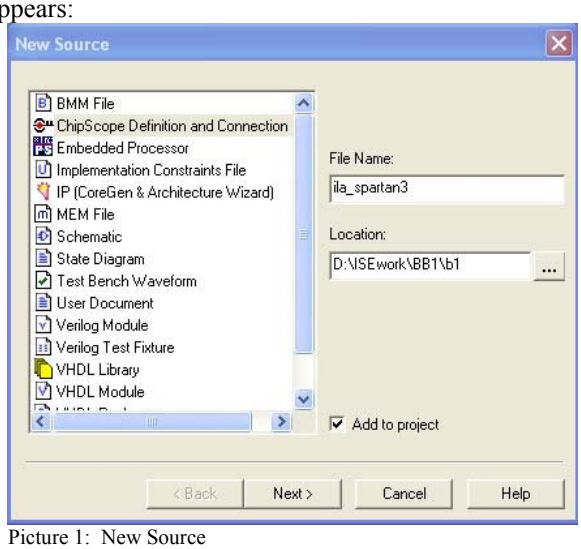
Hochschule für Angewandte
Wissenschaften Hamburg
FAKULTÄT TECHNIK UND INFORMATIK

# Tutorial for „ChipScope Pro“

The Xilinx Tool „ChipScope Pro“ facilitates the implementation of a Logic Analyser Core on the Spartan III board.

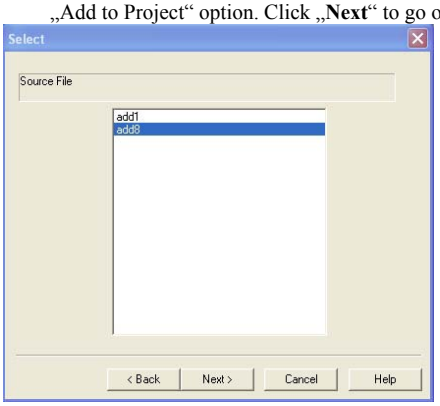
## 1.1 Creation of a Logic Analyser Core

A „Core“ file ~.cdc can be created with → **Project → New Source.**
The following menu appears:

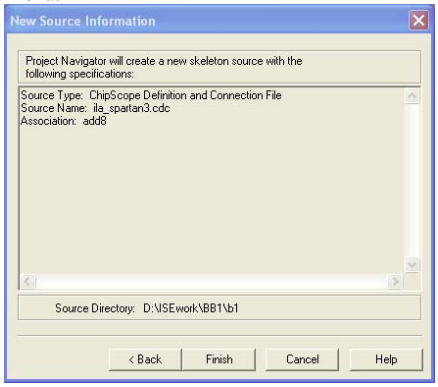



Picture 1: New Source

Choose „ChipScope Definition and Connection“, give a file name and activate the „Add to Project“ option. Click „**Next**“ to go on to the next menu:

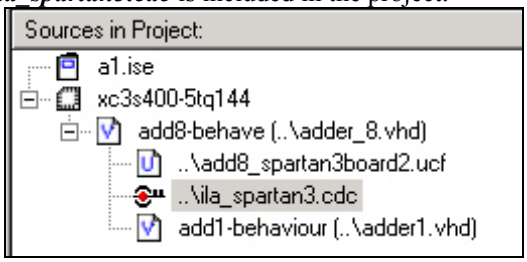



Picture 2: Assign to the top entity.
Choose the name of the top entity and click “Finish“.




Picture 3: Summary

This way the „Core“ file ***ila_spartan3.cdc*** is included in the project.




Picture 1: Included file ila_spartan3.cdc

## 1.2 Usage of the system clock as LA clock

In order for the system clock to be available to the „ChipScope“ Logic Analyser, the entity must be listed under ports in the VHDL source code:

```
SYSCLK      : in bit;
N_SYSCLK    : out bit;
```

In the architecture, for instance:

```
N_SYSCLK <= not (SYSCLK);
```

In the ~.ucf file:

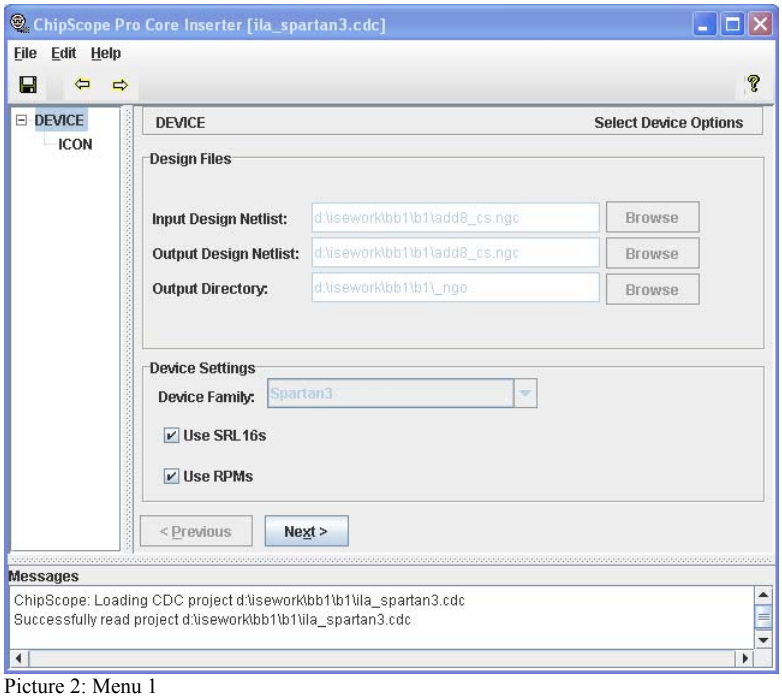
```
NET SYSCLK LOC = P127;
NET N_SYSCLK LOC = P46;
```

If a higher frequency is desired as circuit timing, a frequency increase up to approx. 400 MHz can be reached by using the DCM-module. For details see Appendix A.

## 1.3 Adaptation with „Core Inserter“

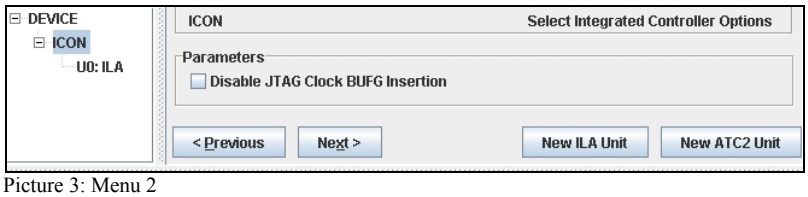
By double-clicking on the symbol of the ~.cdc file ila_spartan3.cdc, the „Core Inserter“ is invoked and 4 menus appear:

First menu: The netlist paths for input and output and the output directory are specified. The device family is set to „SPARTAN 3“.
The options: Use SRL 16s ( SeriellShift LUT 16 bit) and RPMs (relationally placed macros) should be activated.




Picture 2: Menu 1

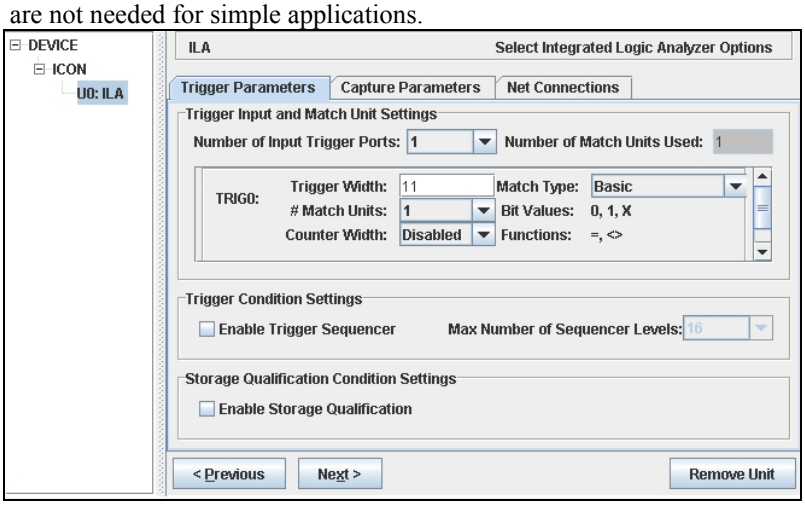
Second menu: Disable JTAG Clock BUFG Insertion can remain inactive .




Picture 3: Menu 2

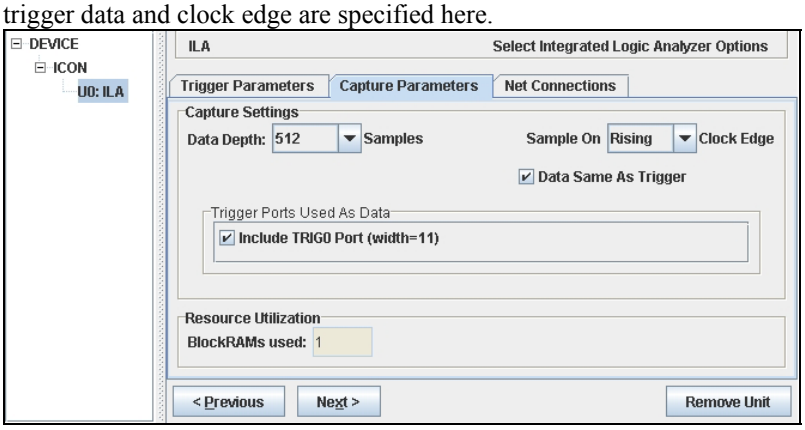
Third menu: The trigger parameters are listed according to data length and allocation.

Trigger counters, trigger sequencer and storage qualification are not needed for simple applications.



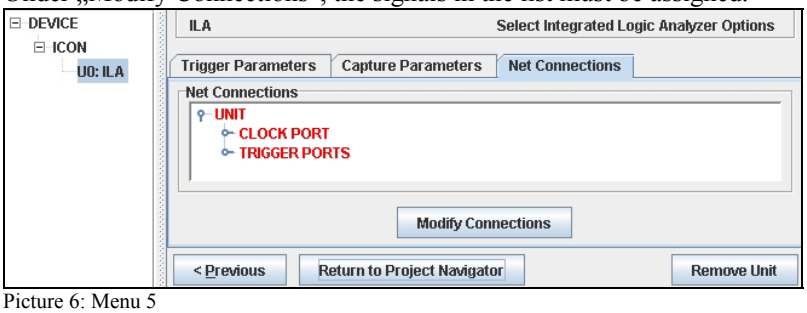

Picture 4: Menu 3

Fourth menu: Data registration parameters such as memory depth and data width of the trigger data and clock edge are specified here.



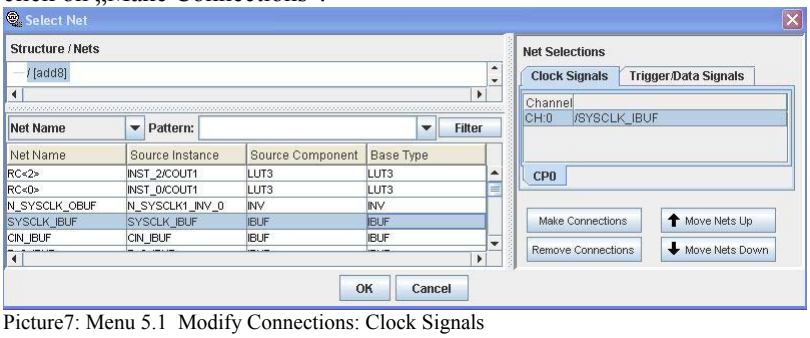

Picture 5: Menu 4

Fifth menu: Under „Modify Connections“, the signals in the list must be assigned.



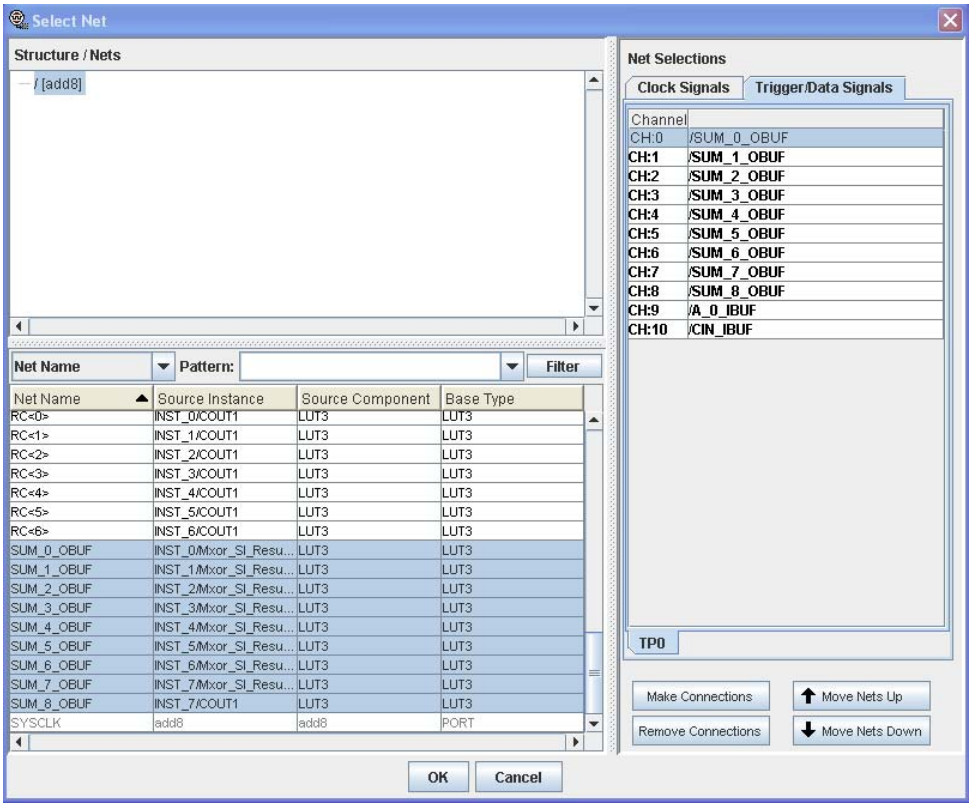

Picture 6: Menu 5

The desired signal is marked in the left side table, and the channel number on the right side; click on „Make Connections“.




Picture7: Menu 5.1 Modify Connections: Clock Signals

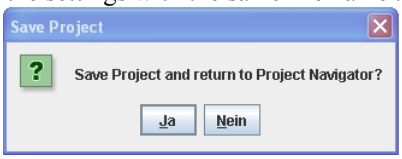
By clicking on „Net Name“, the list of signal names is sorted alphabetically; a larger set of signals can be marked in the „Trigger Data Signals“ list.



Picture 8: Menu 5.2 Modify Connections: Trigger Data Signals

Click **OK** to return to menu 5, and from there with 

to ISE main menu, by saving the settings with the same file name .

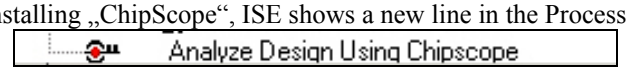


Picture 9: saving „Core“ project

Next, the implementation is started and the Spartan 3 board is programmed.
During this action, you must watch to make sure that the progress bar runs steadily to the end.
Otherwise, close „ChipScope“ and „Impact“ (without saving) and start „Impact“ again.

## 1.4 Start of„ChipScope“

After installing „ChipScope“, ISE shows a new line in the Process window:



Picture 10: Header of „ChipScope“

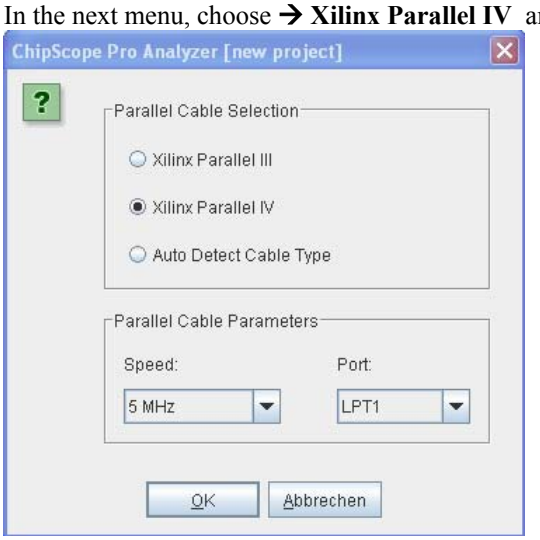
Double-click to start„ChipScope“.
The main window of „ChipScope Pro“ opens:

Picture 11: „ChipScope“ main window

First, the JTAG connection must be made: **JTAG Chain → Xilinx parallel Cable** :


Picture 12: → JTAG Chain → Xilinx Parallel Cable

In the next menu, choose → **Xilinx Parallel IV** and click **OK**:

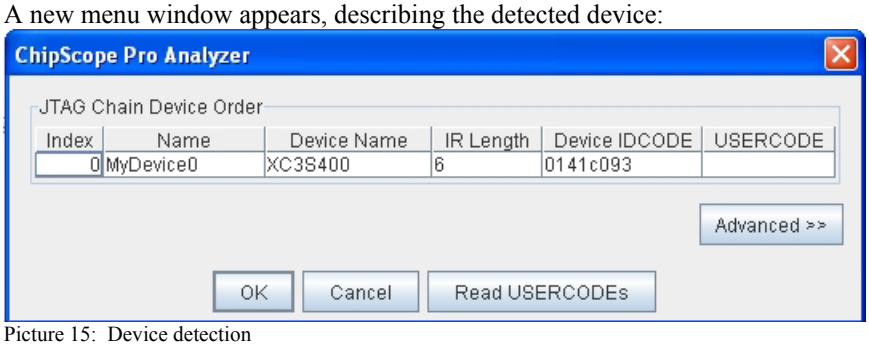



Picture 13: Parallel Cable Selection

Ignore the „Windows Firewall Alert“ message that appears and close it with **OK**:

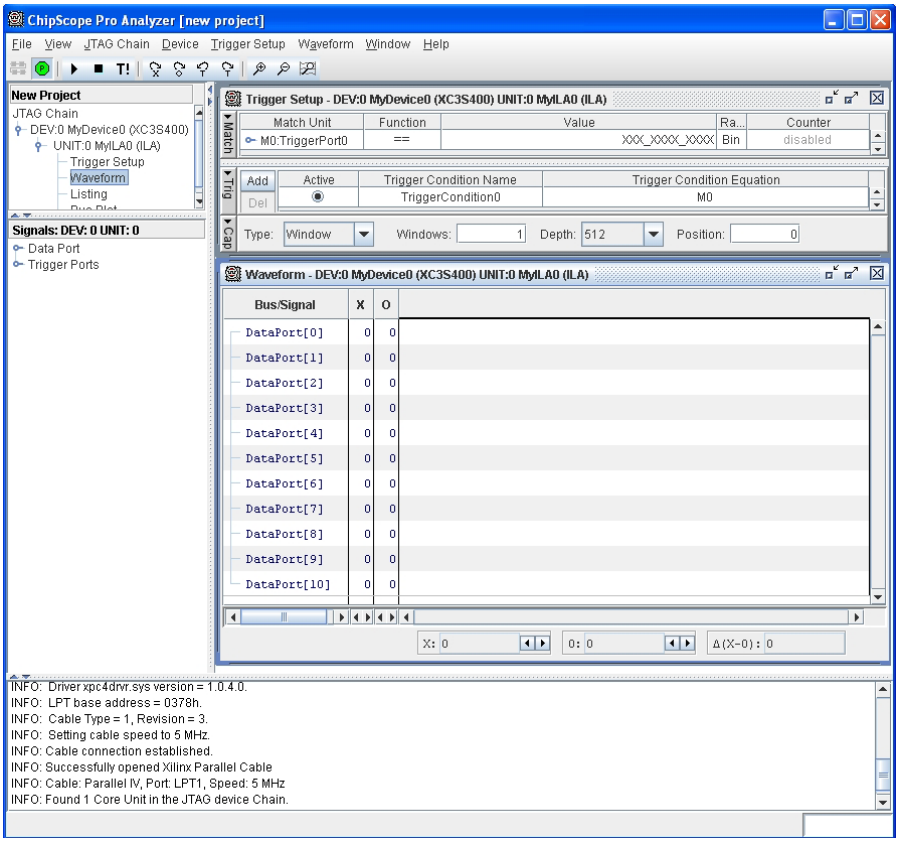

Picture 14: „Windows Firewall Alert“ message

A new menu window appears, describing the detected device:




Picture 15: Device detection

Quit with **OK;** the trigger window, the data window and the console window appear in the main window:



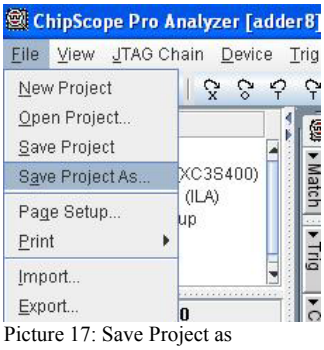

Picture 16: „ChipScope „ main window after device detection took place

The last line in the console window is important:

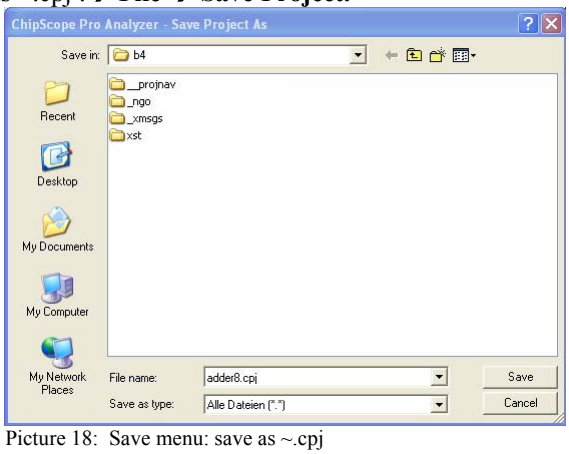
**INFO Found 1 Core Unit found in the JTAG device Chain**

If it reads: Found 0 Core Unit found ... then „ChipScope“ and „Impact“ must be closed and programmed again.

In order to see your own signal names in the Waveform List, you can overwrite these (DataPort(0), etc.) and save them as ~.cpj :→ **File → Save Project.**



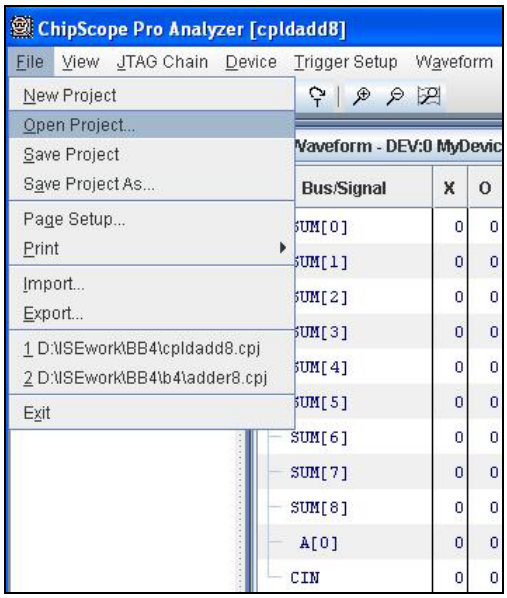

Picture 17: Save Project as




Picture 18: Save menu: save as ~.cpj

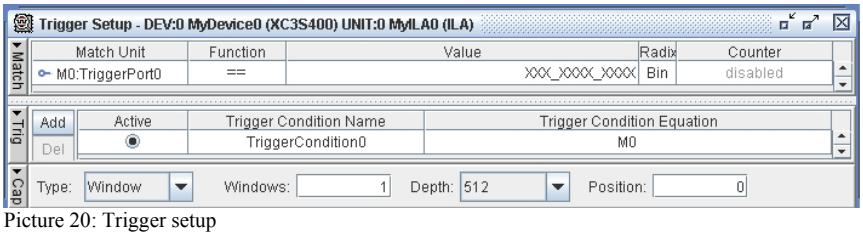
A complete project file can be loaded with → **File → Open Project**:

As a result you can see the changed signal names:



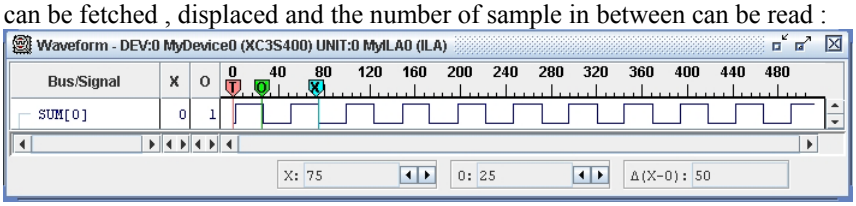

Picture 19: Open Project

In the trigger window, you can indicate a trigger condition. If everything is marked with ‚X‘, click on  to start the data intake.




Picture 20: Trigger setup

The X-cursor and an O-cursor in the upper left corner below the trigger-cursor can be fetched , displaced and the number of sample in between can be read :



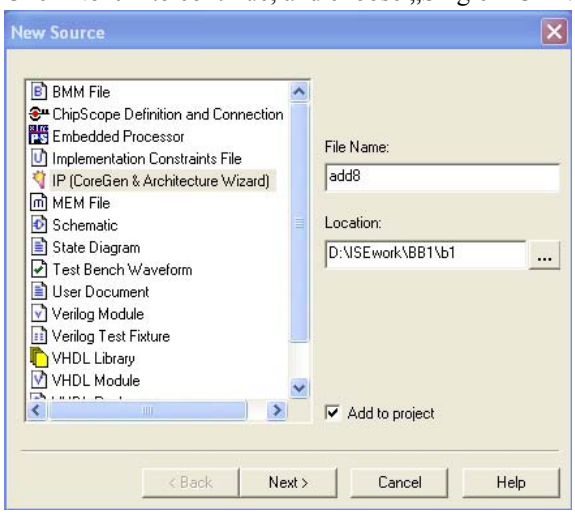

Picture 21: X-Cursor and O-Cursor

You can make use of the buttons „Go to X-Cursor“, „Go to O-Cursor“ and „Zoom in“ and „Zoom out“:

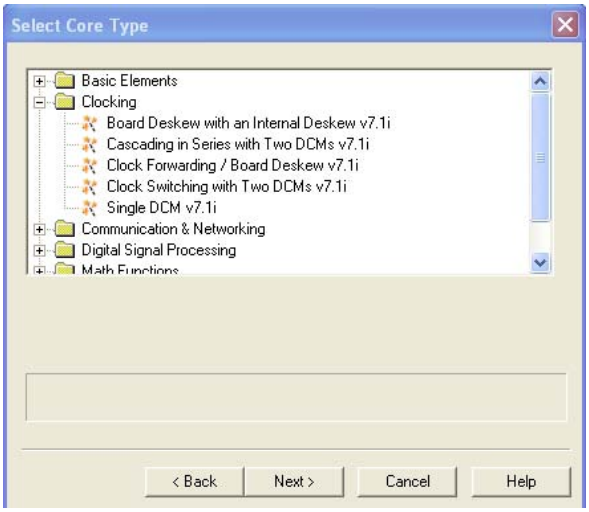

# Appendix A:

## Usage of the frequency multiplication through the DCM-module

To start the „DCM-Wizard“ (building menu) you must first select the menu item IP (CoreGen & Architecture Wizard) with → **Project** → **New Source**; give a file name and activate the „Add to Project“ option.
Click **Next >** to continue, and choose „Single DCM v7.1i“.

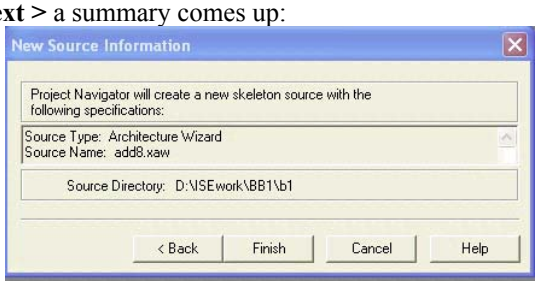



Picture 1_A: New Source



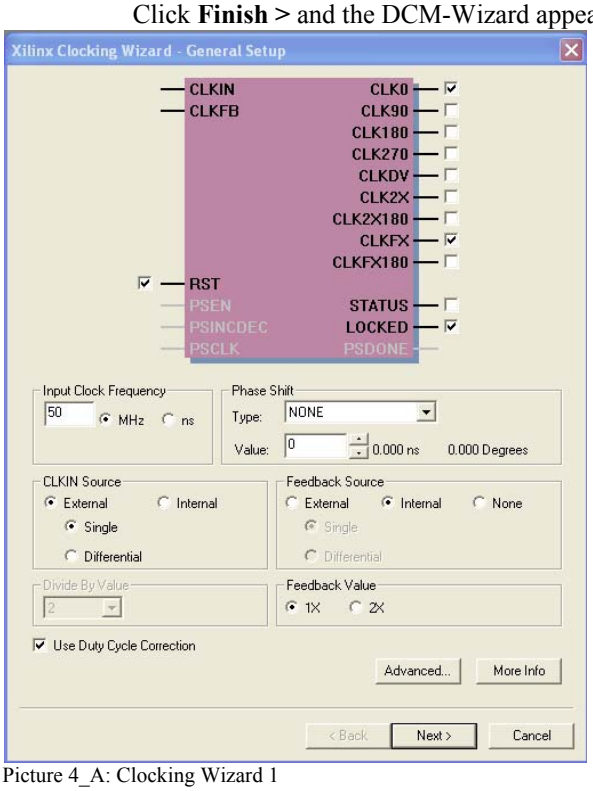

Picture 2_A: Select Core Type

With **Next >** a summary comes up:

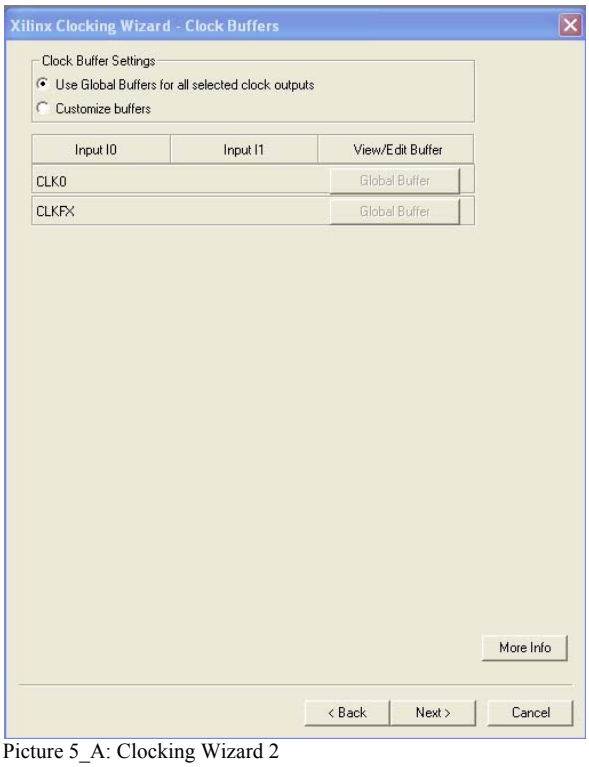



Picture 3_A: New Source Information

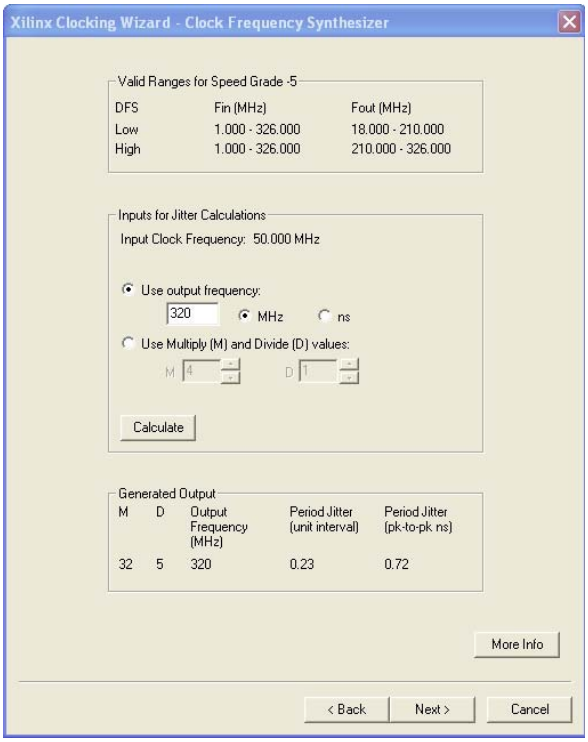
Click **Finish >** and the DCM-Wizard appears:



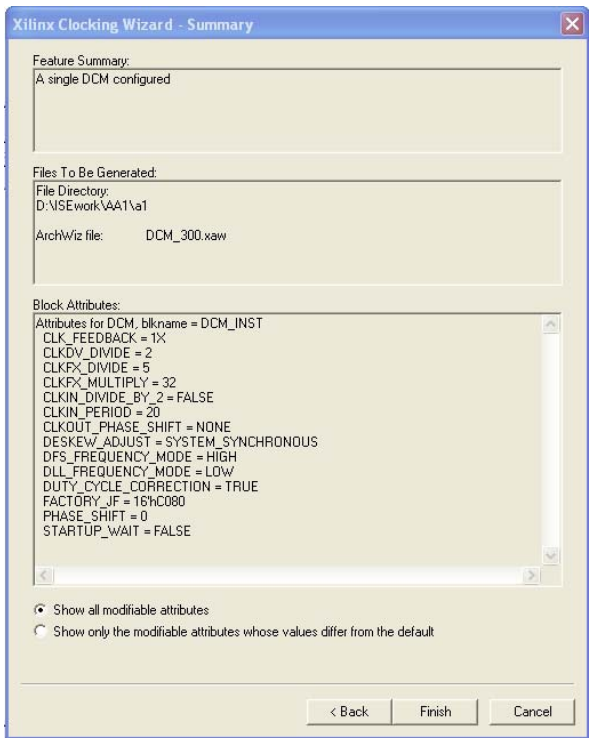

Picture 4_A: Clocking Wizard 1




Picture 5_A: Clocking Wizard 2

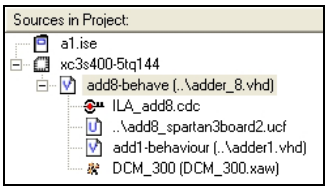



Picture 6_A: Clocking Wizard 3



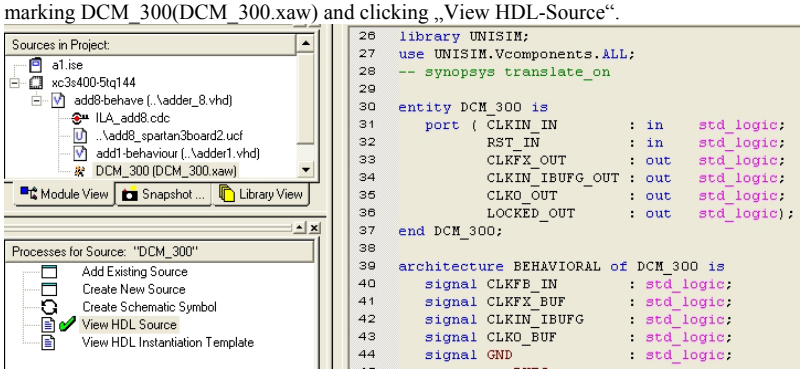

Picture 7_A: Clocking Wizard 4

After this, the read-file ~.xaw is attached to the project:




Picture 8_A: ~.xaw file in the project

Corresponding VHDL source code is also generated in the working folder; this can be viewed in the „Processes for Source“ window by marking DCM_300(DCM_300.xaw) and clicking „View HDL-Source“.




Picture 9_A: Viewing DCM_300.vhd

Now this DCM-component must be instantiated:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
--DCM
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;


entity ADD8 is

     port(   CIN ,SYSCLK    : in bit;
             A,B      : in bit_vector ( 7 downto 0) ;
             N_SYSCLK : out bit;
            SUM       : out bit_vector ( 8 downto 0) );
end ADD8;
```

```
architecture BEHAVE of ADD8 is

--Function: To_StdLogic
function To_stdlogic ( b: bit) return std_logic is
   variable result: std_logic;
   begin
   if b='0' then
      RESULT := '0';
   else
      RESULT := '1';
   end if;
   return RESULT;
end To_stdlogic;


--Komponentendeklaration- - Portliste
component ADD1
   port(X,Y,CIN : in  bit;
        SI,COUT : out bit );
end component;

--Komponente DCM
component DCM_300
   port ( CLKIN_IN        : in    std_logic;
          RST_IN          : in    std_logic;
          CLKFX_OUT       : out   std_logic;
          CLKIN_IBUFG_OUT : out   std_logic;
          CLKO_OUT        : out   std_logic;
          LOCKED_OUT      : out   std_logic);
end component;



signal RC :  bit_vector( 6 downto 0);
signal GND, CLKFX : std_logic;


begin
N_SYSCLK <= not (to_bit(CLKFX));
GND <= '0';
--Komponenteninstanziierung

DCM_300_INST : DCM_300
    port map(CLKIN_IN => to_stdlogic (SYSCLK),
             RST_IN =>  GND,
             CLKFX_OUT => CLKFX,
             CLKIN_IBUFG_OUT => open,
             CLKO_OUT      => open,
             LOCKED_OUT     => open );
```

The DCM_300.vhd instantiates the DCM-module on the Spartan-board.
The divisor/multiplication factor sets the quotient from CLKFX_MULTIPLY
and CLKFX_DIVIDE.
The maximum factor is 32/1, and as seen in the Wizard, the maximum frequency is 326 MHz.
With a division factor of 8/1, a board frequency of 50 MHz can be increased to 392 MHz on most boards.

```
   DCM_INST : DCM
   generic map( CLK_FEEDBACK => "1X",
            CLKDV_DIVIDE => 2.000000,
            CLKFX_DIVIDE => 1,
            CLKFX_MULTIPLY => 8,
            CLKIN_DIVIDE_BY_2 => FALSE,
            CLKIN_PERIOD => 20.000000,
            CLKOUT_PHASE_SHIFT => "NONE",
            DESKEW_ADJUST => "SYSTEM_SYNCHRONOUS",
            DFS_FREQUENCY_MODE => "HIGH",
            DLL_FREQUENCY_MODE => "LOW",
            DUTY_CYCLE_CORRECTION => TRUE,
            FACTORY_JF => x"C080",
            PHASE_SHIFT => 0,
            STARTUP_WAIT => FALSE)
      port map (CLKFB=>CLKFB_IN,
                CLKIN=>CLKIN_IBUFG,
                DSSEN=>GND,
                PSCLK=>GND,
                PSEN=>GND,
                PSINCDEC=>GND,
                RST=>RST_IN,
                CLKDV=>open,
                CLKFX=>CLKFX_BUF,
                CLKFX180=>open,
                CLK0=>CLK0_BUF,
                CLK2X=>open,
                CLK2X180=>open,
                CLK90=>open,
                CLK180=>open,
                CLK270=>open,
                LOCKED=>LOCKED_OUT,
                PSDONE=>open,
                STATUS=>open);

end BEHAVIORAL;
```

After synthesis, the DCM-module is integrated as follows: