maximivärden med blå streckad kurva, medelvärden med röd heldragen kurva och minimivärden med grön streckad kurva.

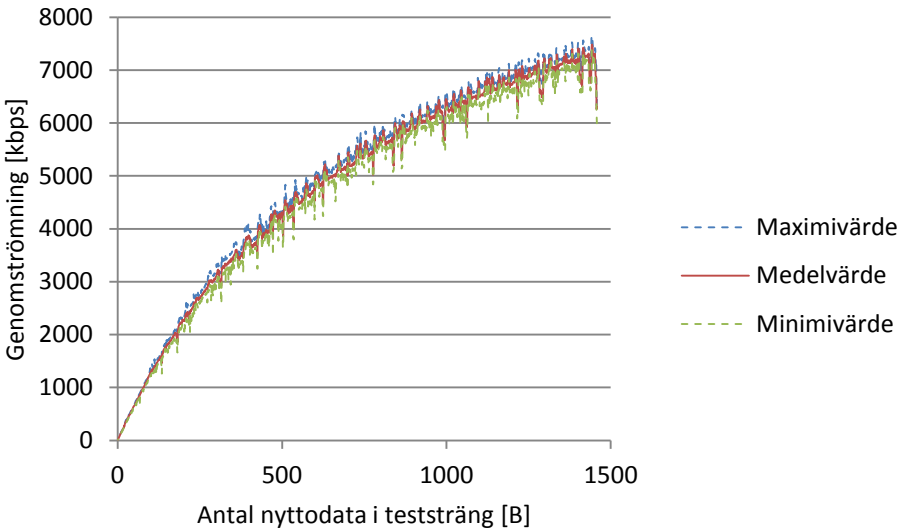


Fig. 24. Resultat från test 4 av genomströmning över hela området för nyttodatalängderna (1458–1) B och N=15.

Mellan maximi- och minimivärdelinjen börjar nu framträda en större skillnad vilket betyder att responstiderna under testet varierar mera vid samma längd på testdata. I och med att ett TCP-test med hög belastning fullföljs så leder detta till att DMA-avbrotten har ett högt antal för både införsel och utförsel av data till och från RAM respektive innan värddatorn tar sig tid att skapa en ny teststrängslängd.

[1] När skillnaden mellan maximal MSS i IPS och värddatorn är två eller mera börjar NicheStack-mjukvaran i IPS sända ett extra TCP-meddelande före responsmeddelandet. Därvid faller genomströmningen med ca 1000 kbps.

Taskarna omnämnda i kapitlen 5.6.1 och 5.6.4 belastar nu processorn högt av både inkommande och utgående meddelanden. Se vidare i samma kapitel om taskarnas uppgifter och deras koppling till kommunikation och tolkning av inkommande meddelande. I och med att DMA-avbrotten nu har ett högt antal och inträffar ofta och taskarna belastar processorn högt kommer ett högt antal trapp- och trappnivåväxlingar och högt antal och djupa nedåtgående pikar att framträda mera på genomströmningslinjen. Genomströmningslinjen har fått en klart ostabilare gång.

6.5.5 Låg belastning: UDP

I figur 25 visas test 5 (se tabell 1) med sjunkande längd på nyttodata i teststrängarna vid ett UDP-test med låg belastning. Figuren visar samma typ av olinjäritet som vid ett TCP-test med låg belastning (figur 21).

Vid jämförelse mellan TCP- och UDP-testet med låg belastning (figur 21 respektive figur 25) kan skönjas en något större skillnad mellan minimi- och maximilinjen.

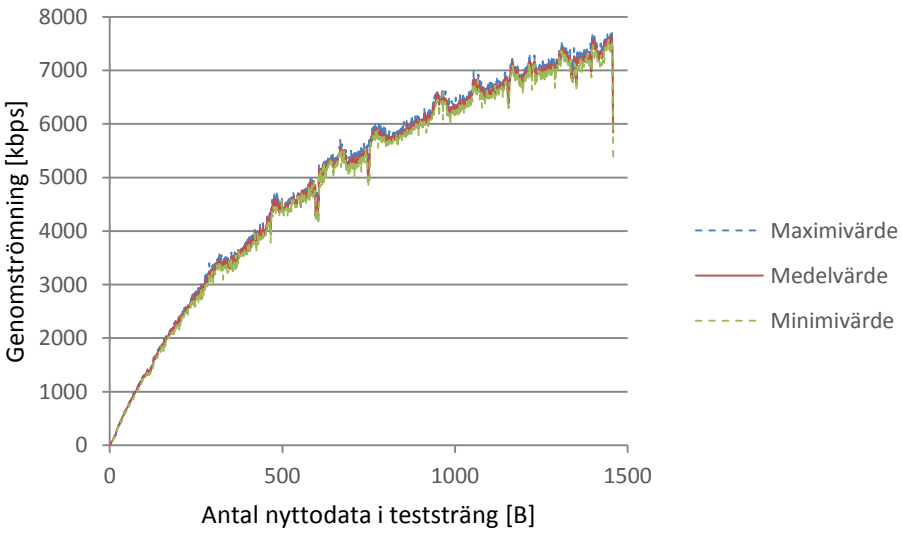


Fig. 25. Resultat från test 5 av genomströmning över hela området för nyttodatalängderna (1458–1) B och N=3.

Gällande processhanteringen måste den nu hantera flera protokoll i.o.m. att TCP- och UDP-meddelanden sänds. Även en något större känslighet noteras för avbrott vid UDP-testet vilket betyder en större variation på responstiderna. Det finns inga större skillnader gällande antal sända och mottagna teststrängar per ny teststrängslängd och antal låga DMA-avbrott för både in- och utförsel av data till och från RAM respektive.

Det finns en generell funktionell skillnad vid (låg/hög belastning) UDP-tester gentemot (låg/hög belastning) TCP-tester i detta system, som påverkar den funktionella helheten och givetvis genomströmningslinjen. Vid (låg/hög belastning) UDP-tester används två protokoll vilket leder till att kommunikations- och tolkningstasken belastas hårdare. I och med UDP-testerna skapas en UDP-socket i IPS. NicheStack-mjukvarukomponentens task (NätverksTask) måste nu dels lyssna på och vara redo att sända meddelanden på plattformens TCP:s IP-adress och dess port (30) och dels lyssna på och vara redo att sända meddelanden på plattformens UDP:s IP-adress på dess port (10.000) för att kunna hantera båda typerna av protokollstackar.

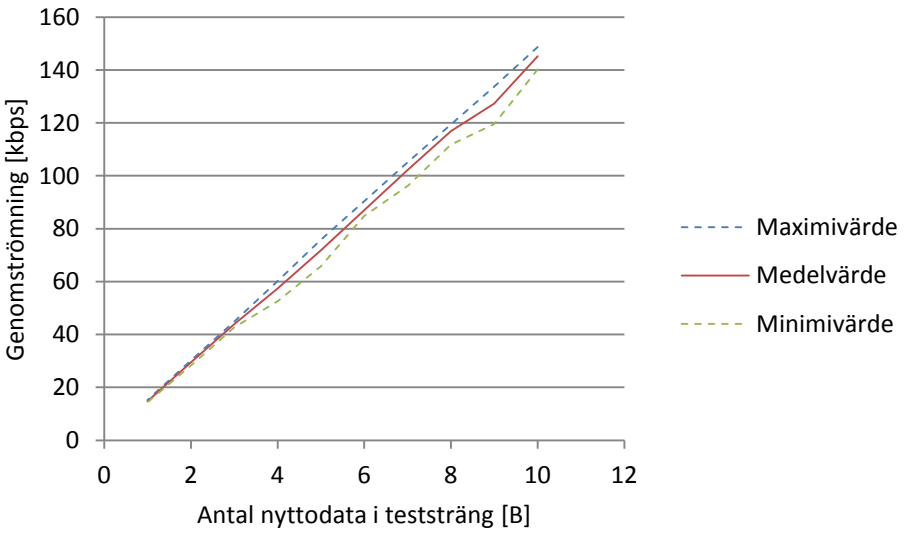


Fig. 26. Resultat från test 6 av genomströmning i början av området för nyttodatalängderna (1–10) B och N=3.

I figur 26 visas test 6 (se tabell 1**)** med en stigande längd på nyttodata i teststrängarna vid ett UDP-test med låg belastning. Vid jämförelse mellan TCP- och UDP-testet med låg belastning (figur 22 respektive figur 26) finns inga större skillnader gällande genom-

strömningslinjerna, som är ganska linjära ända till respektive teststrängs minimivärde på 1 B om än processhanteringen nu måste hantera flera protokoll i.o.m. att TCP- och UDP-meddelanden sänds. Inga större skillnader kan noteras på minimi- och maximilinjen, känslighet för avbrott, antal sända och mottagna teststrängar per ny teststrängslängd eller antal låga DMA-avbrottspikar för både in- och utförsel av data till och från RAM eller tasksvängningar.

I figur 27 visas test 7 (se tabell 1**)** med en stigande längd på nyttodata i teststrängarna vid ett UDP-test med låg belastning. Vid jämförelse mellan TCP- och UDP-testet med låg belastning (figur 23 respektive figur 27) noteras ingen generell skillnad vid UDP-testet om än processhanteringen nu måste hantera flera protokoll i.o.m. att TCP-meddelanden tas emot (kommando och nyttodata sänds från värddator) och sänds (kvitteras från IPS) och UDP-meddelanden sänds (nyttodata sänds omvandlat från IPS).

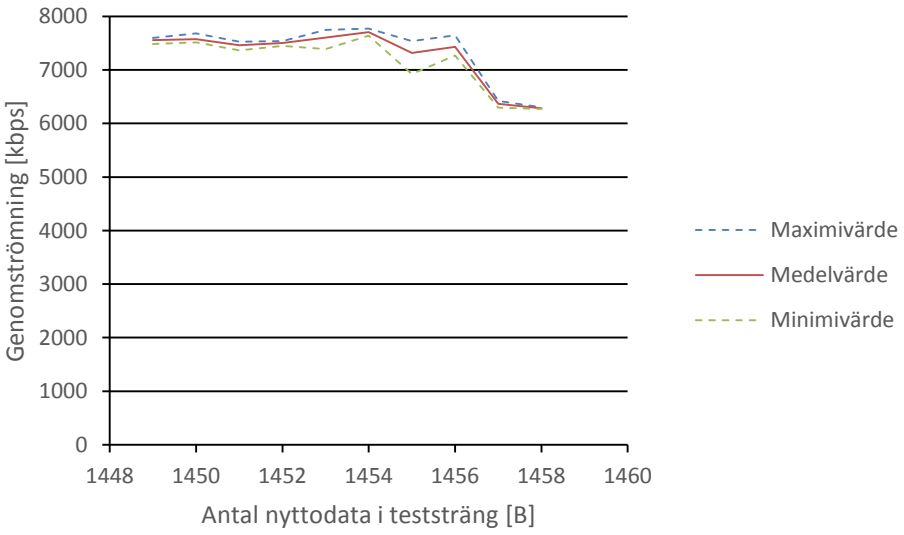


Fig. 27. Resultat från test 7 av genomströmning i slutet av området för nyttodatalängderna (1449–1458) B och N=3.

Gällande genomströmningslinjerna har testerna nära likadan gång och genomströmningslinjer. Båda testerna fungerar ända till teststrängens maximilängd på 1458 B[1], har lika låga antal DMA-avbrott för både in- och utförsel av data till och från RAM. Dock kan noteras en större känslighet för avbrott och taskbyten och -svängningar vid UDP-testet.

6.5.6 Hög belastning: UDP

I figur 28 visas test 8 (se tabell 1**)** med stigande längd på nyttodata i teststrängarna vid ett UDP-test med hög belastning. Figuren visar genomströmningen över hela skalan av längder på teststrängar sända till och mottagna från plattformen.

Läsaren hänvisas till kapitel 6.5.5 angående användning av två protokoll. När TCP- och UDP-kommunikation måste hanteras samtidigt belastas kommunikations- och tolkningstasken högt. Vid jämförelse mellan TCP- och UDP-testet med hög belastning (figur 24 respektive figur 28) noteras att processorn belastas högt av taskarna och avbrotten. Detta leder till ett stort antal och djupt nedåtgående pikar och djupt gående trapp- och trappnivåväxlingar på genomströmningslinjen vilka framträder mycket klart och tydligt på genomströmningslinjen, som därför ses ha en mycket ostabil gång.

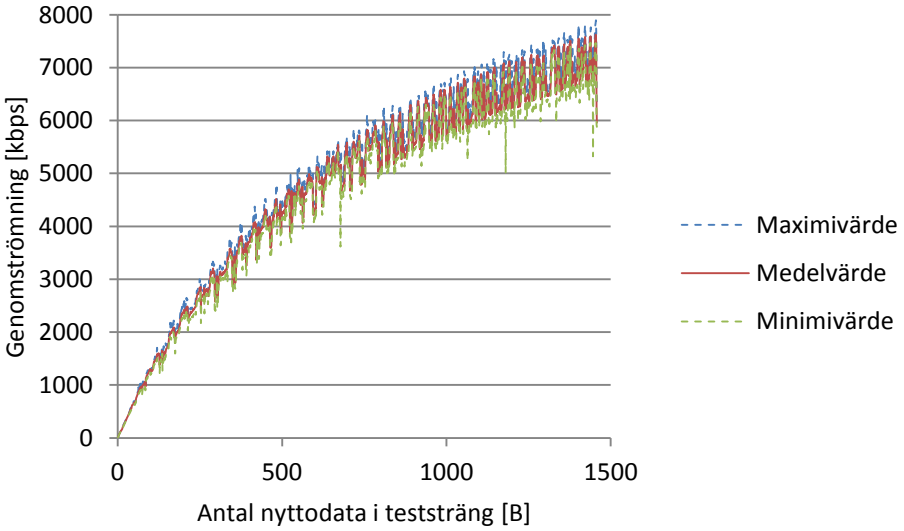


Fig. 28. Resultat från test 8 av genomströmning över hela området för nyttodatalängderna (1–1458) B vid test 8 (N=15**)**.

Vid jämförelsen angående olinjäritet i genomströmningslinjerna finns likheter. Minimi- och maximilinjen har dock betydligt större skillnad och är känslig för avbrott och taskbyten vilket förorsakar en större variation på responstiderna i UDP-testet. Gällande relationen mellan antalet DMA-avbrott för både in- och utförsel av data till och från RAM är de lika för båda testerna. Genomströmningen påverkas av systemmjukvarans och -

hårdvarans konstruktion i samverkan mellan taskar, deras parametrar och olika avbrott, deras samverkan och prioriteter.

6.6 Bildprestanda

Enligt figur 25 är IPS-plattformens MGSH ca 7,5 Mbps. Tillgängliga pixel/s är:

$$p_s = \frac{G_{str}}{p_b},$$

där G_{str} är genomströmning $(\mathrm{b/s})$, p_b är använda antalet bitar, som en pixel representerar (b) och p_s är antal tillgängliga pixel $(\mathrm{pixel/s})$.

Med en bredd på 8 b/pixel blir 937.000 pixel/s tillgängliga. Om man skapar en liksidig kvadrat av tillgängliga pixel/s så är den liksidiga bildkvadratens pixelsidlängd:

$$a = \sqrt{\frac{p_s}{n}},$$

där p_s är tillgänliga pixel $(\mathrm{pixel/s})$ (se ovan), *n* är önskade antal bilder $(\mathrm{bilder/s})$ och *a* är bildkvadratens sidbredd $(\mathrm{antal\ pixel/sida})$ i monitorn.

Resultatet med en hastighet på 937.000 pixel/s tillgängliga genererar en kvadratisk bild/s med storleken (970×970) pixel. Önskas åtta kvadratiska bilder/s med samma antal pixel/s tillgängliga så genereras åtta kvadratiska bilder/s med storleken (342×342) pixel. Minskas antalet b/pixel till 4 b så ökar genomströmningen till 1.875 kpixel/s och nu genereras 16 kvadratiska bilder/s med storleken (342×342) pixel.

En TV-monitor (äldre) innehåller (640×480) pixel, alltså 307.200 pixel, som bildar hela skärmytan. Om 937.000 pixel/s är tillgängliga från IPS via Ethernet-bussen kan man visualisera $\frac{937.000}{307.200}$, alltså ca 3,05 fulla skärmytor/s (bilder/s) med storleken (640×480) pixel i monitorn. Minskas antalet b/pixel till 4 b så ökar genomströmning-

en i IPS via Ethernet-bussen till 1.875.000 pixel/s. Nu kan visualiseras $\frac{1.875.000}{307.200}$, alltså ca 6,1 fulla skärmytor/s (bilder/s) med storleken (640×480) pixel i monitorn. Önskas endast en bildstorlek på (342×342) pixel i monitorn (ca 53 % och ca 70 % av skärmbredd respektive -höjd) kan ca 16 kvadratiska skärmytor/s (bilder/s) visualiseras.

6.7 Resurseffektivitet och optimering

I tabell 2 visas ett utdrag från senaste Quartus II-hårdvarukompilering (något förkortat).

Tabell 2. Utdrag från ett kopplingssammandrag av senaste Quartus II-kompilering av plattformens HW. Kompilatorn utnyttjade 9 % av kombinatoriska-ALUT-, 1 % av minnes-ALUT-, 1 % av DSP- och 38 % av PLL-elementtyper i FPGA.

Enhet(er)	Typ eller använda/möjliga antal använda enheter	Använda enheter
FPGA-familj	Stratix III	
Enhet	EP3SL150F1152C2	
Kombinatoriska-ALUT	10.366/113.600	9 %
Minnes-ALUT	798/56.800	1 %
Logiska register[2]	13.543/113.600	12 %
Register totalt	13.971	
FPGA-kretsstift	341/744	46 %
Totala minnesblocksbitar	2.427.976/5.630.976	43 %
DSP-block (18-bitars element)	4/384	1 %
PLL-block	3/8	38 %
DLL-block	1/4	25 %

[2] dedikerade register

För att höja prestandan på plattformen görs olika optimeringar av mjukvaran (se tabell 3 för optimeringskomponenter, -variabler och -alternativ). Vissa grundtaskar och utskrifter i JTAG-MMI suspenderas (blockeras) under prestandatesterna.

Tabell 3. Optimeringskomponenter, -variabler och -alternativ.

Optimeringskomponent eller tasknamn	Optimerings-variabel	Optimeringsalternativ
SystemInitTaskens prioritet	X_1	1/5
NätverksTaskens prioritet	X_2	2
NätverksTidsTaskens prioritet	X_3	3
Ethernet-task	X_4	4
Mjukvaruoptimering	X_5	O0–O3
Instruktionscacheminne	X_6	0–64 kB
Datacacheminne	X_7	0–64 kB
Processortyp	X_8	Nios II/e / Nios II/f
Nyttodatasträngens storlek (i TCP)	X_9	1–1458
LED-uppdateringstask	X_{10}	EBl/Bl
7-segmentteckenrutans uppdateringstask	X_{11}	EBl/Bl
JTAG-task	X_{12}	EBl/Bl
Utskrifter via JTAG-MMI	X_{13}	EBl/Bl

Beroende på vilka delar optimeringen fokuserar på och till vilken grad optimeringen sker påverkas exekveringshastigheten av mjukvaran. Vissa delar av mjukvaran blir snabbare och vissa delar kvarstår med samma exekveringstid.

Fördelar med optimering är att mjukvarans exekveringshastighet ökar p.g.a. att man t.ex. använder och behöver hantera ett mindre antal register vid beräkning eller hantering av data. Dock kan man få betala ett pris för detta genom att kompilatorn fordrar längre tid vid kompileringen av källkoden för att lösa problemet på ett annat sätt än det sedvanliga.

En nackdel med optimering är om en kompilator antar en viss bredd på en variabel, t.ex. integer-värde (32-bitars bredd). Vid optimering är sättet på hur en variabel deklareras viktig. I ett dylikt fall uppstår en osäker zon vid läsningen av en 64-bitars enhet och skrivning till variabeln. När 64-bitars enheten läses och data skrivs till variabeln kommer endast enhetens lägre 32 bitar att skrivas till variabelns lägre 32 bitar. Den övre halvan av 64-bitars enheten nås inte fullt ut. Läsaren hänvisas till (Chellappa, Franchetti, & Püschel, 2008) för fördjupning i olika optimeringstekniker.

Tabell 4. Genomströmningen med olika optimeringar. Se tabell 3 för betydelse av optimeringsvariabel.

X_1	X_2	X_3	X_4	X_5	X_6 [B]	X_7 [B]	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	MGSH [kbps]
5	2	3	4	O0	512	512	Nios II/f	1456	Bl	Bl	Bl	Bl	3857
1	2	3	4	O0	32 k	32 k	Nios II/f	1456	Bl	Bl	Bl	Bl	4973
1	2	3	4	O0	64 k	32 k	Nios II/f	1456	Bl	Bl	Bl	Bl	5445
1	2	3	4	O0	64 k	64 k	Nios II/f	1456	Bl	Bl	Bl	Bl	5493
1	2	3	4	O3	64 k	64 k	Nios II/f	1456	Bl	Bl	Bl	Bl	7592

Optimeringar i tabell 4 visar hur MGSH ökar vid olika optimeringar. Genom optimeringar ökas genomströmningen med ca 97 % i plattformen.

De mest betydande optimeringsvariablerna är X_5, X_6 och X_7. I figur 29 visas genomströmningslinjer *med* och *utan* optimeringar (övre respektive nedre genomströmningslinje) vid en TCP-test med låg belastning.

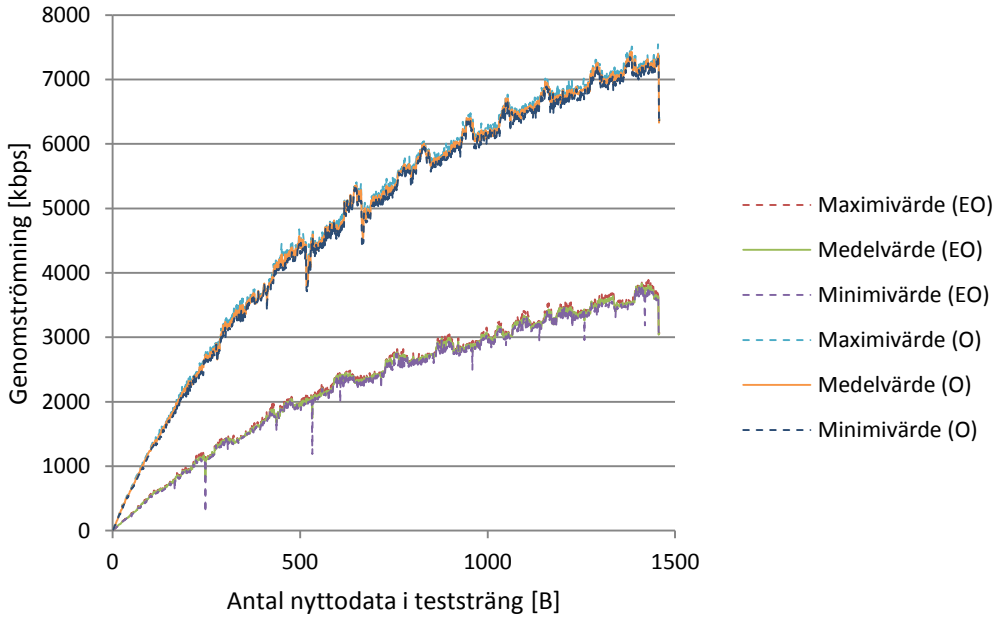


Fig. 29. Resultat från test *med* optimeringar (O) (övre genomströmningslinje enligt optimeringsalternativ i nedersta rad i tabell 4) och *utan* optimeringar (EO) (nedre genomströmningslinje enligt optimeringsalternativ i översta rad i tabell 4) av genomströmning över nyttodatalängderna (1–1458) B och N=3.

7 SAMMANDRAG

I denna teknologie licentiatavhandling i automationsteknik planeras, förverkligas och testas en FPGA-plattform för bildbehandling i SoPC-omgivning där antalet bildbehandlingsmetoder endast begränsas av antal HW-celler i FPGA. Plattformen med serveregenskaper har en 1 Gbps Ethernet-buss och kan tillämpas för behandling av bilder och video i realtid.

Forskningsproblemet är att bygga upp en FPGA-plattform för bildbehandling med snabb busskommunikation. Plattformen skall arbeta emot Internet med protokollen TCP och UDP, ha god konfigurerbarhet, bra designverktyg och god testbarhet. Den skall ge möjlighet till många metoder och typer av bildbehandling, vara mångsidig och förutseende samt utvecklingsvänlig och förmånlig med tanke på dynamik.

I och med en FPGA-lösning valdes ansågs Terasics ALTERA DE3-utvecklingskort med alla dess fördelar vid utveckling och gränssnitt emot dotterkortet HSMC-NET innehållande två stycken 1 Gbps Ethernet-gränssnitt vara ett bra val. För att skapa en snabb bildbehandling gjordes ett HW-acceleratorgränssnitt för instansieringar eller implementeringar av många bildbehandlingsmetoder, en HW-komponent för val av metoder kopplad till bildbehandlingsenheten och fyra exempelmetoder med HW-komponenter, alla med en pixels resolution. För snabb och självständig ut- och införsel av data till och från en Ethernet-krets (av två möjliga) på dotterkortet skapades två självständiga DMA IP-komponenter och RAM-buffertar kopplade till varandra. I och med många taskar inom plattformen som exekveras parallellt implementerades en Nios II/f-processsor IP-komponent som exekverar operativsystem, annan mjukvara och fullföljer access till många enheter. För styrning av tidsfördelning av taskarna inom operativsystemet implementerades en Timer IP-komponent som även sköter om andra synkrona uppgifter inom systemet. För hantering av TCP- och UDP-trafiken finns en NicheStack-mjukvarukomponent inkluderad. För framtida behov av stora datamängder sattes även ett gränssnitt in emot ett höghastighetsminne med en kapacitet på 1 GB.

Genom valet av ALTERA DE3-kortet kopplas automatiskt Alteras Qsys-utvecklingsverktyg in, som drivs med ny optimerad FPGA-hårdvaruteknik samt utvecklingsverktygen Quartus II och Eclipse. I och med detta blir plattformen kompatibel med Alteras övriga system, tillgängliga och kommande IP-komponenter och HW- eller SW-lösningar för samanvändning via eller genom utvecklingsverktygens användargränssnitt och -referenser. Nu är även sammansättningen kopplad till mjukvaran, som befinner sig inom SW-sfär med HAL-biblioteket och tillgängliga och kommande API-abstraktioner. Mjukvaran är i övrigt sammansatt av applikations- och avbrottsrutiner, operativsystem, taskar och drivrutiner där HW/SW Co-Design valdes som konstruktions- och implementeringsmetod för att undvika omfattande efterkonstruktioner, konstruera hårdvara och mjukvara samtidigt, hårdvaru- och mjukvarumålmedvetet och utnyttja synergin mellan dessa. Genom Co-Design bibehålls helhetssynen över systemet och beslut för nästa steg underlättas. Hårdvara och mjukvara är vägande faktorer vid denna metod och förutsätter att användaren har god kännedom om deras funktion.

För att testa plattformen skapades ett mångsidigt IPS testsystem-program i C#, som körs i Windows 7, med vilket man kan kommunicera med plattformen via en 1 Gbps Ethernet-buss och spara mätdata i Excel-filer. Manuella tester utfördes med IPS testsystem-programmet och TCP- och UDP-trafik med alla styr- och bildbehandlingskommandon och det konstateras att alla kommandon fungerar korrekt i alla lager och på alla nivåer utan omförsök eller -sändningar.

Prestandatester utfördes även med IPS testsystem-programmet och TCP- och UDP-trafik, stigande och sjunkande längd på testdata och låg och hög belastning. Alla tester visade samma struktur och trend för genomströmning. Maximigenomströmningen för plattformen är ca 7,5 Mbps. Med en upplösning på 8 b/pixel kan en (1) kvadratisk bild/s genereras med storleken (970×970) pixel. Optimeringar hade en stor betydelse vid höjning av genomströmningen. Genomströmningen måste ses i perspektivet av att processorn har en arbetsfrekvens på endast 50 MHz. I sjunde paragrafen från slutet diskuteras höjning av processorkraft.

Det primära syftet med denna *avhandling* var inte att uppnå högsta möjliga genomströmning i plattformen utan att samordna alla funktioner, mjukvara och hårdvara, IP- och HW-komponenter, -gränssnitt och -enheter till en helfungerande sammansättning och plattform, som löser de problem och kriterier, som finns omnämnda i forskningsproblemet. Framtida projekt för IPS kan kategoriseras i flera grupper som förändringar, förbättringar, kompletteringar men även insättningar av nya funktioner rörande nya koncept.

Bildbehandlingsplattformen kan även anpassas för bruk i kontrollautomationssystem. I plattformen skapas då gränssnitt emot A/D- och D/A-omvandlare och I/O-enheter, vilket dagens kontrollautomationssystem förutsätter. Gränssnitten kopplas till en omkringliggande prototyphårdvara uppbyggd av konventionella transformatorer och sensorer respektive konventionella reläer för bruk i kontrollnät(verk).

Tanken på att kunna komma undan VHDL- eller Verilog-programmering i samband med hårdvaruprogrammering är intressant och kunde appliceras på IPS. Genom Kiwisystem som modellerar digitala kretssystem med program kan man m.h.a. C#-programmering skapa Verilog-källkodsfiler. Dessa körs emot kompilator, som sedvanliga Verilog-filer. För mera information se (Greaves & Singh, 2010).

Även en höjning av (mjukvarans) exekveringshastighet och genomströmning i IPS är en intressant framtida utmaning. Detta kan t.ex. göras m.h.a. ökning av processorns exekveringshastighet via PLL och SoC-, multiprocessor-, bildkomprimerings-, parallellprocessor-, C2H- eller VectorBlox MXP Matrix Processor-lösning (se (Severance & Lemieux, 2013) och (Lemieux, 2012)). I den sistnämnda referensen lovas t.o.m. en ökning på 100−1.000 gånger prestandan i ett sedvanligt Nios II/f-system och all programmering sker i C utan VHDL- eller Verilog-programmering.

Fullgörande av mjukvaran har medfört betydande fördjupningar i denna. Gällande operativsystemet har användning av nya funktioner varit i fokus. Gällande taskarna har det medfört förändringar och justeringar av existerande taskar och prioriteter men även skapande av ny task för JTAG-MMI. Angående applikationerna har fördjupningar skett

genom insättningar och kompletteringar av nya applikationer inkluderandes MMI-kommandon för Ethernet- och JTAG-bussen och speciellt bildbehandlings- och styrkommandon. En uppbyggnad av ny och fördjupande i mellanmjukvara med flermetodsegenskaper har fullföljts med justeringar och kompletteringar av mjukvara med flermetodsegenskaper och ständig inskrivning av ny data till bildbehandlingskärnan. Med nya faciliteter har access till bildbehandlingsacceleratorer erhållits och enheterna för bildbehandling och val av behandlingsmetod. Drivrutinerna har berörts genom skapande av nya och komplettering av existerande drivrutiner för olika kommunikationsprotokoll och kommunikation med DMA 0-, DMA 1- och Timer-enheten. För uppföljande av skeenden i realtid har även berörts meddelandesignalering till JTAG-MMI i IPS i avbrottsrutinerna. Fördjupning i hantering av kommunikationen har även skett genom egenutvecklade protokoll i IPS och Ethernet- och JTAG-MMI i värddatorn. Fullgörande av hårdvaran har medfört fördjupningar i Timer-, DMA 0- och DMA 1-enheten genom tillägg av utsignaler från enheterna kopplade till ett GPIO-gränssnitt. Genom detta arrangemang har enheternas tillstånd kunnat följas upp i realtid m.h.a. mätinstrument. Under testerna kan man även styra plattformens olika beteenden m.h.a. PIO-enheter och styrsignaler kopplade till dessa enheter. GPIO-enheten ger m.h.a. in- och utgångssignaler och skapade tilläggssignaler tillgång till innersta delen av hårdvaran. Via denna kan fritt valda hårdvaruenheter eller mätpunkter i FPGA monitoreras i realtid med yttre mätinstrument. Genom initiering av MAC- och GMII-enheten kan kommunikation till och från Ethernet-gränssnitten och hastigheten på Ethernet-bussen justeras.

Genom att skapa en bildbehandlingsenhet med *ett* gemensamt hårdvarugränssnitt för *många* bildbehandlingsacceleratorer till vilket man kan ansluta nya metoder så undviks omkonfigurering av hela gränssnittet. I plattformen finns några exempel på bildbehandlingsmetoder representerade med denna tillämpning. Via skapad enhet för val av metod och dess utgångar kopplade till bildbehandlingsacceleratorn görs metodval. Avslutningsvis har en fördjupning i Co-design skett, som är den sammanlänkande faktorn mellan hårdvara och mjukvara. Här har arbetet fokuserats kring och fördjupats i att på systemnivå konstruera hårdvaru- och mjukvarumålmedvetet, samtidigt och att utnyttja synergin mellan dessa.

Gemensamma faktorer när man skapar en mall för ett dylikt system är att man måste beakta och definiera men även tydligt avgränsa *vilka* arbeten, som mjukvaran och hårdvaran fullgör och *vilken* del av dessa, som har den *styrandes* och *beräknandes* roll enligt von Neumanns modell med fördelning av styrande- och dataflöde. Man måste även tydligt beakta och definiera arbetsenheternas sätt att *kommunicera* med varandra och vad varje enhets (SW eller HW) *in-* och *utgångssignaler* representerar. Vad det *endast* gäller mjukvaran måste man beakta och definiera dess roll(er) gentemot och med operativsystemet, som sköter arbetsfördelningen mellan taskarnas processer och skötande av data med in- och utsignaler.

Vad det gäller applikationerna måste man beakta och definiera de tillämpningar och funktioner som systemet skall ha och deras sätt att använda annan mjukvara. Även mellanmjukvaran i detta system, bildbehandlingen för inkommande och utgående data, dess kommunikation med många olika protokoll men även enheten för val av dessa måste beaktas och definieras. För skötsel av allmän data till specifikt gränssnitt och omgivning måste även beaktas och definieras drivrutinerna så att dessa har en bred arbetsyta emot olika mjukvara och hårdvara. Man måste även beakta och definiera avbrottrutinerna, som driver operativsystemet, kommunikation och hantering av nödvändiga tidsbundna uppgifter. Sist måste mjukvaran också reserveras för oväntade felsituationer som den måste kunna hantera.

Vad det gäller *endast* hårdvaran, måste man beakta och definiera IP- och HW-kärnornas funktioner emot deras beteende och behov. Vidare måste en bildbehandlingsenhet med *ett* gemensamt hårdvarugränssnitt för *många* bildbehandlingsacceleratorer och en enhet för val av bildbehandlingsmetod beaktas och definieras. Även hårdvaran måste reserveras för oväntade felsituationer som den måste kunna hantera.

Avslutningsvis *förrän* konstruktionsarbetet kan startas måste rollen hos Co-design med samtidig konstruktion av hårdvara och mjukvara, målmedveten konstruktion av dessa och utnyttjande av synergin mellan dessa samt signalkopplingar mellan dessa beaktas och definieras. Vid behov kan slutlig placering av mellanmjukvara, bildbehandlingsenhet, enhet för val av bildbehandlingsmetod och drivrutiner beaktas för att uppnå en

mångsidig systemlösning med största möjliga omfång med tanke på deras nytta och användning. Logistiken är en viktig faktor för att undvika onödiga upprepningar och vinna tid vid systemets skapande. Slutligen är ordningsföljden vid skapandet av funktionsblocken och konstruktionsflödet inkluderat de sätt på hur HW- och SW-blocken skapas, deras funktioner och hur de sammankopplas viktig.

REFERENSFÖRTECKNING

Ailawadi, D. (December 2009). *Fundamentals of Parallel Processing.* 8th Indo-German Winter Academy, IIT Roorkee. Hämtat från http://www.leb.eei.uni-erlangen.de/winterakademie/2009/report/content/course02/pdf/0210.pdf

Altera. (2007a). Avalon Memory-Mapped Interface Specification. Hämtat från http://www.cs.columbia.edu/~sedwards/classes/2009/4840/mnl_avalon_spec.pdf

Altera. (2007b). Using the NicheStack TCP/IP Stack Nios II Edition Tutorial. Hämtat från http://www.126doc.com/p-42346255.html

Altera. (2007c). Using the NicheStack TCP/IP Stack Nios II Edition Tutorial. Hämtat från http://read.pudn.com/downloads164/ebook/746760/Usingthenichestacktcpipctack.pdf

Altera. (2008). Designing with the Nios II processor and SOPC Builder. Hämtat från ftp://ftp.altera.com/outgoing/asia/download/08conf/NIOS_yuanyd.pdf

Altera. (2010a). Nios II Software Developer's Handbook. Hämtat från http://people.ece.cornell.edu/land/courses/ece5760/NiosII_doc/Nios_SW_handbook.pdf

Altera. (2010b). Stratix III Device Handbook. (1). Hämtat från https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/stx3/stratix3_handbook.pdf

Altera. (2011a). Using MicroC/OS-II RTOS with the Nios II Processor. Hämtat från http://www.altera.com/literature/tt/tt_nios2_MicroC_OSII_tutorial.pdf

Altera. (2011b). MicroC/OS-II Real-Time Operating System. Hämtat från http://www.altera.com/literature/hb/nios2/n2sw_nii52008.pdf

Altera. (2011c). Overview of the Hardware Abstraction Layer. Hämtat från http://www.altera.com/literature/hb/nios2/n2sw_nii52003.pdf

Altera. (2011d). Ethernet and the NicheStack TCP/IP Stack - Nios II Edition. Hämtat från https://www.altera.co.jp/ja_JP/pdfs/literature/hb/nios2/n2sw_nii52013.pdf

Altera. (2011e). Using the NicheStack TCP/IP Stack - Nios II Edition Tutorial. Hämtat från https://www.altera.com/en_US/pdfs/literature/tt/tt_nios2_tcpip.pdf

Altera. (2014a). *Avalon Interface Specification.* Hämtat från http://www.altera.com/literature/manual/mnl_avalon_spec.pdf?GSA_pos=1&WT.oss_r=1&WT.oss=avalon interface specification

Altera. (2014b). *Nios II Processor Peripherals and Interfaces.* Hämtat från Nios II Processor Peripherals and Interfaces: https://www.altera.com/products/processors/ni2-peripherals.html

Altera. (2014c). Nios II Processor Reference Handbook. Hämtat från http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf 2014

Altera. (2014d). *Software Development Tools for the Nios II Processor.* Hämtat från http://www.altera.com/devices/processor/nios2/tools/ide/ni2-ide.html 2014

Altera. (2014e). *What's New for IP in v14.0 Release*. Hämtat från http://www.altera.com/products/ip/news/ip-whats-new.html

Altera. (2014f). Hämtat från Quartus II Integrated Synthesis: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/qts/qts_qii51008.pdf

Altera. (2014g). Introduction to the Quartus II Software. Hämtat från https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/archives/intro_to_quartus2.pdf

Altera. (2014h). Quartus Prime Standard Edition Handbook Volume 1: Design and Synthesis. Hämtat från http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf 2015

Altera. (2014i). Quartus II Handbook Volume 1: Design and Synthesis. Hämtat från http://www.altera.com/literature/hb/qts/qts_qii5v1.pdf

Altera. (2014j). *Nios II Processor: The World's Most Versatile Embedded Processor.* Hämtat från http://www.altera.com/devices/processor/nios2/ni2-index.html

Altera. (2014k). Nios II Processor Reference Handbook. Hämtat från https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/nios2/n2cpu_nii5v1.pdf

Altera. (2014l). Alternative Nios II Boot Methods. Hämtat från https://www.altera.com/products/processors/support.html#software-development

Altera. (2014m). Developing Programs Using the Hardware Abstraction Layer. Hämtat från https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/nios2/n2sw_nii52004.pdf

Altera. (2015a). Cyclone V SoCs: Lowest System Cost and Power. Hämtat från https://www.altera.com/products/soc/portfolio/cyclone-v-soc/overview.html

Altera. (2015b). *DE3 Development and Education Board*. Hämtat från http://www.altera.com/education/univ/materials/boards/de3/unv-de3-board.html

Altera. (2015c). Quartus II Development Software. Altera. Hämtat från http://wl.altera.com/literature/lit-qts.jsp 2015

Altera. (2015d). Stratix FPGAs. Hämtat från https://www.altera.com/products/fpga/stratix-series/stratix-iii/overview.html

Altera. (2015e). *DMA Controller*. Hämtat från https://www.altera.com/products/intellectual-property/all-ip/interface-protocols/m-eur-dma-cont.html

Altera. (2015f). Instantiating the Nios II Processor. Hämtat från https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/nios2/n2cpu_nii51004.pdf

Altera. (2016). Qsys - Altera's System Integration Tool. Hämtat från https://www.altera.com/products/design-software/fpga-design/quartus-prime/features/qts-qsys.html

ARM, T. S. (2008). What is the difference between a von Neumann architecture and a Harvard architecture? *ARM Technical Support Knowledge Articles*. Hämtat från ARM Technical Support Knowledge Articles: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka3839.html

Arpaci-Dusseau, R. H., & Arpaci-Dusseau, A. C. (2014). Operating Systems: Three Easy Pieces. Scheduling: Introduction. Hämtat från http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched.pdf

BDTi. (2010). *BDTi/ InsideDSP*. Hämtat från Synfora's PICO High-Level Synthesis Tool Achieves BDTI Certification: http://www.bdti.com/InsideDSP/2010/03/18/Synfora 2013

Benedetti, A., & Perona, P. (1998). Real-time 2-D feature detection on a reconfigurable computer. *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)* (ss. 586-593). Santa Barbara: IEEE Conference Publications. doi:10.1109/CVPR.1998.698579

Bharat, B. A., & Sumit, P. T. (2009). *Computer Architecture and Parallel Processing.* Laxmi Publications Pvt. Ltd (2009). Hämtat från https://www.abebooks.com/Computer-Architecture-Parallel-Processing-Bharat-Bhushan/8649491623/bd

Bowen, O., & Bouganis, C.-S. (2008). Real-time image super resolution using an FPGA. *2008 International Conference on Field Programmable Logic and Applications* (ss. 89-94). Heidelberg: IEEE. doi:10.1109/FPL.2008.4629913

Chellappa, S., Franchetti, F., & Püschel, M. (2008). *How To Write Fast Numerical Code: A Small Introduction.* Electrical and Computer Engineering. Carnegie Mellon University. Hämtat från https://users.ece.cmu.edu/~franzf/papers/gttse07.pdf

Chou, C., Mohanakrishnan, S., & Evans, J. (1993). *FPGA Implementation of Digital Filters.* Department of Electrical & Computer Engineering. University of Kansas. Hämtat från http://www.scarpaz.com/2100-papers/chou98fpga.pdf

Chu, P. (2012). *Embedded SoPC design with NIOS II processor and VERILOG examples.* John Wiley & Sons, Inc., Hoboken, New Jersey. Hämtat från http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118011031.html

Cooke, S. (2003). *The Bip Buffer - The Circular Buffer with a Twist.* Hämtat från http://www.codeproject.com/Articles/3479/The-Bip-Buffer-The-Circular-Buffer-with-a-Twist

Cools, W. (2010). Extending COS-II with FPDS and Reservations. Hämtat från http://www.win.tue.nl/~mholende/publications/Cools2010.pdf

Desmouliers, C., Aslan, S., Oruklu, E., & Saniie, J. (2009). FPGA-based design of a high-performance and modular video processing platform. *2009 IEEE International Conference on Electro/Information Technology* (ss. 393-398). Windsor: IEEE. doi:10.1109/EIT.2009.5189649

Desmouliers, C., Aslan, S., Oruklu, E., Saniie, J., & Martinez, V. F. (2010). HW/SW Co-design Platform for Image and Video Processing Applications on Virtex-5 FPGA Using PICO. *Electro/Information Technology (EIT), 2010 IEEE International Conference.* IEEE. doi:10.1109/EIT.2010.5612173

Desmouliers, C., Oruklu, E., Aslan, S., Saniie, & Vallina, F. M. (2012). Image and Video processing platform for FPGAs using high-level synthesis. *Computers & Digital Techniques, IET*, 6(6), 414-425. Hämtat från http://www.ece.iit.edu/~eoruklu/IIT/Publications_files/IET%20HLS%20Video%20processing.pdf

Diego, C. C., & Cerd, J. (2008). Dynamic scheduling of multiproduct pipelines with multiple delivery due dates. *Computers & Chemical Engineering*, *32*(4–5). Hämtat från http://ac.els-cdn.com/S0098135407000592/1-s2.0-S0098135407000592-main.pdf?_tid=e3c37e2a-ef2f-11e4-82c8-00000aab0f01&acdnat=1430395149_62326408258b379c9c14e0227d210832

Donahoo, & Calvert. (2009). *TCP/IP Sockets in C* (2nd uppl.). Hämtat från http://store.elsevier.com/TCP-IP-Sockets-in-C/Michael-Donahoo/isbn-9780123745408/

Duffy, J. (2011). Cisco, Juniper, HP drive Ethernet switch market in Q4. *Network World*. Hämtat från http://www.networkworld.com/article/2245430/lan-wan/cisco--juniper--hp-drive-ethernet-switch-market-in-q4.html

Embedded Micro. (2013). *What is an FPGA?* Hämtat från https://embeddedmicro.com/tutorials/mojo-fpga-beginners-guide/what-is-an-fpga

Finc, M., Trost, A., Zajc, B., & Žemva, A. (2003). HW/SW Co-design of Real-time Video Applications Using a Custom Configurable Prototyping Platform. *Electrotechnical Review, 70(5),* ss. 247–253. Hämtat från http://ev.fe.uni-lj.si/5-2003/finc.pdf

Gaspar, J. (2013). Chapter 7. Boost.Circular Buffer. Hämtat från http://www.boost.org/doc/libs/1_59_0/doc/html/circular_buffer.html

Godse, D., & Godse, A. (2006). *Computer Organisation and Architecture.* Technical Publication Pune. Hämtat från https://www.abebooks.com/9788189411565/Computer-Organisation-Architecture-A.P.Godse-D.A.Godse-818941156X/plp

Gorgon, M. (1997). Universal reprogrammable architecture for implementation dedicated image processors based on FPGA. *1997 Sixth International Conference on Image Processing and Its Applications. 2*, ss. 556-560. Dublin: IEEE. doi:10.1049/cp:19970955

Gorgon, M. (2012). Parallel performance of the fine-grain pipeline FPGA image processing system. *Opto-Electron. Rev., 20(2)*, ss. 153–158. doi:10.2478/s11772-012-0021-2

Gorgon, M., & Pryzybylo, J. (2001). FPGA based controller for heterogenous image processing system. *Proceedings Euromicro Symposium on Digital Systems Design* (ss. 453-457). Warsaw: IEEE. doi:10.1109/DSD.2001.952366

Gottlieb, A., & Almasi, G. S. (1989). *Highly Parallel Computing.* Redwood City, Calif.: Benjamin-Cummings Publishing Co. Hämtat från http://dl.acm.org/citation.cfm?id=160438

Greaves, D., & Singh, S. (2010). Designing application specific circuits with concurrent C# programs. *Eighth ACM/IEEE International Conference on Formal Methods and Models for Codesign* (ss. 21-30). Grenoble: IEEE. doi:10.1109/MEMCOD.2010.5558627

Ho, H., Klepko, R., Ninh, N., & Wang, D. (2011). A high performance hardware architecture for multi-frame hierarchical motion estimation. *IEEE Transactions on Consumer Electronics. 57(2)*, ss. 794-801. IEEE. doi:10.1109/TCE.2011.5955224

Holenderski, M. (2014). *µC/OS-II.* System Architecture and Networking, Department of Mathematics and Computer Science. Eindhoven University of Technology. Hämtat från http://www.win.tue.nl/~mholende/ooti/ooti_rt_course_ucos.pdf

IETF. (1981). INTERNET STANDARD RFC 791. (J. Postel, Red.) Marina del Rey, California 90291, USA: Information Sciences Institute, University of Southern California. Hämtat från https://tools.ietf.org/html/rfc791#page-11

Junaid, K. A., & Ravindrann, G. (2007). FPGA accelerator for medical image compression system. *3rd Kuala Lumpur International Conference on Biomedical Engineering 2006, 15*, ss. 396-399. Kuala Lumpur. doi:1007/978-3-540-68017-8_100

KAMAMI. (2009). Terasic DE3-150. Hämtat från https://kamami.pl/terasic/177330-terasic-net.html

Kleinrock, L. (1964). Analysis of A time-shared processor. *NAVAL RESEARCH LOGISTICS, 11(1)*, ss. 59–73. doi:10.1002/nav.3800110105

Kremer, G. (2009). Hämtat från http://home.kpn.nl/pj.fondse/ide68k/docs/uCOS-II_Intro.pdf

KTH. (2008). *IS2206 Forskningsmetoder för datorsystemteknik.* Hämtat från http://www.kth.se/student/kurser/kurs/IS2206

Labrosse, J. J. (2002a). *MicroC/OS-II The Real-Time Kernel* (andra uppl.). CMP Books, CMP Media LLC. Hämtat från https://www.amazon.com/MicroC-OS-II-Kernel-CD-ROM/dp/1578201039

Labrosse, J. J. (2002b). MicroC/OS-II: µC/OS-II Reference Manual. (andra uppl.), 405-532. Hämtat från http://www.win.tue.nl/~mholende/automotive/uCOS-II-RefMan.pdf

Labrosse, J. J. (2014). *Micrium Embedded Software.* (J. J. Labrosse, Red.) Hämtat från Micrium Community Blog: http://micrium.com/hardware-accelerated-rtos-%C2%B5cos-iii-hw-rtos-and-the-r-in32m3/

Labrosse, J. J. (2016). *Micrium Embedded Software.* (J. J. Labrosse, Red.) Hämtat från Older books: http://micrium.com/books/older-books/

Lacassagne, L., Milgram, M., & Garda, P. (1999). Motion detection, labeling, data association and tracking in real-time on RISC computer. *Image Analysis and Processing, 1999.* (ss. 520–524). IEEE. doi:10.1109/ICIAP.1999.797648

Laudon, J., Golla, R., & Grohoski, G. (2009). *Multicore Processors and Systems.* (S. W. Keckler, O. Kunle, & H. P. Hofstee, Red.) Hämtat från http://www.springer.com/la/book/9781441902627

Laurenti, G., Djafarian, K., & Catan, H. (den 26 March 2002). *Patentnr US 6363470 B1*. Circular buffer management. Hämtat från https://www.patexia.com/us-patents/06363470

Lemieux, G. (2012). FPGA-based Embedded Supercomputing with the VectorBlox MXP Matrix Processor. Hämtat från http://vectorblox.com/wp/wp-content/uploads/VectorBlox-MXP-Intro-2012-12-11.pdf

Maxfield, C. (2004). *The Design Warrior's Guide to FPGAs: Devices, Tools and Flows.* Elsevier Inc. Hämtat från http://www.sciencedirect.com/science/book/9780750676045

McConnel, T. (2010). *ESC - Xilinx Extensible Processing Platform combines best of serial and parallel processing*. (T. McConnel, Redaktör, & EE|Times, Producent) Hämtat från http://www.eetimes.com/document.asp?doc_id=1313958

McGrath, D. (2006). *FPGA market to pass $2.7 billion by '10, In-Stat says*. (D. McGrath, Redaktör, & EE|Times, Producent) Hämtat från http://www.eetimes.com/document.asp?doc_id=1161569

Micheli, G. D., & Gupta, R. K. (1997). Hardware/software co-design. *Proceedings of the IEEE. 85(3)*, ss. 349-365. IEEE. doi:10.1109/5.558708

Microsoft. (2016). Stopwatch.Frequency Field. Hämtat från https://msdn.microsoft.com/en-us/library/system.diagnostics.stopwatch.frequency%28v=vs.110%29.aspx

Mirzaei, S., Hosangadi, A., & Kastner, R. (2006). FPGA Implementation of High Speed FIR Filters. Hämtat från http://cseweb.ucsd.edu/~kastner/papers/iccd06-fir_add_shift.pdf

Molander, B. (1998). *Vetenskapsfilosofi.* Thales.

Molyneaux, I. (2014). *The Art of Application Performance Testing: From Strategy to Tools* (2nd uppl.). (A. Oram, & B. Anderson, Red.) O'Reilly Media, Inc.

Nass, R. (April 2010). *Xilinx puts ARM core into its FPGAs.* (R. Nass, Red.) Hämtat från http://www.embedded.com/electronics-products/electronic-product-reviews/embedded-tools/4115523/Xilinx-puts-ARM-core-into-its-FPGAs

National Instruments. (2012). *FPGA Fundamentals*. Hämtat från http://www.ni.com/white-paper/6983/en/

Nelson, A. E. (2000). *Implementation of image processing algorithms on the FPGA hardware.* Master's thesis, Faculty of the Graduate School of Vanderbilt University. Nashville: Vanderbilt University. Hämtat från http://www.isis.vanderbilt.edu/sites/default/files/Nelson_T_0_0_2000_Implementa.pdf

NetEvents (Producent), & Roberts, G. (Regissör). (2006). *The History of Ethernet* [Film]. Hämtat från https://www.youtube.com/watch?v=g5MezxMcRmk 2014

Nobel, R. (2011). Actual throughput on Gigabit Ethernet. Hämtat från http://rickardnobel.se/actual-throughput-on-gigabit-ethernet/

Pederson, D. O. (2015). *A Framework for Hardware-Software Co-Design of Embedded Systems*. (D. O. Pederson, Redaktör, & F. o. EECS, Producent) Hämtat från http://embedded.eecs.berkeley.edu/Research/hsc/abstract.html

Prabhu, G. M. (2009). Dynamic Scheduling Techniques. Hämtat från http://www.cs.iastate.edu/~prabhu/Tutorial/PIPELINE/dynamSchedTech.html

Quinn, M. J. (2003). *Parallel Programming in C with MPI and OpenMP.* Oregon, USA: McGraw-Hili, en affärsenhet inom McGraw·Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY 10020. Hämtat från http://epcc.sjtu.edu.cn/wordpress/wp-content/uploads/2013/05/parallel-programming-in-c-with-mpi-and-openmp.pdf

Severance, A., & Lemieux, G. G. (2013). Embedded supercomputing in FPGAs with the VectorBlox MXP Matrix Processor. *2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)* (ss. 1-10). Montreal: IEEE. doi:10.1109/CODES-ISSS.2013.6658993

Stallings, W. (2012). *Appendix I Additional Instructions Pipeline Topics.* (W. Stallings, Red.) Hämtat från Computer Organization and Architecture, Ninth Edition: http://www.cs.vassar.edu/~jones/Stallings/I-Pipeline.pdf

Stevens, B. F., & Rudoff, A. M. (2004). *UNIX Network Programming - The Sockets Networking API* (3rd ed. uppl., Vol. vol. 1). Boston: Addison-Wesley.

Stevens, W. R. (1988). *UNIX Network Programming* (2nd uppl.). (E. s. Regina, Red.) Prentice Hall PTR, Prentice-Hall, Inc., A Simon & Schuster Company.

Telecom News Now. (2011). *My oh My – Ethernet Growth Continues to Soar; Surpasses Legacy*. (K. Campbell, Redaktör, & Telecom_News_Now, Producent) Hämtat från JSA Jaymie Scotto & Associates: http://www.jaymiescotto.com/jsablog/2011/07/29/my-oh-my-ethernet-growth-continues-to-soar-surpasses-legacy/

Terasic. (2009). *DE3 User Manual* . Hämtat från https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=China&CategoryNo=39&No=260&PartNo=4

Terasic. (2010). Terasic HSMC NET User Manual. Hämtat från http://www.mouser.com/pdfdocs/Terasic_HSMC_NET_User_Manual.pdf

Terasic. (2013). HSMC-NET User Manual. Hämtat från http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=71&No=355&PartNo=3

tPAD. (2010). *tPad User Manual.* Hämtat från http://www.cl.cam.ac.uk/teaching/1213/ECAD+Arch/labs/files/tPad_User_Manual.pdf 8 2013

Tsai, C.-J., Chen, Y.-T., & Tseng, C.-C. (2015). An efficient application processor architecture for multicore software video decoding. *IEEE Transactions on Circuits and Systems for Video Technology*, *25*(2), 325-338. doi:10.1109/TCSVT.2014.2329365

Urhan, T., & Franklin, M. J. (2001). Dynamic pipeline scheduling for improving interactive query performance. *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases.* Roma, Italy: Morgan Kaufmann Publishers Inc.

Wayback Machine. (1996). *Internet Archive Wayback Machine.* Hämtat från History of FPGAs: https://web.archive.org/web/20070412183416/http://filebox.vt.edu/users/tmagin/history.htm

Wiśniewski, R. (2009). *Synthesis of Compositional Microprogram Control Units or Programmable Devices.* Hämtat från http://zbc.uz.zgora.pl/Content/27955/Remigiusz_Wisniewski_Synthesis_of_CMCUs_for_Programmable_Devices.pdf

Xilinx. (1997). *Xilinx, Inc. History*. Hämtat från http://www.fundinguniverse.com/company-histories/xilinx-inc-history/

Xilinx. (2012). *RTL Design and IP Generation Tutorial, PlanAhead Design Tool.* Xilinx. Hämtat från http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/PlanAhead_Tutorial_RTL_Design_IP.pdf

Xilinx. (2014a). *Xilinx xupv2p reference designs.* Hämtat från Xilinx University Program: www.xilinx.com/univ/xupv2p demo ref designs.htm

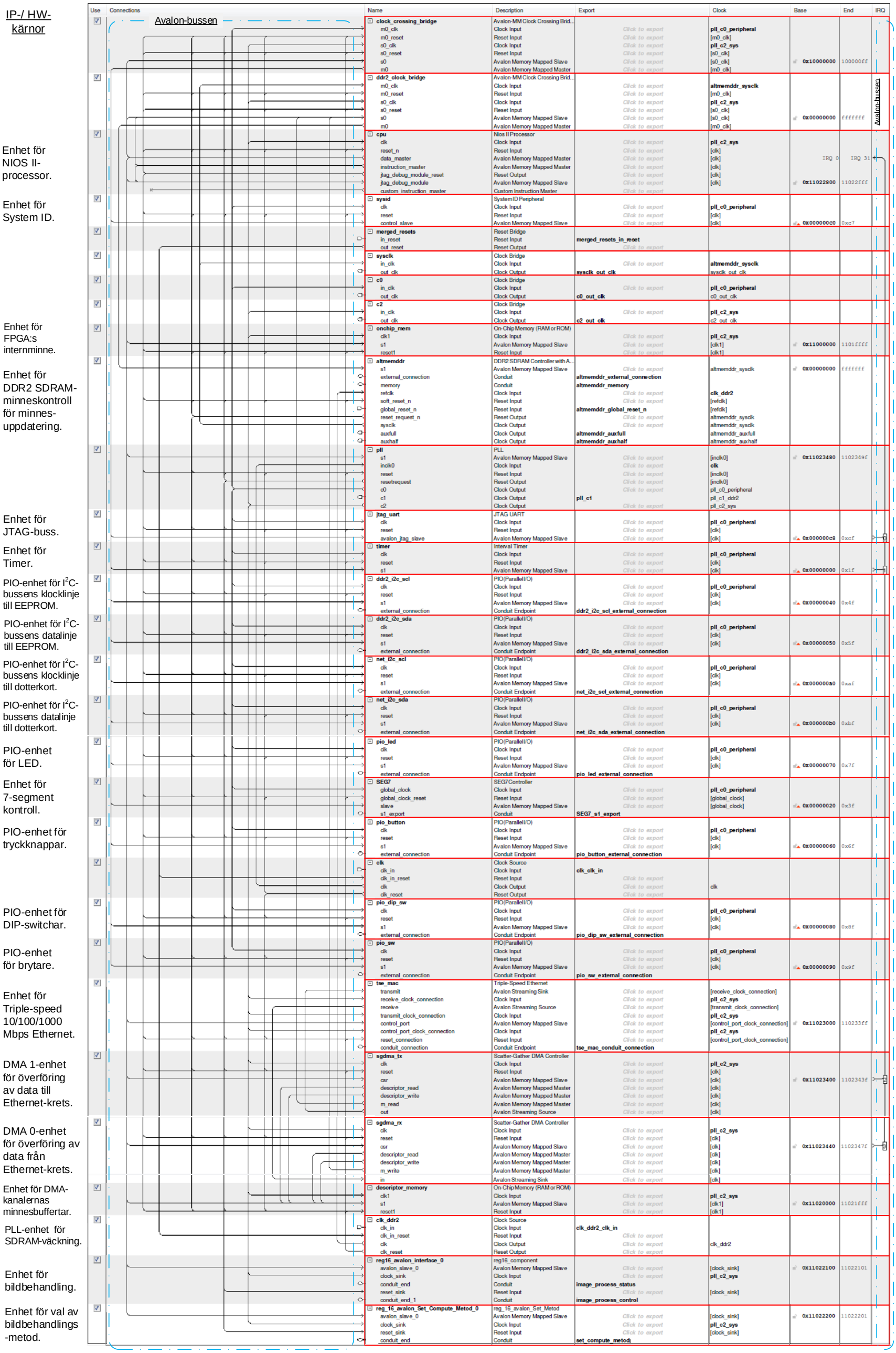
Xilinx. (2014b). *Zynq-7000 All Programmable SoC Overview.* Hämtat från http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

Xilinx. (2015). FPGA vs. ASIC. Hämtat från http://www.xilinx.com/fpga/asic.htm

ZIPcores. (2013). APPLICATION NOTE : ZC001, Using the Valid-Ready pipeline protocol. Hämtat från http://www.zipcores.com/datasheets/app_note_zc001.pdf

BILAGA 1. QSYS-KOMPONENTER I HÅRDVARAN

IP-/ HW-kärnor



Enhet för NIOS II-processor.

Enhet för System ID.

Enhet för FPGA:s internminne.

Enhet för DDR2 SDRAM-minneskontroll för minnes-uppdatering.

Enhet för JTAG-buss.

Enhet för Timer.

PIO-enhet för I²C-bussens klocklinje till EEPROM.

PIO-enhet för I²C-bussens datalinje till EEPROM.

PIO-enhet för I²C-bussens klocklinje till dotterkort.

PIO-enhet för I²C-bussens datalinje till dotterkort.

PIO-enhet för LED.

Enhet för 7-segment kontroll.

PIO-enhet för tryckknappar.

PIO-enhet för DIP-switchar.

PIO-enhet för brytare.

Enhet för Triple-speed 10/100/1000 Mbps Ethernet.

DMA 1-enhet för överföring av data till Ethernet-krets.

DMA 0-enhet för överföring av data från Ethernet-krets.

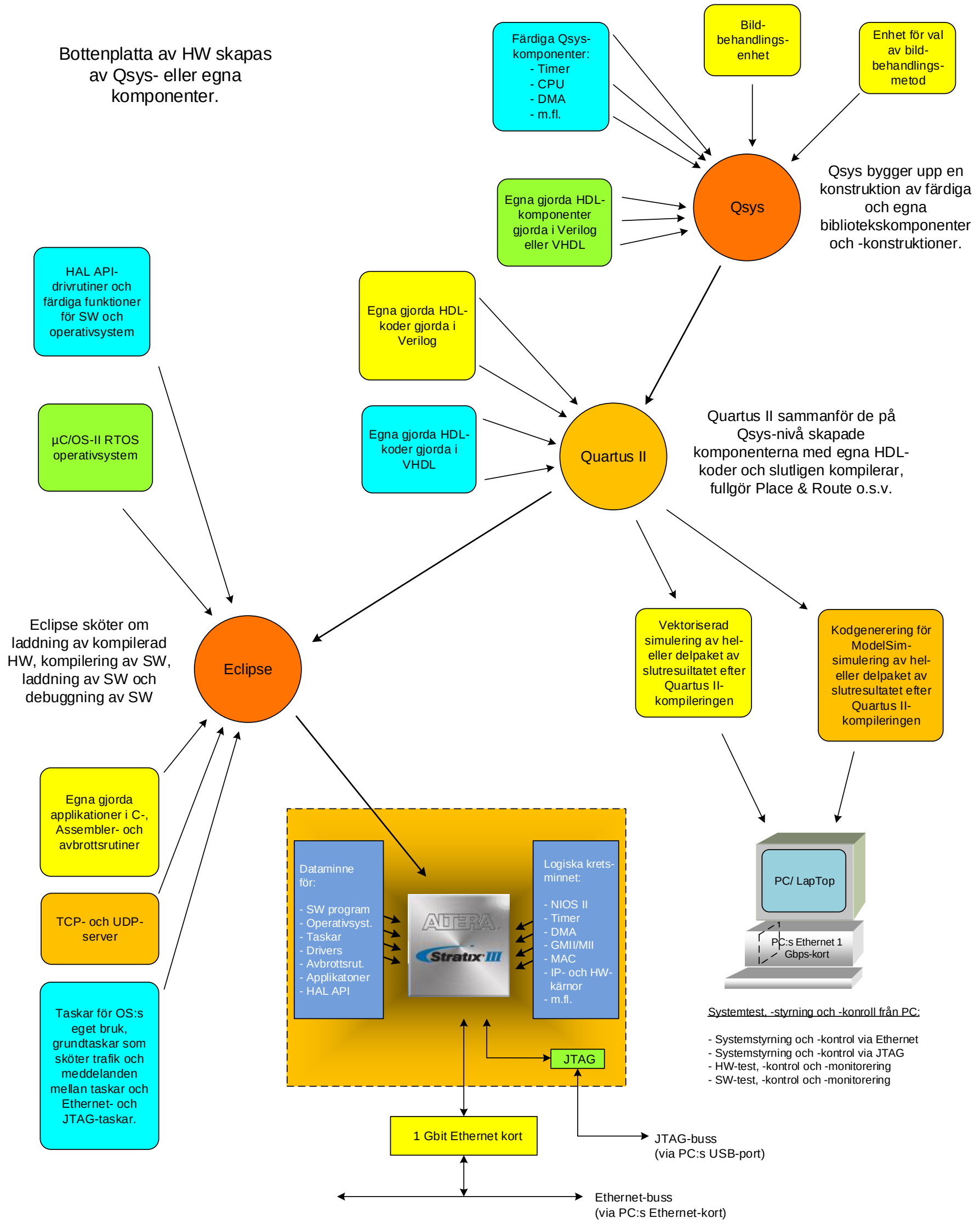
Enhet för DMA-kanalernas minnesbuffertar.

PLL-enhet för SDRAM-väckning.

Enhet för bildbehandling.

Enhet för val av bildbehandlings-metod.

BILAGA 2. ÖVERBLICK ÖVER VIKTIGA VERKTYG, HJÄLPMEDEL OCH KOMPONENTER



Via 1 Gbps Ethernet- och JTAG-buss kontrolleras, styrs och testas IPS från PC.

BILAGA 3. ADRESSKARTA AV HÅRDVARAN

System Contents | Address Map | Clock Settings | Project Settings | System Inspector | HDL Example | Generation

	sgdma_tx.descriptor_read	sgdma_tx.descriptor_write	sgdma_tx.m_read	sgdma_rx.descriptor_read	sgdma_rx.descriptor_write	sgdma_rx.m_write	clock_crossing_bridge.m0	ddr2_clock_bridge.m0	cpu.data_master	cpu.instruction_master
jtag_uart.avalon_jtag_slave							0x000000c8 0x000000cf			
onchip_mem.s1									0x11000000 0x1101ffff	
timer.s1							0x00000000 0x0000001f			
altmemddr.s1								0x00000000 0x0fffffff		
ddr2_i2c_scl.s1							0x00000040 0x0000004f			
ddr2_i2c_sda.s1							0x00000050 0x0000005f			
pio_lcd.s1							0x00000070 0x0000007f			
pio_button.s1							0x00000060 0x0000006f			
pll.s1									0x11023480 0x1102349f	
SEG7.slave							0x00000020 0x0000003f			
pio_dip_sw.s1							0x00000080 0x0000008f			
pio_sw.s1							0x00000090 0x0000009f			
tse_mac.control_port									0x11023000 0x110233ff	
sgdma_tx.csr									0x11023400 0x1102343f	
sgdma_rx.csr									0x11023440 0x1102347f	
descriptor_memory.s1	0x11020000 0x11021fff	0x11020000 0x11021fff		0x11020000 0x11021fff	0x11020000 0x11021fff				0x11020000 0x11021fff	
net_i2c_scl.s1							0x000000a0 0x000000af			
net_i2c_sda.s1							0x000000b0 0x000000bf			
clock_crossing_bridge.s0									0x10000000 0x100000ff	
ddr2_clock_bridge.s0			0x00000000 0x0fffffff			0x00000000 0x0fffffff			0x00000000 0x0fffffff	0x00000000 0x0fffffff
cpu.jtag_debug_module									0x11022800 0x11022fff	0x11022800 0x11022fff
sysid.control_slave							0x000000c0 0x000000c7			
reg16_avalon_interface_0.avalon_slav...									0x11022100 0x11022101	
reg_16_avalon_Set_Compute_Metod_0...									0x11022200 0x11022201	
ddr2_i2c_scl.s1 via clock_crossing_bri...									0x10000040 0x1000004f	
pio_lcd.s1 via clock_crossing_bridge									0x10000070 0x1000007f	
net_i2c_scl.s1 via clock_crossing_bridge									0x100000a0 0x100000af	
net_i2c_sda.s1 via clock_crossing_bri...									0x100000b0 0x100000bf	
SEG7.slave via clock_crossing_bridge									0x10000020 0x1000003f	
sysid.control_slave via clock_crossing...									0x100000c0 0x100000c7	
altmemddr.s1 via ddr2_clock_bridge			0x00000000 0x0fffffff			0x00000000 0x0fffffff			0x00000000 0x0fffffff	0x00000000 0x0fffffff
ddr2_i2c_sda.s1 via clock_crossing_br...									0x10000050 0x1000005f	
timer.s1 via clock_crossing_bridge									0x10000000 0x1000001f	
jtag_uart.avalon_jtag_slave via clock_c...									0x100000c8 0x100000cf	
pio_sw.s1 via clock_crossing_bridge									0x10000090 0x1000009f	
pio_button.s1 via clock_crossing_bridge									0x10000060 0x1000006f	
pio_dip_sw.s1 via clock_crossing_brid...									0x10000080 0x1000008f	

BILAGA 4. RTL-SCHEMA ÖVER HÅRDVARAN