INF3430 Høsten 2008

ChipScope PRO - En kort innføring

Innhold

Innledning	. 3
Generering av Chipscope kjerner	. 4
Generering av ICON (Integrated Controller) modul	. 4
Generering av ILA (Integrated Logic Analyzer) modul	. 6
Eksempeldesign	10
Instantiering av ChipScope modulene ICON og ILA i toppnivåarkitekturen	10
Scopetop.vhd	10
First.vhd	12
Bin2seg7.vhd	13
ChipScope Pro Analyze	15

Innledning

ChipScope PRO er et verktøy for å måle og monitorere interne signaler i en Xilinx FPGA for å kunne gjøre effektiv feilsøking (debugging) for å finne feil som simuleringer av en eller annen grunn ikke avdekker. Funksjonen er meget lik en tradisjonell logikkanalysator, der man kan studere mange signaler samtidig enten i form av timingdiagrammer eller lister (jfr. waveforms og lists i Modelsim). Hovedforskjellen er at en logikkanalysator benyttes for å måle og monitorere eksterne signaler. Man styrer innsamling av data ved å velge ut en samplingsklokke. En hovedutfordring man generelt har i elektronikk er å kunne effektivt fange øyeblikket der problemene viser seg. Å fange et slikt øyeblikk kalles trigging. Så det er viktig å kunne lage seg effektive triggebetingelser. Når triggebetingelsen(e) inntreffer blir data lagret og vi kan observere og feilsøke i timingdiagrammet som Chipscope lager. Triggebetingelsene kan variere fra enkle betingelser der man søker etter et bestemt mønster i dataene eller man lager mer avanserte betingelser f.eks. vi kan ønske at en bestemt mønster/begivenhet skal skje N antall ganger før datainnsamling starter eller man kan lage triggebetingelser som består av at en sekvens av begivenheter skal skje før data lagres. Vi kan også bestemme om og hvor mange sampel som skal lagres før triggebetingelsen(e) skal slå til. Dette ønsker man ofte fordi på den måten får man med seg historikken i signalene. Man kan tenke seg at man trigger på en kjent feil og at man kan da spore tilbake i tid fra tidspunktet feilen meldte seg og forhåpentligvis finne opphavet til feilen.

Før vi ser nærmere på bruk av ChipScope (Analyzer) må vi først klargjøre FPGA'en for bruk av ChipScope. Vi trenger å få laget to moduler som kalles ICON og ILA. Til å lage disse modulene benytter vi verktøyet ChipSCope Core Generator. Etter at modulene er generert kan vi instantiere dem i toppnivåarkitekturen vår eller vi kan benytte programmet ChipScope Core inserter. I dette eksemplet skal vi benytte instantiering i toppnivåarkitekturen.

Verktøyflyten for vår bruk av Chipscope kan summeres opp slik:

- 1. Oppretting av et ISE prosjekt
- 2. Generering av ICON modul (Integrated Controller)
- 3. Generering av ILA modul (Integrated Logic Analyzer)
- 4. Instantiering av ICON og ILA entiteter i toppnivåarkitekturen.
- 5. Syntese, place and route, Bitstream generering
- 6. Oppstart av ChipScope analyzer og oppsett av et analyzer prosjekt.

Generering av Chipscope kjerner

Generering av ICON (Integrated Controller) modul

Vi starter Chipscope Core generator ved Start=>Programs=>ChipScope Pro 9.1i => Xilinx ChipScope Pro Core Generator og får fram følgende bilde. Man går igjennom "Wizard'en" for å generere ICON og ILA modulene som vist i de påfølgende figurene.

ChipScope Pro Core Generator	
ChipScope Pro Core Generator	Core Type Selectio
Select Core Type To Generate ICON (Integrated Controller)	
O ILA (Integrated Logic Analyzer)	
 VIO (Virtual Input/Output Core) ATC2 (Agilent Trace Core 2) 	
Previous	>

Figur 1. Hovedvindu i ChipScope Core generator. Valg av ICON kjerne

ChipScope Pro C	ore Generator	<u>_ ×</u>	
ICON		General Options	
Design Files Output Netlist:	NINF3430\H2008\Laboppgaver\chipscope\Chipscope\icon.edn	Browse	
Device Settings Device Family:	Spartan3		
ICON Parameter	s		
	Number of Control Ports: 1		
	Disable Boundary Scan Component Instance		
	Boundary Scan Chain: USER1 💌		
	Disable JTAG Clock BUFG Insertion		
Enable Unused Boundary Scan Ports (only if necessary)			
Core Utilization	7 DE Counte 20 DEAM Counte		
Lut Count: 9	7 FF Count: 28 BRAM Count: 0)	
	< Previous Next >		

Figur 2. ICON. Valg av plassering og teknologi

ChipScope Pro Core Generator	_ 🗆 ×
ICON	Example and Template Options
HDL Example File Settings	
Generate HDL Example File	
HDL Language: VHDL 💌	
Synthesis Tool: Xilinx XST	
Batch Mode Argument Example File Settings	
✓ Generate Batch Mode Argument Example File (.arg)	
< Previous Generate Core	

Figur 3. ICON. Generering av eksempel på VHDL instatiering

👻 ChipScope Pro Core Generator	<u> </u>
ICON Cor	e Generation
Messages Enable BSCAN Instance: true BSCAN chain: USER1 Enable UTAG global clock buffer: true Enable unused BSCAN ports: false Force RPM Grid Usage: no Resource Utilization Estimate LUT:97 FF:28 BRAM:0	
Warning: EDIF Netlist being generated	
Post Processing EDIF netlist C:\IFI\INF3430\H2008\Laboppgaver\chipscope\Chipspe\	icon.edn
Generating constraints file C:\IFI\INF3430\H2008\Laboppgaver\chipscope\Chipscope\ico	n.ncf
Generating batch mode argument file C:\IFI\INF3430\H2008\Laboppgaver\chipscope\Chipscope\icon.arg	
ChipScope Pro Core Generator Version : 9.1.03i Build : 09103.7.81.1059	=
Example Usage File=C:\IFI\INF3430\H2008\Laboppgaver\chipscope\Chipscope\icon_xst_example.vhd	
Generating batch mode argument file C:\IF\INF3430\H2008\Laboppgaver\chipscope\Chipscope\icon_xst_vhdl_example.arg	
CORE GENERATION COMPLETE	
	▼
< Previous Start Over	

Figur 4. ICON. Oppsummeringsvindu. Sjekk at path'er stemmer

Generering av ILA (Integrated Logic Analyzer) modul

ChipScope Pro Core Generator	Core Type Selection
Select Core Type To Generate	
O ICON (Integrated Controller)	
ILA (Integrated Logic Analyzer)	
○ VIO (Virtual Input/Output Core)	
○ ATC2 (Agilent Trace Core 2)	
	4.5

Figur 5. Valg av ILA (Integrated Logic Analyzer)

	General Options
	General Options
Design Files—	
Output Netlist	\IFI\INF3430\H2008\Laboppgaver\chipscope\Chipscope\ila.edn Browse
Device Settings	
Device Family:	Spartan3 🔽 Use SRL16s 🗹 Use RPMs
Clock Settings	
Sample On R	ising Edge Of Clock

Figur 6. ILA. Valg av filplassering., teknologi og flanke på samplingsklokke

ILA	Trigger Port Optic
Trigger Input and Match Unit Sett	tings
Number of Input Trigger Ports:	1 ▼ Number of Match Units Used: 2
TRIGO: Trigger Width:	32 Match Type: Extended w/edges
# Match Units:	2 Bit Values: 0, 1, X, R, F, B
Counter Width:	9 v Functions: =, <>, >, >=, <, <=
Trigger Condition Settings	
rngger conation octango	
Enable Trigger Sequencer	Max Number of Sequencer Levels: 4
Enable Trigger Sequencer	Max Number of Sequencer Levels: 4
Enable Trigger Sequencer	Max Number of Sequencer Levels: 4
Enable Trigger Sequencer	Max Number of Sequencer Levels: 4
Enable Trigger Sequencer Storage Qualification Condition S Enable Storage Qualification Trigger Output Settings	Max Number of Sequencer Levels: 4
Enable Trigger Sequencer Enable Condition Condition S Enable Storage Qualification Findle Storage Qualification Trigger Output Settings Enable Trigger Output Port	Max Number of Sequencer Levels: 4
Enable Trigger Sequencer Storage Qualification Condition S Enable Storage Qualification Trigger Output Settings Enable Trigger Output Port Core Utilization	Max Number of Sequencer Levels: 4
	Max Number of Sequencer Levels: 4

Figur 7. ILA. Valg av triggerport og triggeregenskaper

LA			Data Port Options
Data Port Settir	as		
Data Depth: 40	196 🔻 Samples	Aggregate Data Wid	tth: 32
🗹 Data Same	e As Trigger		
	🖌 Include TF	RIGO port (width=32)	
Core Utilization			
Lut Count:	340 FF Count:	387 BRAM Count:	9

Figur 8. ILA. Valg av tracedybde og velger data samme som trigger

🗐, ChipScope Pro Core Generator	
ILA	Example and Template Options
HDL Example File Settings	
✓ Generate HDL Example File	
HDL Language: VHDL 💌	
Synthesis Tool: Xilinx XST	
Bus/Signal Name Example File Settings	
✓ Generate Bus/Signal Name Example File (.cdc)	
✓ Generate Batch Mode Argument Example File (.arg)	
Previous Generate Core	

Figur 9. ILA. Generering av eksempel på instatiering

🕲 ChipScope Pro Core Generator	<u>_ ×</u>		
ILA Core G	eneration		
Messages Trigger Sequencer Type : None			
External capture : disabled Force RPM Grid Usage: no Resource Utilization Estimate LUT:340 FF:387 BRAM:9			
Warning: EDIF Netlist being generated			
Post Processing EDIF netlist C:\\F\\NF3430\H2008\Laboppgaver\chipscope\Chipscope\ila.e	edn		
Generating constraints file C:\IFI\INF3430\H2008\Laboppgaver\chipscope\Chipscope\la.ncf			
Generating CDC file C:\/FI\/NF3430\H2008\Laboppgaver\chipscope\Chipscope\ila.cdc			
Generating batch mode argument file C:\IFI\INF3430\H2008\Laboppgaver\chipscope\Chipscope\ila.arg			
ChipScope Pro Core Generator Version : 9.1.03i Build : 09103.7.81.1059			
Example Usage File=C:\\FI\INF3430\H2008\Laboppgaver\chipscope\Chipscope\\la_xst_example.vhd	=		
Generating batch mode argument file C:\FI\INF3430\H2008\Laboppgaver\chipscope\Chipscope\ila_xst_vhdl_example.arg			
CORE GENERATION COMPLETE			
	•		
< Previous Start Over			

Figur 10. ILA. Oppsummeringsvindu

Name	Size	Туре	
🔤 ila.cdc	3 KB	CDC File	
ila_xst_vhdl_example.arg	1 KB	ARG File	
ila_xst_example.vhd	2 KB	MTI vhdl	
🔤 ila.ncf	1 KB	NCF File	
🗐 ila.edn	1 288 KB	Text Document	
🔤 ila.arg	1 KB	ARG File	
icon_xst_vhdl_example.arg	1 KB	ARG File	
icon_xst_example.vhd	2 KB	MTIyyhdl	
🔤 icon.ncf	1 KB	NCF Mile	
🗐 icon.edn	101 KB	Text Document	
🔤 icon.arg	1 KB	ARG File	
kokebok		File Folder	
🚔 eks		File Folder	
Figur 11. Genererte ICON og ILA filer			

Eksempeldesign

Instantiering av ChipScope modulene ICON og ILA i toppnivåarkitekturen

Vi skal benytte ChipScope til å se på signalene i et enkelt eksempel. Eksempeldesignet består av en teller (first) som er koblet sammen med en sjusegmentdekoder funksjon. Telleren er fire bit og verdien av telleren vises på det ene sjusegmentdisplayet på testkortet.

Legg merke til at hver av entitetene first og bin2seg7 er utstyrt med et sett ekstra signaler, chip_scope_out kan kobles til ILA modulen. Hvilke signaler man tar ut på chip_scope_out avhenger av hva man er interessert i å se på og hva slags problem man står ovenfor. Ved å ha disse signalene tilgjengelige har man laget en struktur for testing og debugging. I vårt eksempel er det så små moduler at vi har tatt ut alt av signaler, mens i et virkelig design vil den normale situasjonen være at man må begrense seg. Man skal være klar over at bruk av Chipscope spiser opp betydelige mengder av ressursene man har i FPGA'en.

Instantieringen av ICON og ILA er et eksempel på en "Black box" instatiering der det ikke ligger noe VHDL kode under modulene, men foreligger som edn-filer (nettlister på edif-format.)¹ generert av Chipscope core generator og vil være en del av kildefilene i designet.

Scopetop.vhd

```
scopetop.vhd
library IEEE;
use IEEE.std logic 1164.all;
entity scopetop is
 port
  (
                    : in std_logic; -- Klokke fra bryter CLK1/INP1
    clk
                 : in std_logic; -- Global Asynkron Reset
: in std_logic; -- Synkron reset
    reset
    load
    imp in std_logic_vector(3 downto 0); -
max_count : out std_logic; -- Viser telleverdi
seg7 op
                    : in std_logic_vector(3 downto 0); -- Startverdi
    seg7_en
                    : out std_logic_vector(3 downto 0);
                   : out std_logic_vector(6 downto 0)
    qfedcba
  );
end scopetop;
architecture rtl of scopetop is
  -- Område for deklarasjoner
  signal count : std_logic_vector(3 downto 0);
  component first is
    port
    (
      clk
                        : in std_logic; -- Klokke fra bryter CLK1/INP1
                       : in std_logic; -- Global Asynkron Reset
      reset
```

¹ EDIF = Electronic Design Interchange Format

```
scopetop.vhd
                      : in std_logic; -- Synkron reset
      load
      inp
                      : in std_logic_vector(3 downto 0); -- Startverdi
      count
                     : out std_logic_vector(3 downto 0); -- Telleverdi
      max_count
                  : out std_logic; -- Viser telleverdi
      chip_scope_out : out std_logic_vector(7 downto 0) --Chipscope outputs
    );
  end component first;
  component bin2seg7 is
    port
    (
      bin
                     : in std_logic_vector(3 downto 0);
      gfedcba
                    : out std_logic_vector(6 downto 0);
      chip_scope_out : out std_logic_vector(6 downto 0)
    );
  end component bin2seq7;
  --Chipscope spesifikk seksjon
  signal control0 : std_logic_vector(35 downto 0);
  signal trig0 : std_logic_vector(31 downto 0);
  component icon
    port
    (
      control0 : out std_logic_vector(35 downto 0)
    );
  end component;
  component ila
   port
    (
                : in std_logic_vector(35 downto 0);
      control
                : in std_logic;
: in std_logic_vector(31 downto 0)
      clk
     trig0
    );
  end component;
begin
seg7_en <= "1110";</pre>
counter: first
port map
(
 clk
               => clk,
  reset
               => reset,
 load
               => load,
 inp
                => inp,
 count => count,
max_count => max_count,
 chip_scope_out => trig0(13 downto 6)
);
seg7dekoder: bin2seg7
port map
(
 bin
                => count,
  gfedcba
               => gfedcba,
  chip_scope_out => trig0(20 downto 14)
);
```

```
--Chipscope spesifikk seksjon
trig0(0) <= reset;</pre>
triq0(1)
                  <= load;
trig0(5 downto 2) <= inp;</pre>
i_icon : icon
port map
(
 control0 => control0
);
i_ila : ila
port map
(
 control => control0,
 clk => clk,
trig0 => trig0
);
end rtl;
```

First.vhd

```
first.vhd
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric std.all;
entity first is
 port
  (
                     : in std_logic; -- Klokke fra bryter CLK1/INP1
    clk
                   : in std_logic; -- Global Asynkron Reset
: in std_logic; -- Synkron reset
: in std_logic_vector(3 downto 0); -- Startverdi
    reset
    load
    inp
    count : out std_logic_vector(3 downto 0); -- Telleverdi
max_count : out std_logic; -- Viser telleverdi
    chip_scope_out : out std_logic_vector(7 downto 0) --Chipscope outputs
  );
end first;
-- Arkitekturen under beskriver en 4-bits opp-teller. Når telleren når
-- maksimal verdi går signalet MAX_COUNT aktivt.
architecture rtl of first is
  -- Område for deklarasjoner
  signal count i : unsigned(3 downto 0);
  signal max_count_i : std_logic;
begin
  -- Her starter beskrivelsen
```

scopetop.vhd

```
first.vhd
```

```
counter :
  process (reset,clk)
  begin
    if(reset = '1') then
      count i <= (others => '0');
    elsif rising_edge(clk) then
      -- synkron reset
      if load = '1' then
        count_i <= unsigned(inp);</pre>
      else
        count_i <= count_i + 1;</pre>
      end if;
    end if;
  end process counter;
  count <= std_logic_vector(count_i);</pre>
  -- concurrent signal assignment
  max_count_i <= '1' when count_i = "1111" else '0';</pre>
  max_count <= max_count_i;</pre>
  --Signaler som kan vises i ChipScope hvis ønskelig
  chip_scope_out(3 downto 0) <= std_logic_vector(count_i);</pre>
  chip_scope_out(4)
                               <= max_count_i;
end rtl;
```

Bin2seg7.vhd

```
bin2seg7.vhd
Library IEEE;
use IEEE.Std_Logic_1164.all;
use IEEE.numeric_std.all;
entity bin2seg7 is
port
  (
    bin
                   : in std_logic_vector(3 downto 0);
                : out std_logic_vector(6 downto 0);
    qfedcba
    chip_scope_out : out std_logic_vector(6 downto 0)
  );
end bin2seg7;
architecture rtl of bin2seg7 is
  signal gfedcba_i : std_logic_vector(6 downto 0);
begin
  bin2seg7process:
  process(bin)
  begin
    case bin is
       -- segments: gfedcba
      when "0000" => gfedcba_i <= "1000000";</pre>
```

```
bin2seg7.vhd
            "0001" => gfedcba_i <= "1111001";
      when
            "0010" => gfedcba_i <= "0100100";
      when
      when "0011" => gfedcba_i <= "0110000";</pre>
      when "0100" => gfedcba_i <= "0011001";</pre>
      when "0101" => gfedcba_i <= "0010010";</pre>
      when "0110" => gfedcba_i <= "0000010";</pre>
      when "0111" => gfedcba_i <= "1011000";</pre>
      when "1000" => gfedcba_i <= "0000000";</pre>
      when "1001" => gfedcba_i <= "0010000";</pre>
      when "1010" => gfedcba_i <= "0001000";</pre>
      when "1011" => gfedcba_i <= "0000011";</pre>
      when "1100" => gfedcba_i <= "1000110";</pre>
      when "1101" => gfedcba_i <= "0100001";</pre>
      when "1110" => gfedcba_i <= "0000110";</pre>
      when others => gfedcba_i <= "0001110";</pre>
    end case;
 end process;
 gfedcba
                   <= gfedcba_i;
  --Signaler som kan vises i ChipScope hvis ønskelig
 chip_scope_out <= gfedcba_i;</pre>
end rtl;
```

ChipScope Pro Analyze

Vi kan starte ChipScope Pro analyze (fra nå av omtalt som ChipScope) enten innenfra ISE eller utenfra på samme måte som Core generator. Bruk av Chipscope bygger på at man har fått generert en bistreamfil som inneholder ICON og ILA modulene, og der ILA modulen er koblet til signaler fra designet vårt.

Når man har startet ChipScope kan man enten starte et nytt ChipScope prosjekt eller åpne et eksisterende. Alle innstillinger man har gjort blir lagret i prosjektet.

Det første man må gjøre er å konfigurere "download"-kabelen man benytter og detektere kretsene i JTAG-kjeden på kortet. Vi konfigurer denne ved å velge JTAG Chain i menyen. Vi velger Xilinx Parallel Cable og Autodetect. Dersom kabelen er koblet til og strøm er satt på kortet vil alle kretser som er koblet i JTAG-kjeden bli automatisk detektert.



Figur 12. Kabeloppsett. Valg av download kabel

ChipScope P	ro Analyzer [new project]		×							
?	-Parallel Cable Selection									
	 Xilinx Parallel IV 									
	Auto Detect Cable Type	2								
	-Parallel Cable Parameters-									
	Speed:	Port:								
	5 MHz	LPT1								
	OK Cance	el								

Figur 13. Kabeloppsett. Velg autodetect (eller Parallel III)

Ch	ipScope	Pro Analyzer					×
	JTAG C	hain Device Order					
	Index	Name	Device Name	IR Length	Device IDCODE	USERCODE	1
	0	MyDevice0	XC3S200	6	01414093		
	1	MyDevice1	XCF02S	8	05045093		
						Advanced >>	
							_
		L.	Cancel	Read USE	ERCODES		

Figur 14. Kabeloppsett. Detekterte kretser i JTAG kjeden.

Etter FPGA'en vår er detektert skal vi konfigurere den ved å laste ned bitstreamfilen vi på forhånd har laget. Vi velger ut FPGA'en i JTAG chain (XC3S200), høyreklikk og velger Configure. Vi "browser" oss fram til ønsket bitstreamfil og velger Ok. Da vil FPGA'en bli konfigurert med innholdet i denne.

ChipScope Pro Analyze	er [new project]
<u>File View</u> <u>J</u> TAG Chain	<u>D</u> evice <u>W</u> indow <u>H</u> elp
# P	
New Project	2
JTAG Chain	
DEV:0 MyDevice0 (XC)	Demonstra
DEV:1 MyDevice1 (XC	<u>R</u> ename
	<u>Contigure</u>
	Show IDCODE
	Show USERCODE
	Show Configuration Status
	Show JTAG Instruction Register

Figur 15. Konfigurere FPGA'en

ChipScope Pro Analyzer [new project]
<u>File View J</u> TAG Chain <u>D</u> evice <u>W</u> indow <u>H</u> elp
New Project Image: State
ChipScope Pro Analyzer [new project]
JTAG Configuration
File:
Directory: C:\IFI\INF3430\H2008\LaboppgaverIchipscope\Chipscope
Select New File
OK Cancel Look in: C Chinscope
My Recent Documents Desktop

Figur 16. Valg av bitstreamfil

Etter at konfigureringen er ferdig og bitstreamfilen inneholder ICON og ILA modulene vil følgende bilde dukke opp.



Figur 17. ChipScope etter at konfigurasjon er lastet opp



Figur 18. Detaljert utsnitt over ILA til vår krets

Signalene i ILA modulen vil på dette tidspunktet inneholde kryptiske navn. Så det første vi bør gjøre er å gi enkeltsignalene nytt navn i forhold til hvilke signaler de faktisk er koblet til inne i FPGA'en. Dette er en engangsjobb og dette oppsettet vil bli lagret i ChipScope prosjektfilen.



Figur 19. Navneendring til faktiske signalnavn

Signals: DEV: 0 UNIT: 0
P Data Port
🗠 abcdefg
∽ count_
🕶 inp
- CH: 0 RESET
- CH: 1 LOAD
- CH: 2 inp0
- CH: 3 inp1
— CH: 4 inp2 Τ
— CH: 5 inp3 🚽
– CH: 6 count_0
- CH: 7 count_1
- CH: 8 count_2
- CH: 9 count_3
— CH: 10 max_count
Figur 20. Etter navneendring

Det neste vi nå gjør er å eventuelt gruppere sammen signaler for å definere busser. F.eks. så passer det å gruppere sammen inp-signalene til bussen inp_(navn foreslått av verktøy) og count signalene til bussen count_. Det er lettere å studere hex-verdier enn enkeltbit.



Figur 21. Definering av bus

Etter at vi har satt opp alle ønskede signalnavn og definert ønskede busser setter vi opp triggebetingelser. Da velger vi trigger setup i JTAG Chain vinduet og gjerne forstørrer trigger setup vinduet som i figuren under. I vårt eksempel velger vi å trigge på stigende flanke av signalet max_count. Videre ønsker vi å ikke benytte alle 4096 sampler vi har tilgjengelige i ILA, men avgrenser "tracen" vår til 64 sampel. Videre ønsker vi å lagre 32 sampel før (pre-store) triggebetingelsen inntreffer. Dette gjøres ved å sette "position" til 32.



Figur 22. Trigger setup

Man starter datainnsamling ved å trykke på "Play" knappen som vist på neste figur:



Når triggebetingelsen er sann vil datainnsamlingen stoppe og vi vil få vist et bilde tilsvarende figuren under. Legg merke til at vi har tre kursorer i bildet. En T-kursor som viser triggepunktet og en O- og X-kursor. Vi kan endre posisjonene på O- og X-kursorene og vi kan zoome inn området dekket av intervallet mellom O- og X-kursorene.

Man ønsker å avslutte med forced trig trykk "T!" når man tror at triggebetingelsen burde vårt oppfylt eller man bare vil sample for eksempel etter reset for å se på initialverdiene.



Figur 24. Waveformvindu med T-, O- og X-kursorer

🖉 🖗 Waveform - DEV	:0 MyDev	ice0 (XC3S	200) UNIT	:0 MyILAO	(ILA)												
		-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	n	1	2	3	
Bus/Signal	X C		Ö	i	- ī	i	- î	ī	i	ī	- ī	i	Ū	i	ī	ī	
- RESET	0	0															
10001		×															_
°~ inp	0	0									0						_
- LOAD	0	0															
← count	6	5 4	χ 5	X fi	χ 7		(9	XAX	вΧ	cγ	D _N	F	XE	X	(1	χ 2	γ_
max count	0	0	~	~							3						~
a phode for	02 1	2 10	V 12	V 02	V 50	00	10	V 00 V	02 V	46 V	24 000	nt_=D Wi	indow=0	Sample=-2	70	V 24	~
ascuerg	02 1	- 19	Λ <u>1</u> 2	Λ 02	V 20	<u> </u>	10	<u></u> ^		40A	/		<u>^ UE</u>	_ <u></u> 40	(78	<u>A 24</u>	~_
- inp0	0	0											_				_
— inpl	0	0															
— inp2	0	0															
— inp3	0	0															
- count 0	0	1		1		1 1		1 [Г				7		7	
- count 1	1			_		1							-	_	·		-
counc_1			_	_		1											
2	1	1															_
- count_3	0	0	-			J										_	
— a	0	0							L								
- f	1	1			1												
— e	0	0							Г					7			٦
- d	0	0	1			1								-		1	
				1		1		י נ ו							1		-
		-			_										1		_
- ь	0	•														_	
- a	0	0								L							
- DataPort[21]	0	0															
DataPort[22]	0	0															
DataPort[23]	0	0															
- DataPort[24]	0	0															
Data Data Constant																	
DataPort[25]	0	-											_				_
DataPort[26]	0	0															_
DeteBort[271												_					_
																	_
									X:	7	• •	0:-1	10	• •	∆(X-0):	17	
ore to be armed																	_

Eile View JTAG Chain Device Trigger Setup Waveform Listing Window Help

Figur 25. Waveformvindu zoomet til området dekket av O- og X-kursor

Man kan også velge å presentere dataene som en liste av sampler. Vi kan velge ut enkeltsignaler, høyreklikke og velge add to listing. Da får man et bilde tilsvarende figuren under.

	Listing - DEV:) MyDevice0 (KC3S200) UN	IT:0 MyILAO (I	LA)	
	Sample	abcdefg	count	inp	RESET	LOAD
	0	OE	F	0	0	0
	1	40	0	0	0	0
1	2	79	1	0	0	0
	3	24	2	0	0	0
	4	30	3	0	0	0
	5	19	4	0	0	0
	6	12	5	0	0	0
	7	02	6	0	0	0
	8	58	7	0	0	0
	9	00	8	0	0	0
	10	10	9	0	0	0
	11	08	A	0	0	0
1	12	03	В	0	0	0
	13	46	с	0	0	0
	14	21	D	0	0	0
	15	06	E	0	0	0
	16	OE	F	0	0	0
	17	40	0	0	0	0
	18	79	1	0	0	0
	19	24	2	0	0	0
	20	30	3	0	0	0
	21	12	4	0		0
	23	02	6	0	0	0
	24	58	7	i ii	0	0
	25	00	8	0	0	0
	26	10	9	0	0	0
3	27	08	A	0	0	0
	28	03	В	0	0	0
	29	46	с	0	0	0
	30	21	D	0	0	0
	31	06	E	0	0	0
	32	OE	F	0	0	0
	33	40	0	0	0	0
	34	79	1	0	0	0
	35	24	2	0	0	0
	36	30	3	0	0	0
	37	19	4	0	0	0
	38	12	5	0	0	0
	39	02	6	0	0	0
	40	58	7	0	0	0
	41	00	8	0	0	0
	42	10	9	0	0	0
	43	08	A	0	0	0
	44	03	В	0	0	0
	45	1.05		· • •		· •

Figur 26. Listing vindu

Vi kan eksportere inneholde av waveform og listing vinduene til forskjellige formater. F.eks. kan man ønske å benytte et trace til å analysere dataene i et annet program, som input Modelsim eller annet. Eller man kan ønske å benytte waveforms som dokumentasjon. Figuren under viser eksport av listing til en ascii-fil.

Export Signals		X
Format VCD ASCII FBDF	Core DEV:0 MyDevice0 (XC3S200) UNIT:0 MyILA0 (ILA) Signals to Export Listing Signals/Buses	
	Export Cancel	

Figur 27. Eksport av listing til ASCII-fil

For ytterligere informasjon henvises til Xilinx Chipscope Pro Users manual som kan åpnes i programgruppen for ChipScope Pro eller fra Xilinx sin hjemmeside: http://www.xilinx.com/support/documentation/sw_manuals/chipscope_pro_sw_cores_9_1i_ug029.pdf