klee@unibwm.de 14.11.2007

MPROZ3 – A 16 bit minimal processor

MPROZ3 is a modification of the 16 bit processor MPROZ which uses the internal RAM of the Spartan-3 FPGA instead of an external RAM/ROM.

The schematics (ISI 9.2i) and the assembler can be found in mproz3.zip

1. Processor architectur

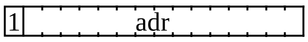
1.1 Register

To minimize the hardware, there is only a 15 bit program counter (PC) and a 1 bit flag (F). No other user accessible registers exist. After a reset PC and F are initialized to 0.

1.2 Instruction set

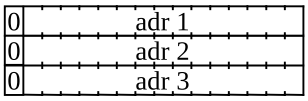
MPROZ supports five instructions:

br adr



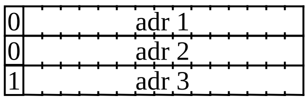
Jump to adr if F=0
Clear F.

add adr1,adr2,adr3 [adr3] = [adr1] + [adr2]



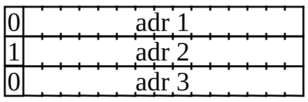
Add the contents of memory location adr1 and the contents of memory location adr2 and store the result in memory location adr3. Store the carry of the addition in F.

nadd adr1,adr2,adr3 [adr3] = not([adr1] + [adr2])



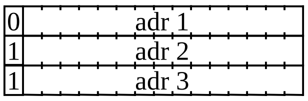
Add the contents of memory location adr1 and the contents of memory location adr2, invert the result and store it in memory location adr3. F=1 if result =0 , else F=0.

or adr1,adr2,adr3 [adr3] = (not[adr1]) or [adr2] = (not[adr2]) nand [adr1]
nand adr2,adr1,adr3



Calculate the OR function of the inverted contents of memory location adr1 and the contents of memory location adr2 and store the result in memory location adr3. F=1 if result =0 , else F=0.

```
and     adr2,adr1,adr3      [adr3] = (not[adr2]) ] and [adr1 = (not[adr1]) nor [adr2]
nor     adr1,adr2,adr3
```



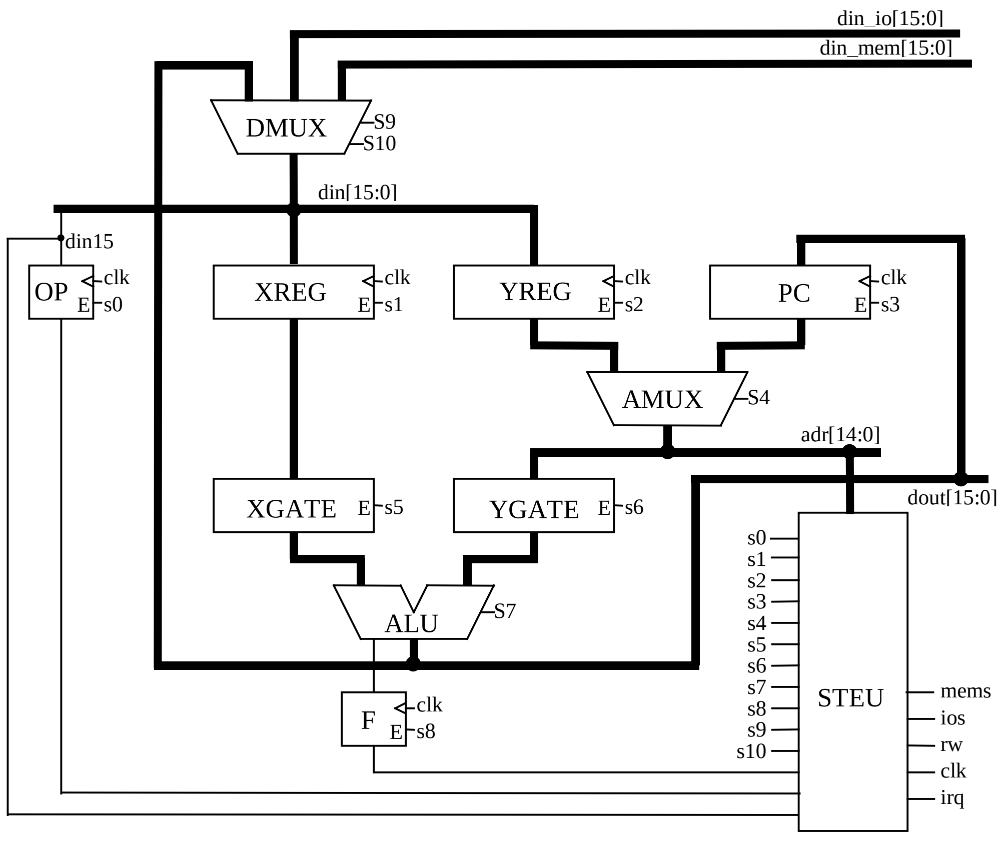
Calculate the AND function of the inverted contents of memory location adr2 and the contents of memory location adr1 and store the result in memory location adr3. F=1 if result =0 , else F=0.

1.3 Interrupt

Because there is no instruction to set or reset the interrupt enable flag (ie), this is done by accessing memory location 3. A read from 3 enables interrupts (ie=1) and a write to 3 disables interrupts (ie=0). An external interrupt request (IRQ~) must be asserted until the interrupt is serviced by MPROZ3. The interrupt is serviced by MPROZ3 when the interrupt enable flag is set and a branch instruction with F=0 is executed. Thereby it is not necessary to save the program counter (PC) and the flag (F), but it is sufficient to save the interrupted branch instruction. This instruction is saved at the memory location which address is stored in memory location 1 (this normally should be address 3, because then the ie flag is automatically cleared, when the instruction is saved). The execution of the interrupt program starts at address 2. The return from interrupt is done by a branch to 3, which also automatically sets the ie flag.

2. Implementation

2.1 CPU



OP:	1 bit register				
F:	1 bit register				
PC:	15 bit register				
XREG:	16 bit register				
YREG:	16 bit register				
AMUX:	IF (s4=0)	THEN	out=in1	ELSE	out=in2
XGATE:	IF (s5=0)	THEN	out=$0000	ELSE	out=in
YGATE:	IF (s6=0)	THEN	out=$0001	ELSE	out=in
ALU:	IF (s7=0)	THEN	out=in1 ADD in2 ,		F=carry
		ELSE	out=~in1 OR in2 ,		F=zero
DMUX:	IF (s10=1)	THEN	out=in1		
		ELSE	IF (s9=1) out=in2		
			ELSE out=in3		
STEU:	generates control signals s0-s10, mems, ios and rw				

state	action
q0	PC→adr, PC+1→PC, din→XREG,YREG
q1	YREG→adr, din→XREG
q2	PC→adr, PC+1→PC, din→ YREG, din15→OP
q3	YREG→adr, din→YREG
q4	XREG [ADD|OR1n] YREG → XREG, [Carry|Zero] → F
q5	PC→adr, PC+1→PC, din→ YREG
q6	YREG→adr, XREG [+,OR1n] 0→dout, [F|Zero] → F
q7	[XREG | 1]+0→PC falls F=0, 0→F

int = irq + ~ie + F

Zustand	s10	s9	s8	s7	s6	s5	s4	s3	s2	s1	s0	mems	ios	r/w
q0	0	a14	0	0	1	0	1	1	1	1	-	a14	~a14	1
q1	0	a14	-	-	-	-	0	0	-	1	-	a14	~a14	1
q2	0	a14	-	0	1	0	1	1	1	0	1	a14	~a14	1
q3	0	a14	-	-	-	-	0	0	1	0	0	a14	~a14	1
q4	1	-	1	OP	1	1	0	0	-	1	-	1	1	1
q5	0	a14	0	0	1	0	1	1	1	0	1	a14	~a14	1
q6	0	a14	OP	OP	0	1	0	0	-	-	-	a14	~a14	0
q7	-	-	1	0	0	int	-	~F	-	0	-	1	1	1

s0 = ~q3

s1 = q0 + q1 + q4

s2 = 1

s3 = q0+q2+q5+q7·(~F)

s4 = q0 + q2 + q5

s5 = q4 + q6 + q7 · int

s6 = ~(q6 + q7)

s7 = (q4+q6) OP

s8 = q4+q6 OP+q7

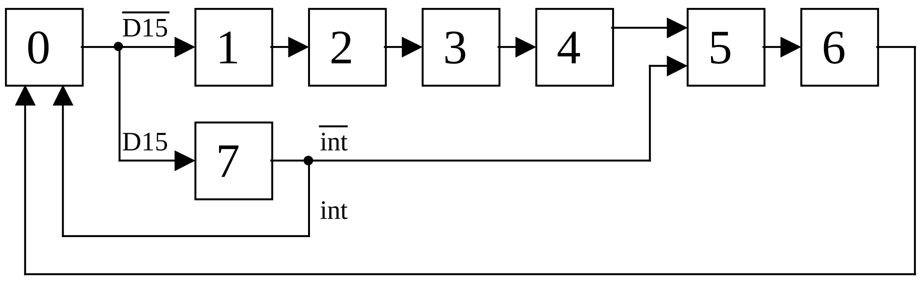
s9 = a14

s10 = q4

mems = q4+q7+a14

ios == q4+q7+~a14

r/w = ~q6



$d0 = q6 + q7 \cdot int$

$d1 = q0 \cdot \overline{D15}$

$d2 = q1$

$d3 = q2$

$d4 = q3$

$d5 = q4 + q7 \cdot \overline{int}$

$d6 = q5$

$d7 = q0 \cdot D15$

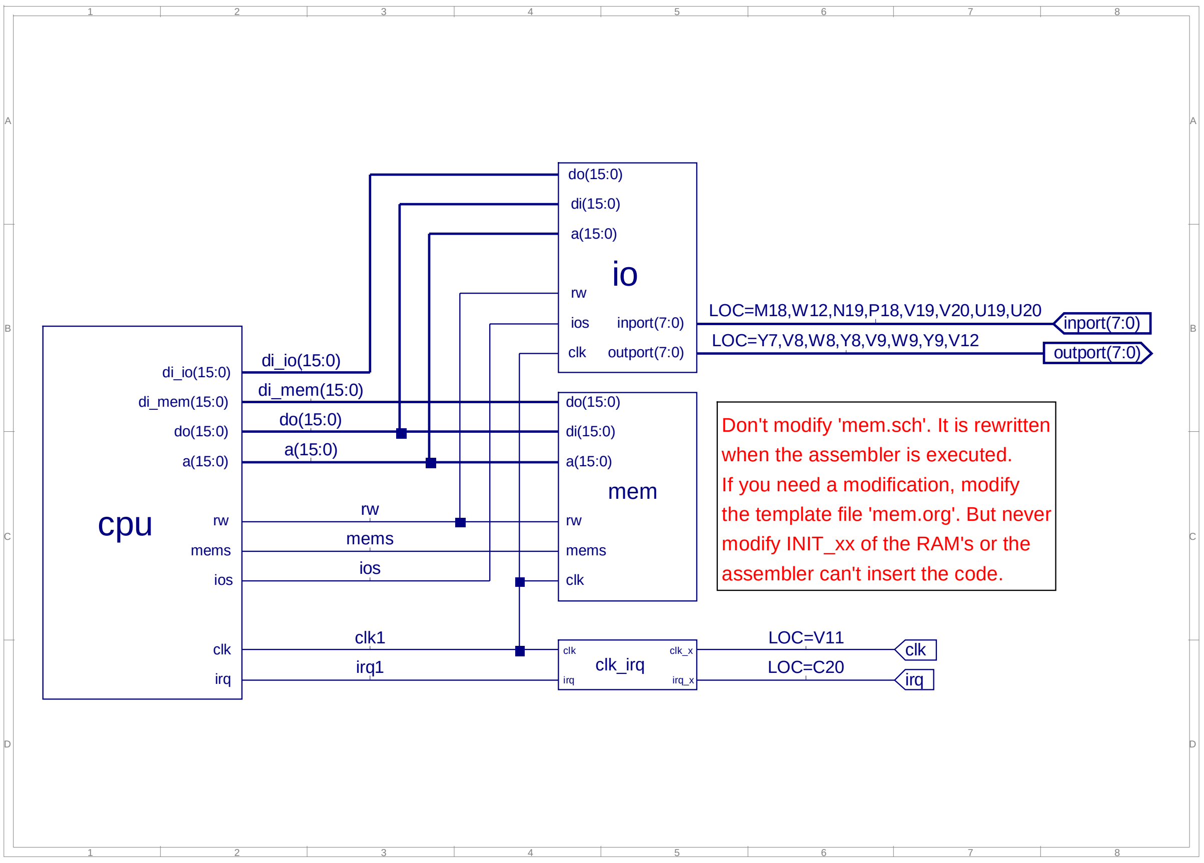
2.2 IO

MPROZ3 uses memory mapped IO.

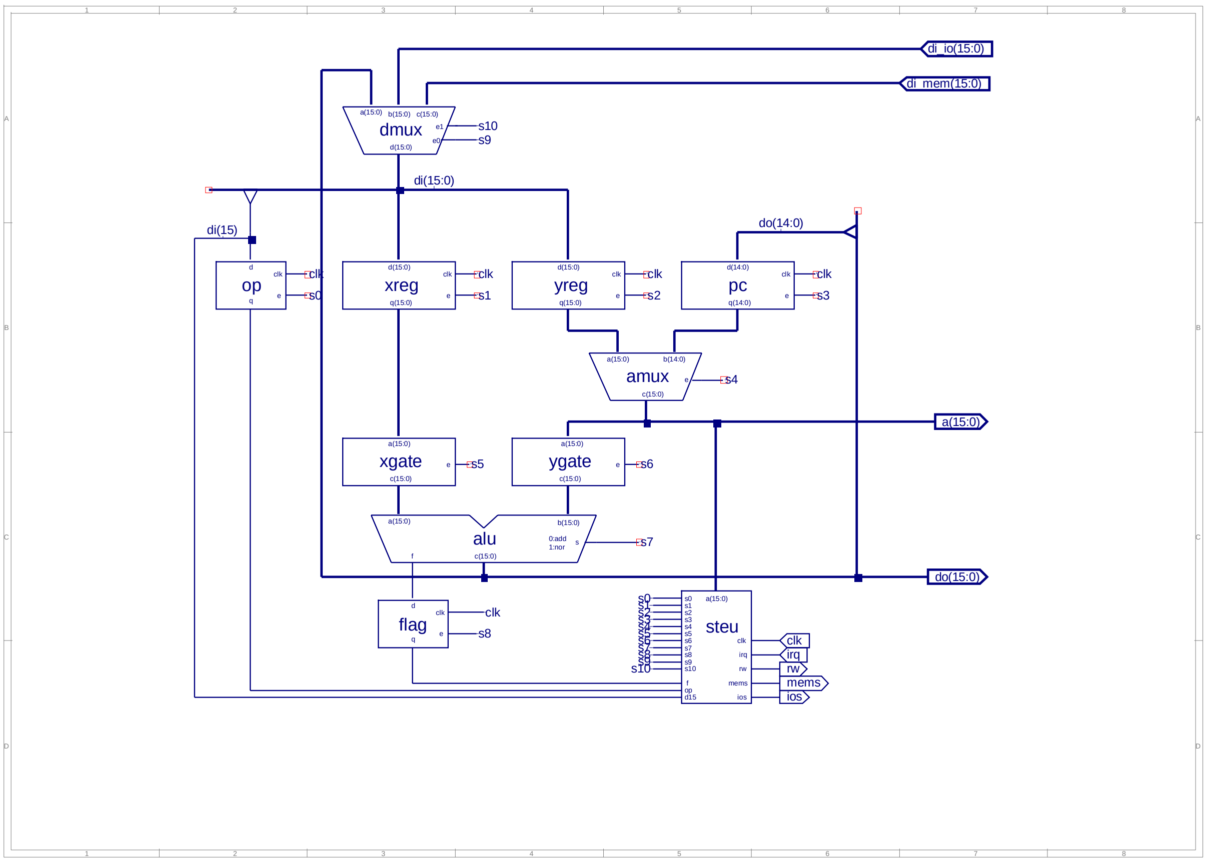
address		
$0000 - $3fff	RAM	32 kByte
$4000 - $7fff	IO	32 kByte

In this implementation the only IO is a 8 bit input and a 8 bit output port at address $7fff.

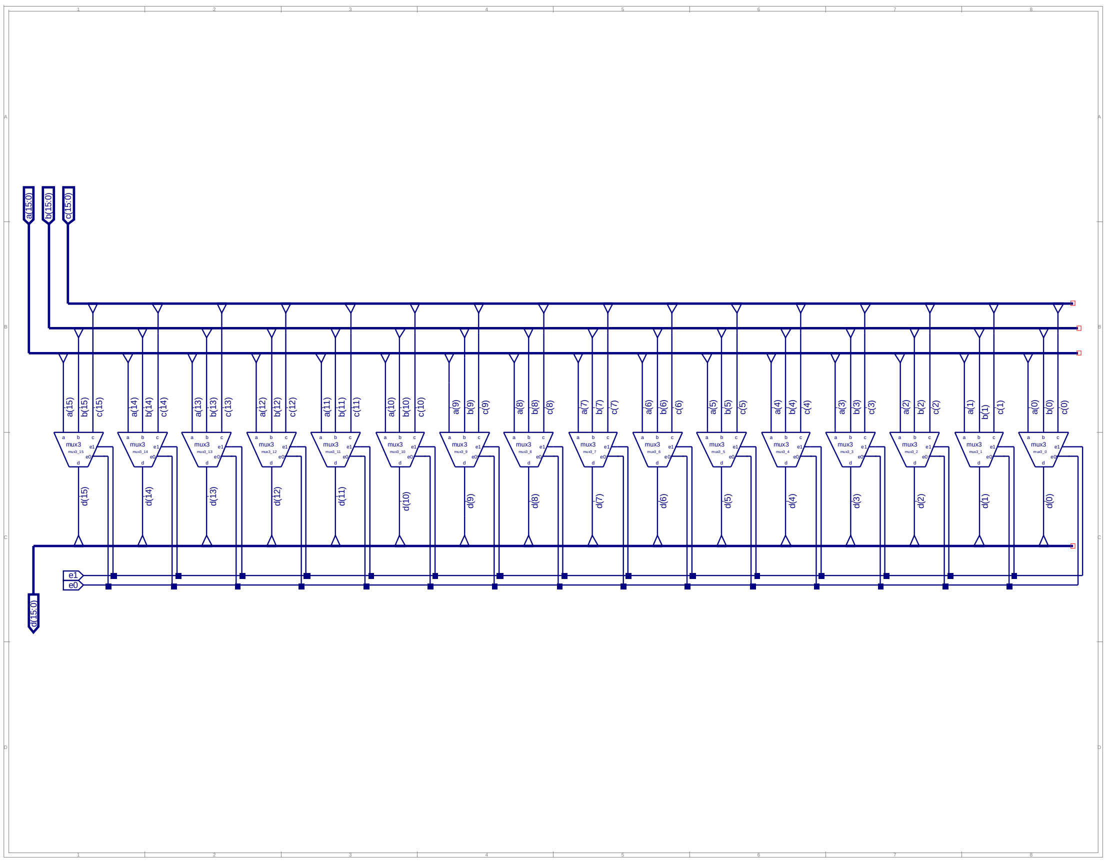
2.3 Schematics


io
do(15:0)
di(15:0)
a(15:0)
rw
ios
clk
inport(7:0)
outport(7:0)
LOC=M18,W12,N19,P18,V19,V20,U19,U20
LOC=Y7,V8,W8,Y8,V9,W9,Y9,V12
inport(7:0)
outport(7:0)
cpu
di_io(15:0)
di_mem(15:0)
do(15:0)
a(15:0)
rw
mems
ios
clk
irq
mem
do(15:0)
di(15:0)
a(15:0)
rw
mems
clk
Don't modify 'mem.sch'. It is rewritten
when the assembler is executed.
If you need a modification, modify
the template file 'mem.org'. But never
modify INIT_xx of the RAM's or the
assembler can't insert the code.
clk1
irq1
clk_irq
clk
irq
clk_x
irq_x
LOC=V11
LOC=C20
clk
irq

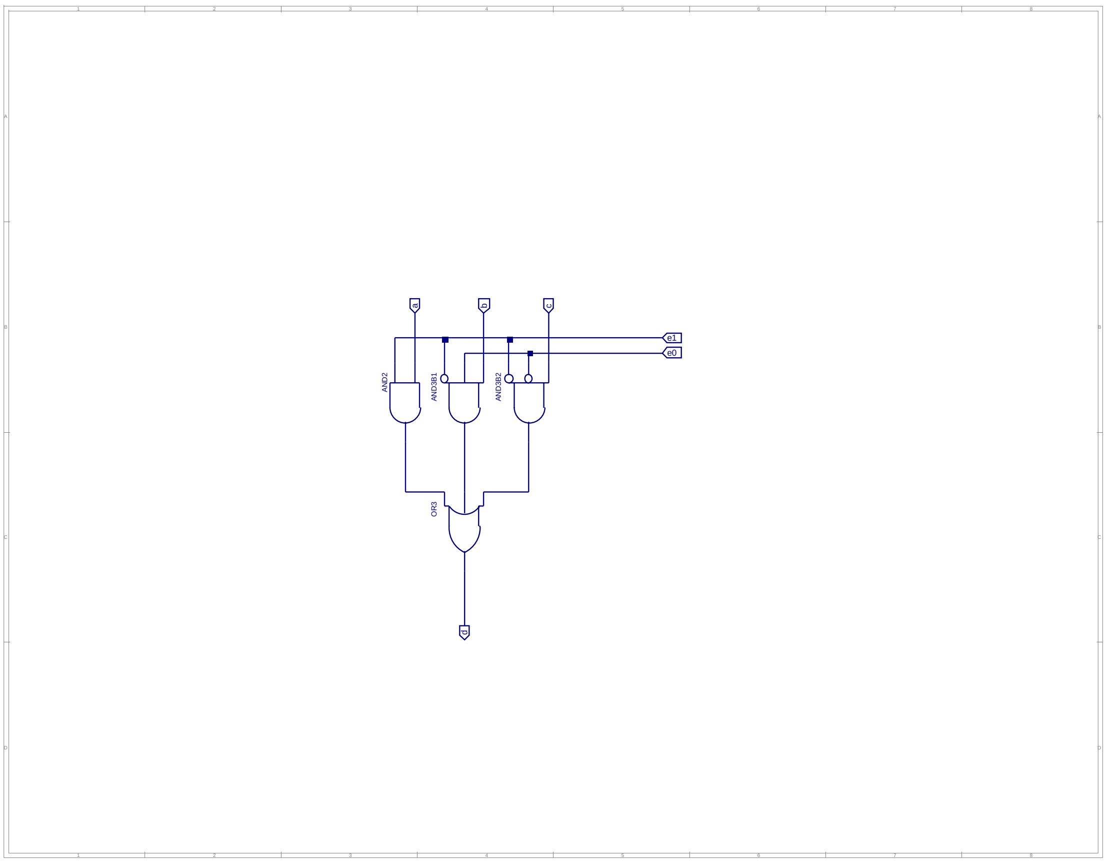
2.3.1 CPU



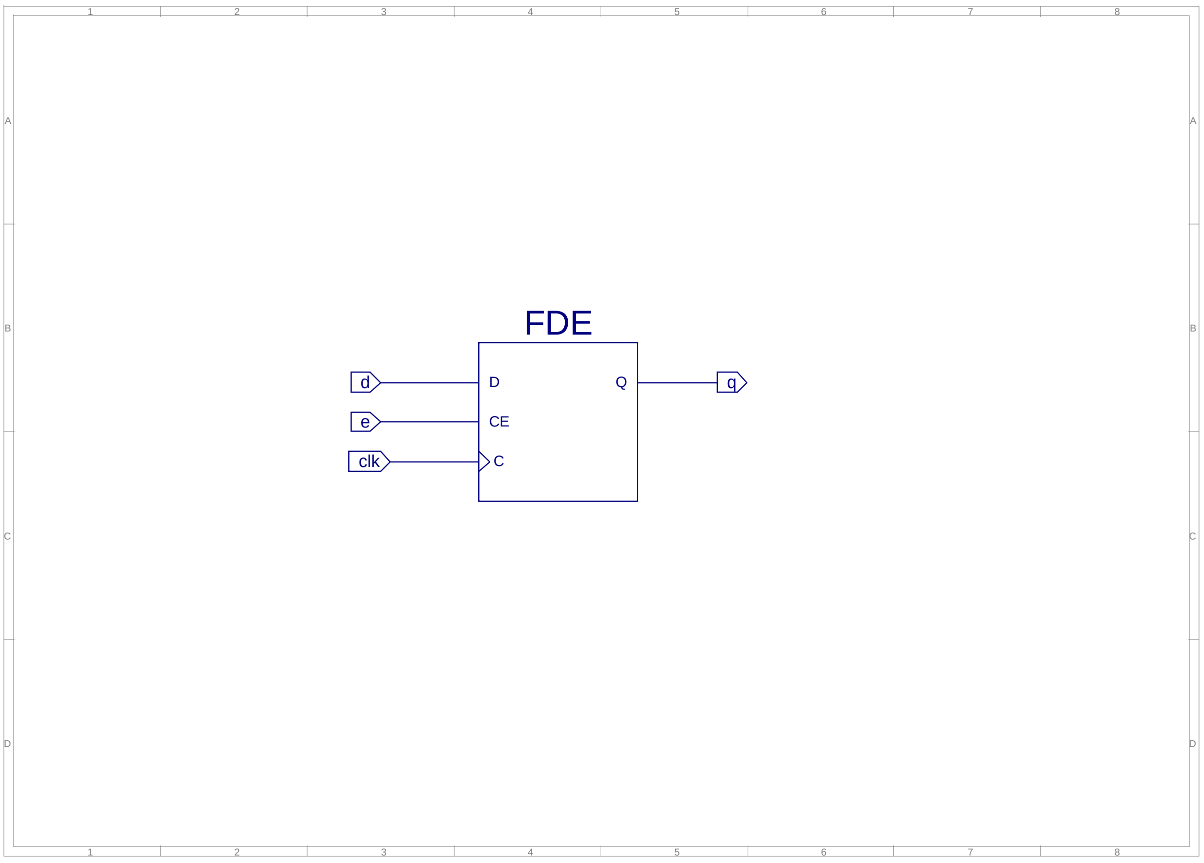
2.3.1.1 DMUX



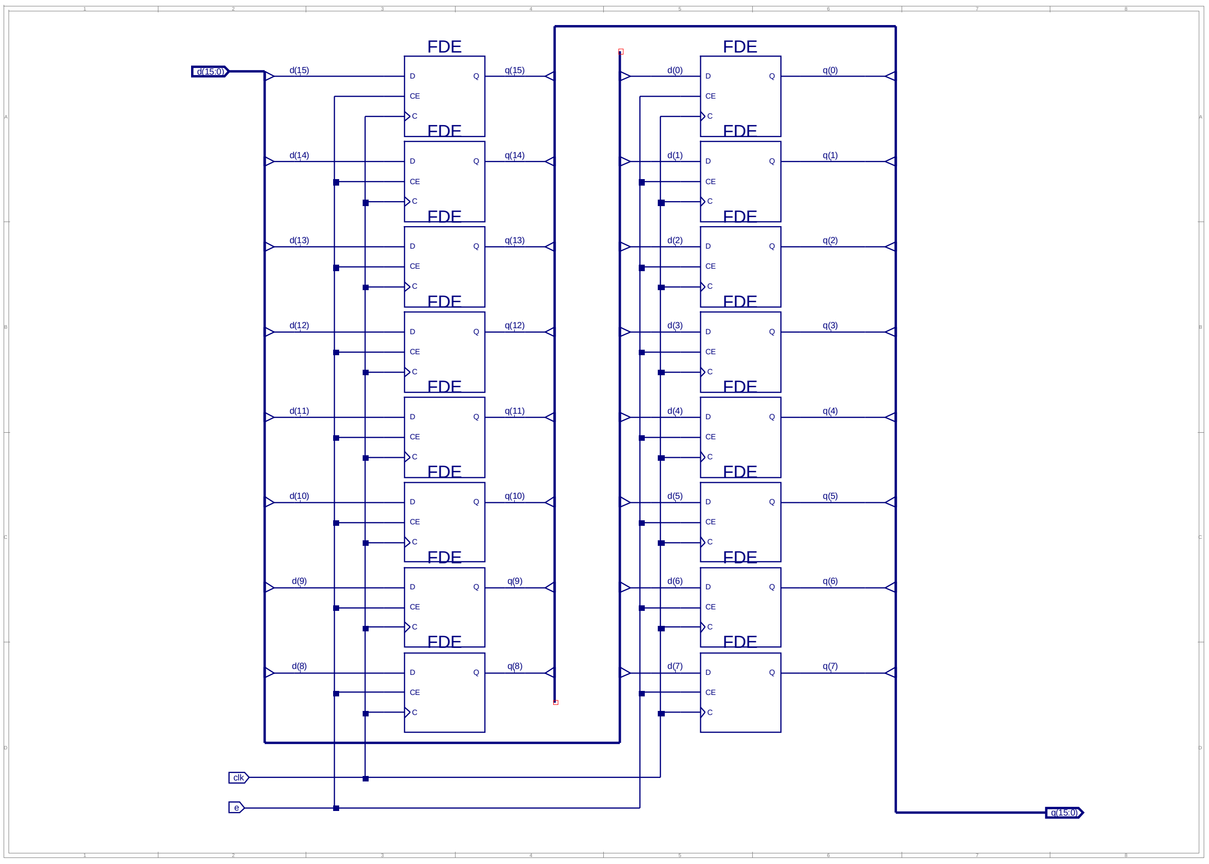
2.3.1.1.1 MUX3



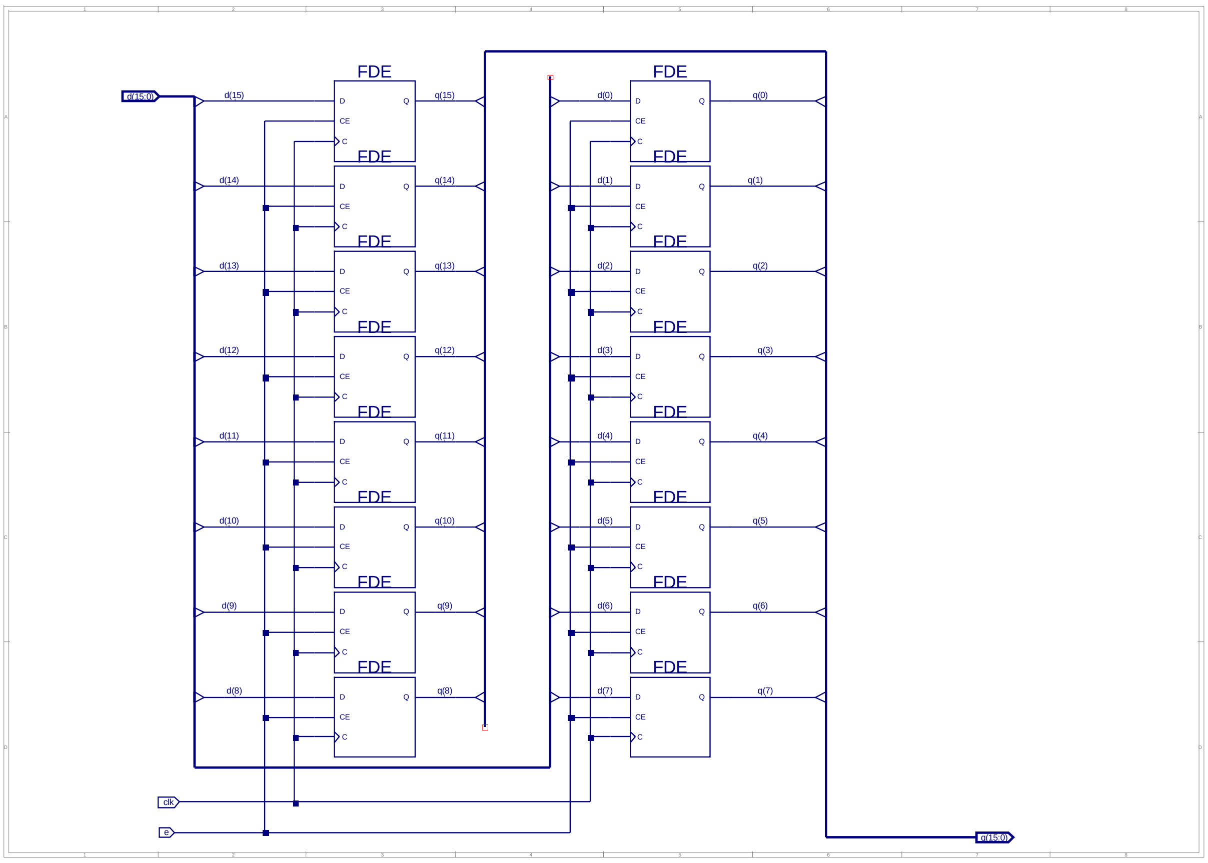
2.3.1.2 OP



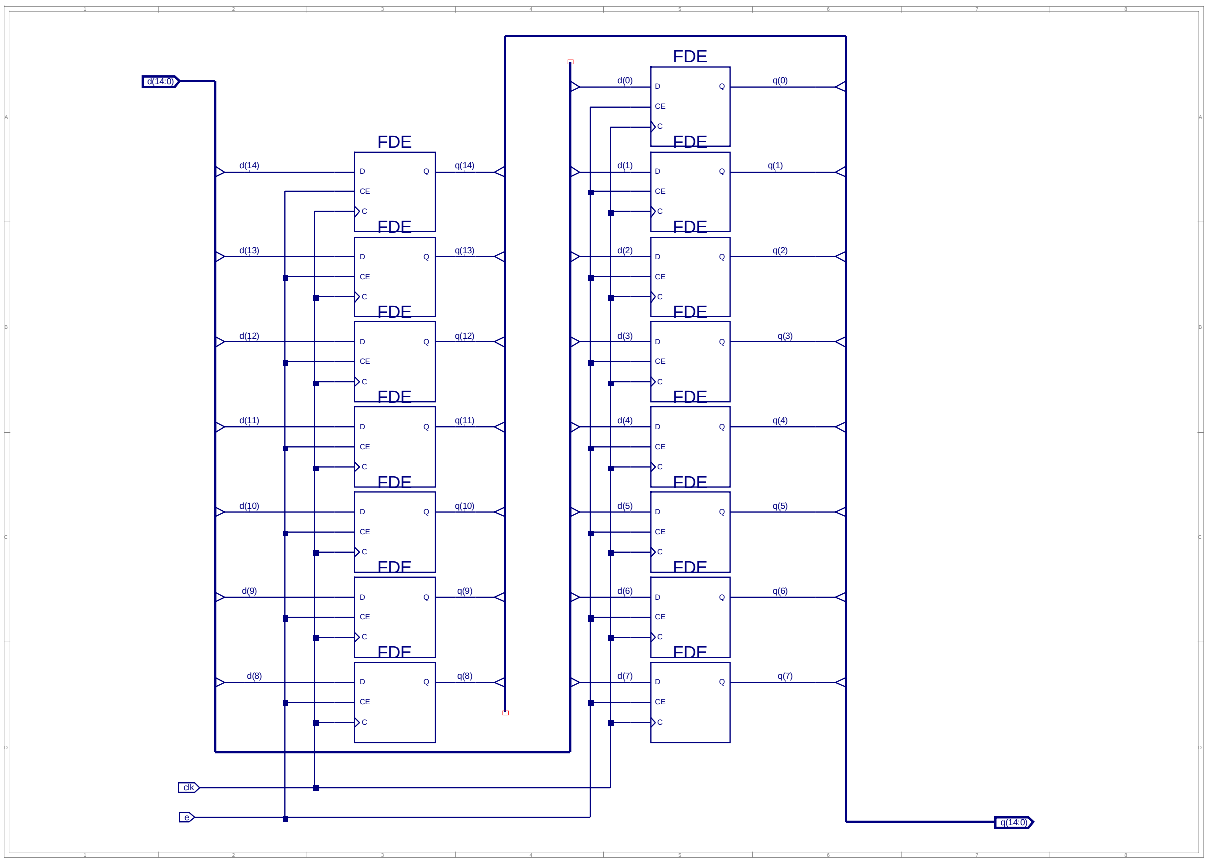
2.3.1.3 XREG



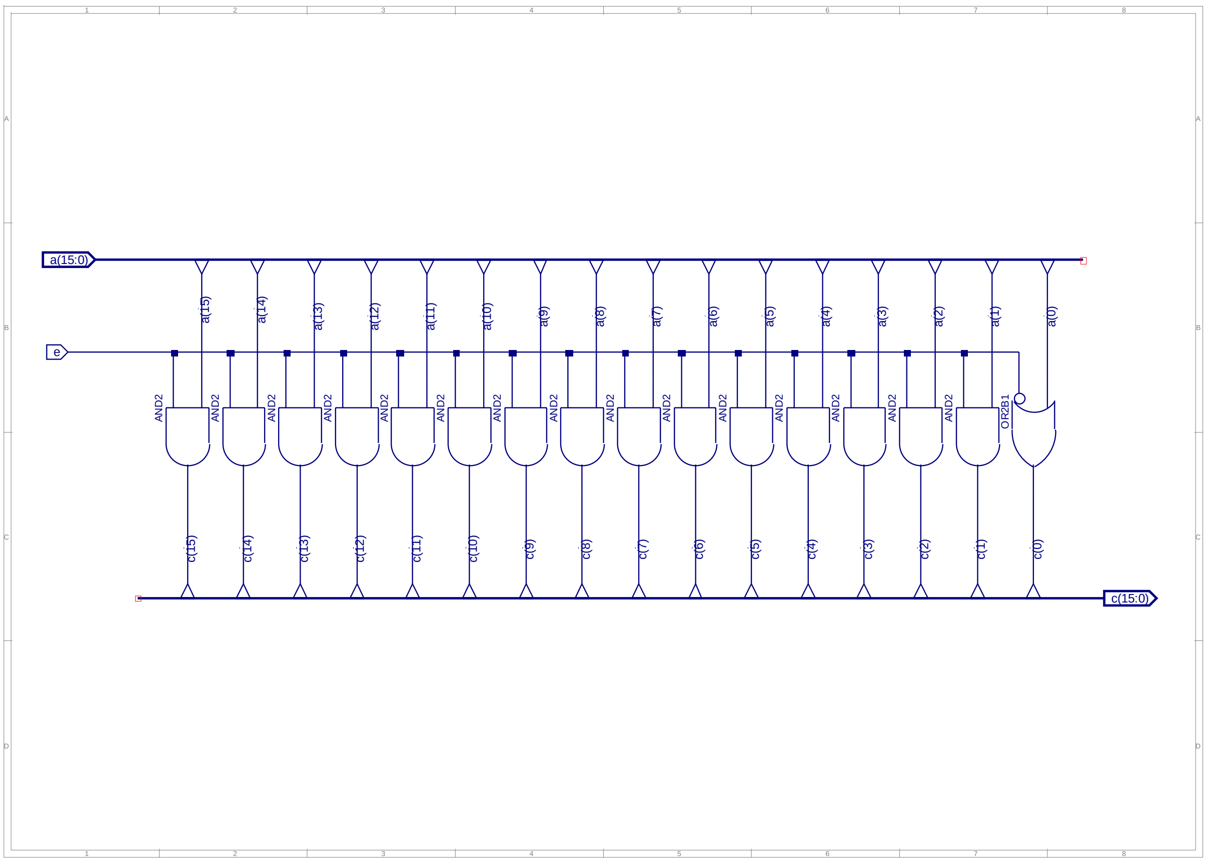
2.3.1.4 YREG



2.3.1.5 PC

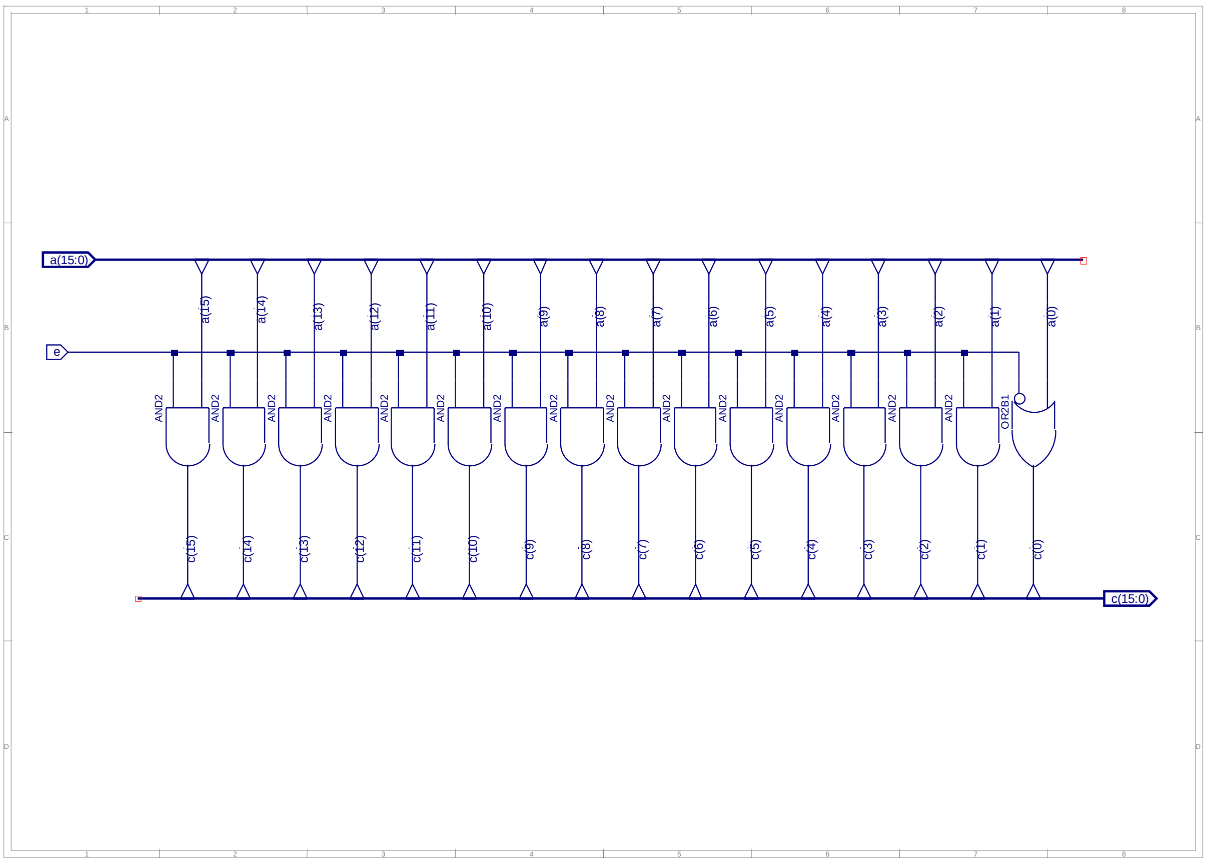


2.3.1.6 XGATE



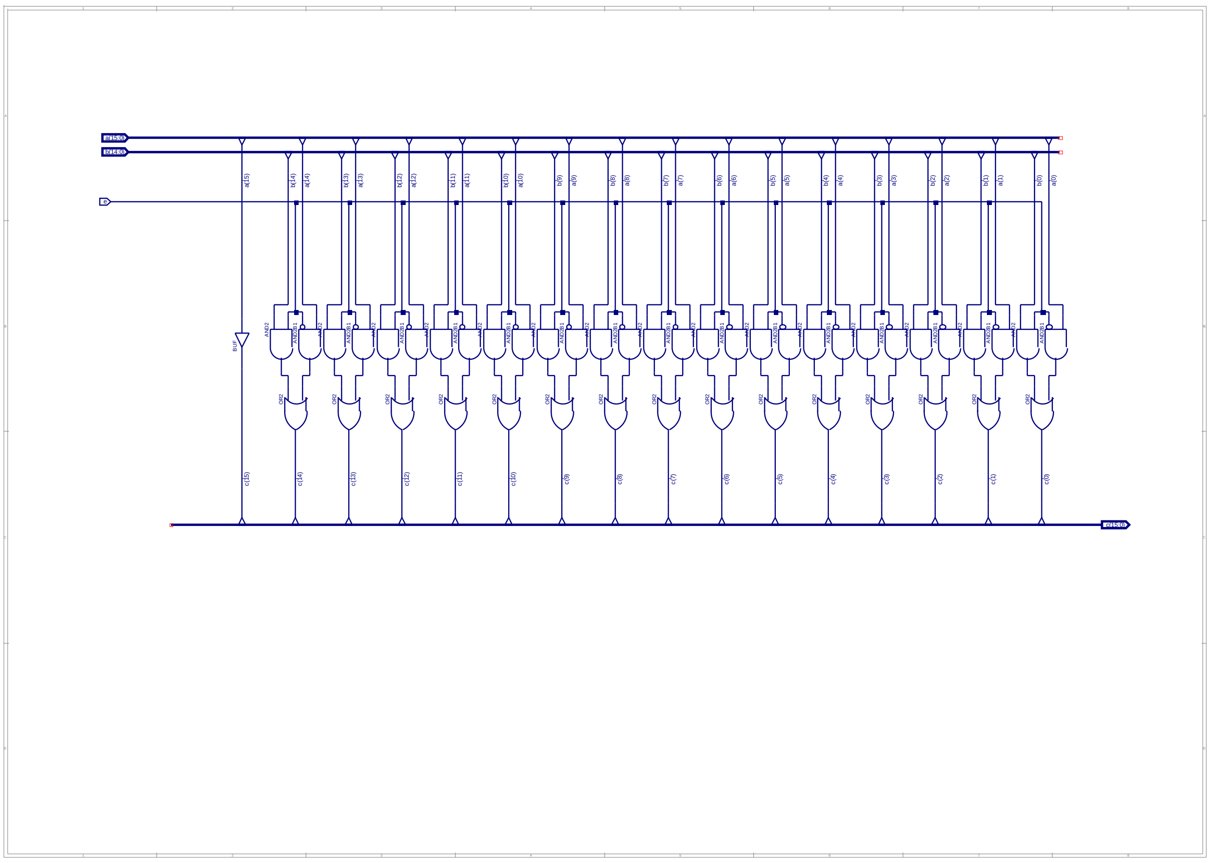
a(15:0)
e
a(15) a(14) a(13) a(12) a(11) a(10) a(9) a(8) a(7) a(6) a(5) a(4) a(3) a(2) a(1) a(0)
AND2
OR2B1
c(15) c(14) c(13) c(12) c(11) c(10) c(9) c(8) c(7) c(6) c(5) c(4) c(3) c(2) c(1) c(0)
c(15:0)

2.3.1.7 YGATE

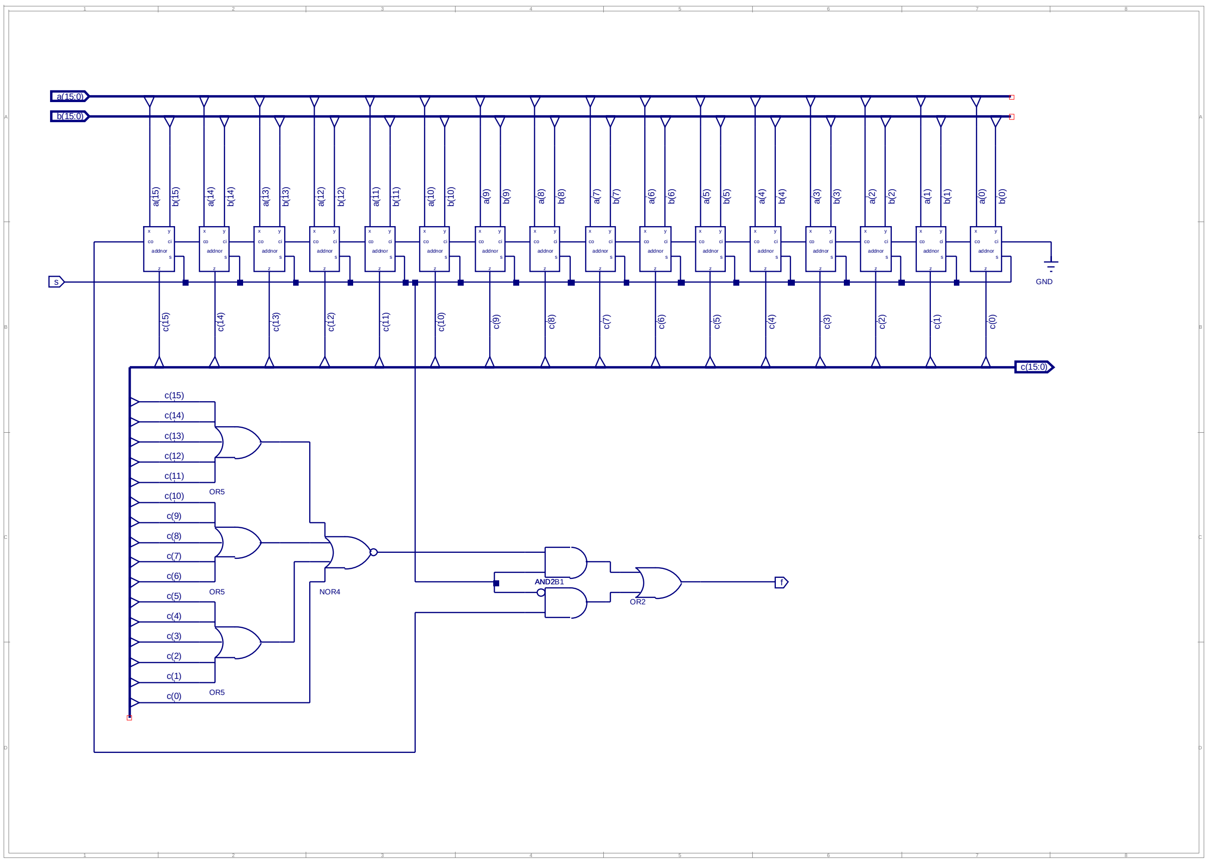


a(15:0)
e
a(15) a(14) a(13) a(12) a(11) a(10) a(9) a(8) a(7) a(6) a(5) a(4) a(3) a(2) a(1) a(0)
AND2
OR2B1
c(15) c(14) c(13) c(12) c(11) c(10) c(9) c(8) c(7) c(6) c(5) c(4) c(3) c(2) c(1) c(0)
c(15:0)

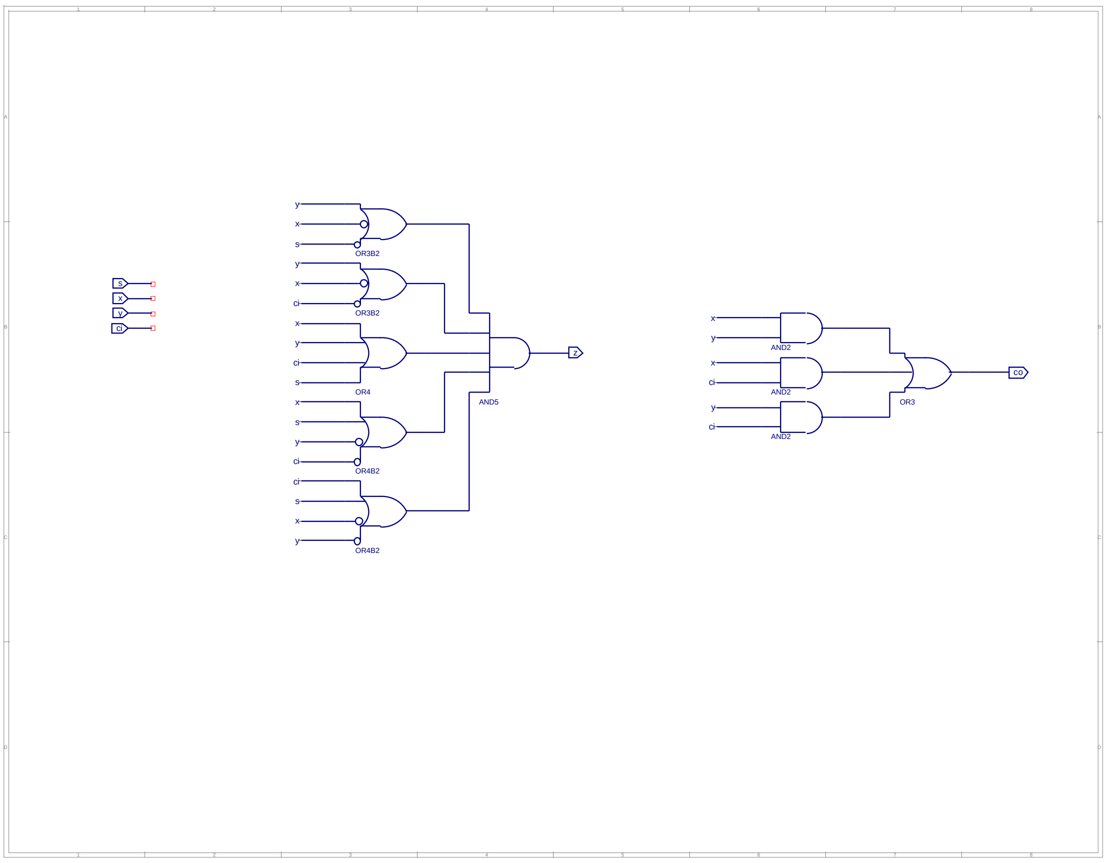
2.3.1.8 AMUX



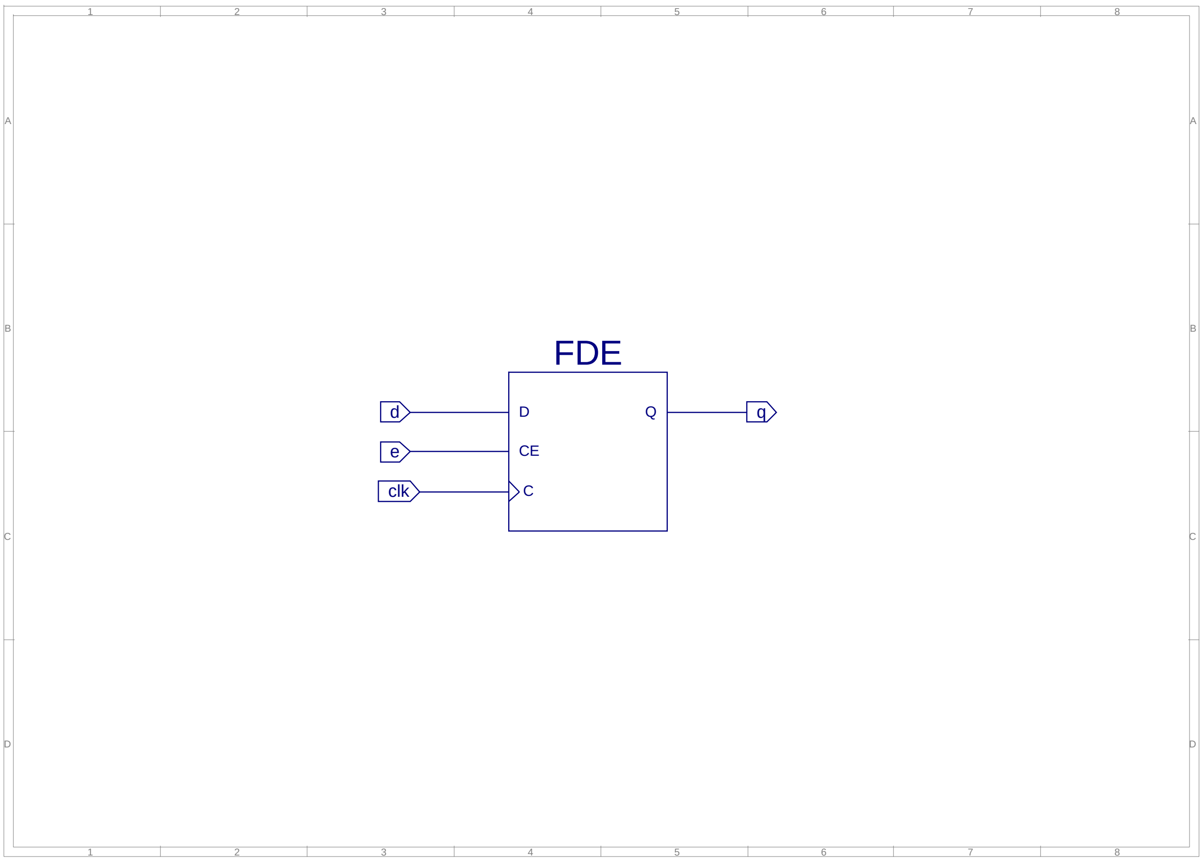
2.3.1.9 ALU



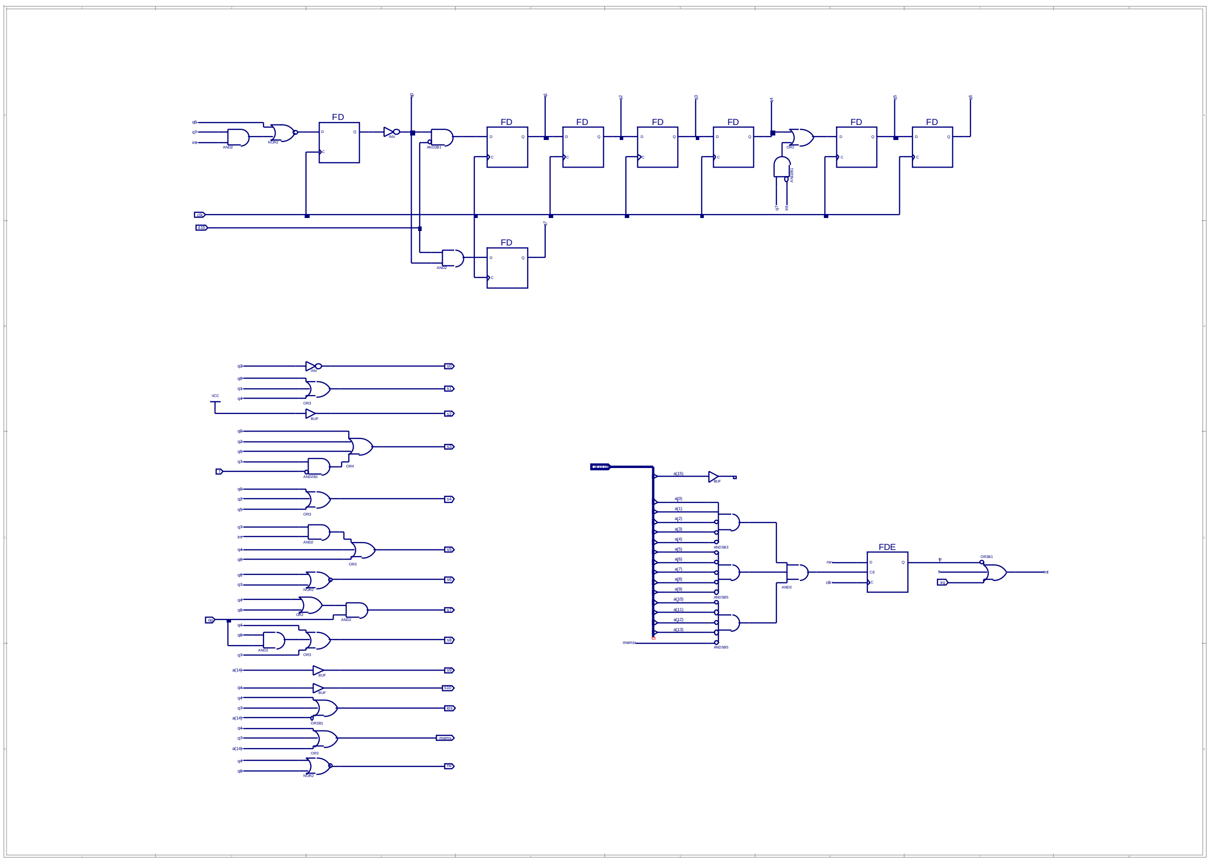
2.3.1.9.1 ADDNOR



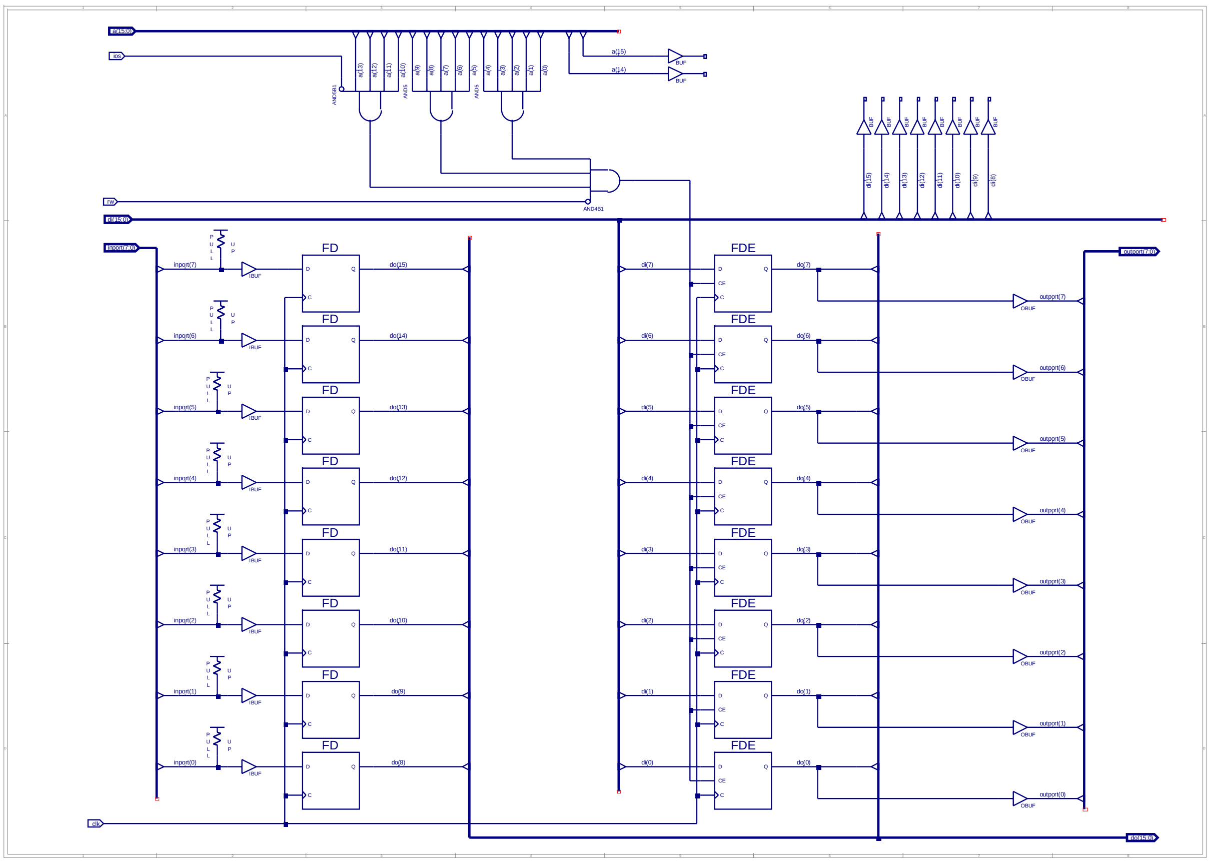
2.3.1.10 FLAG



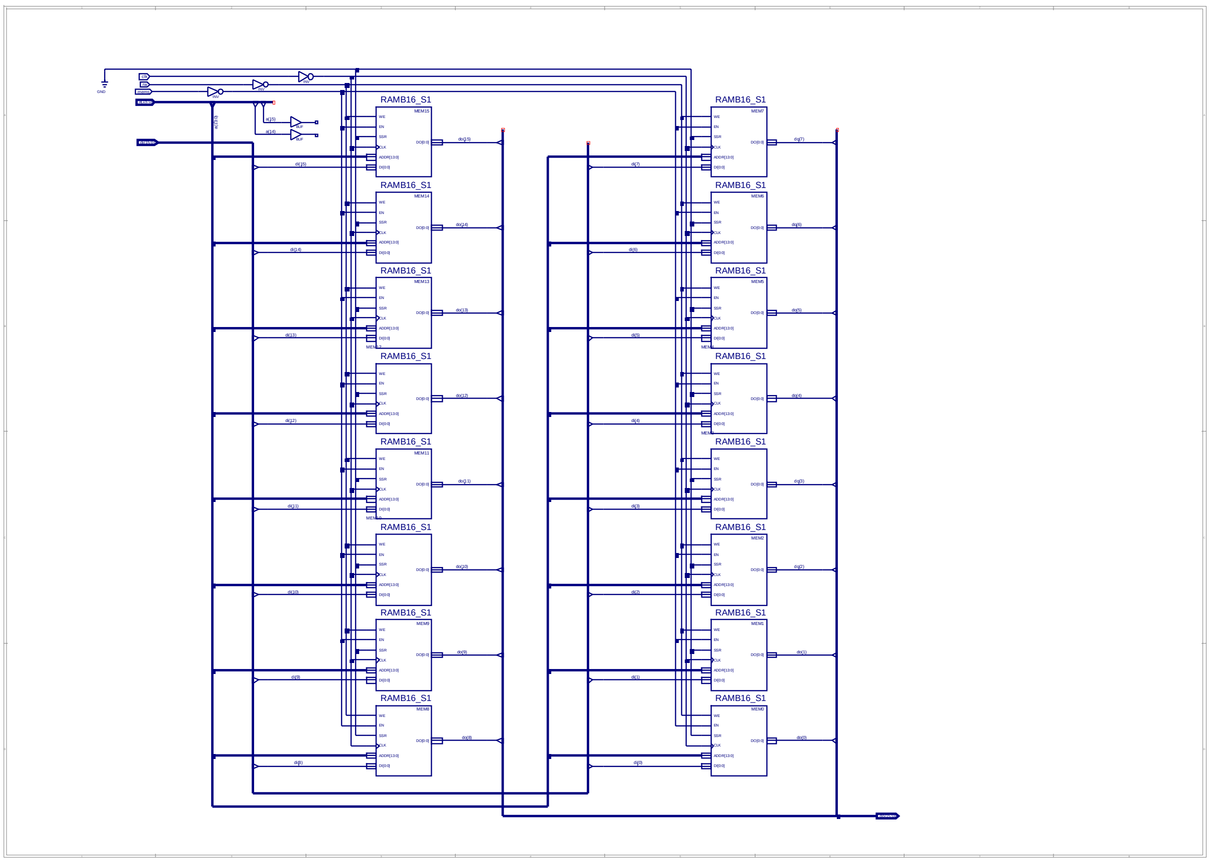
2.3.1.11 STEU



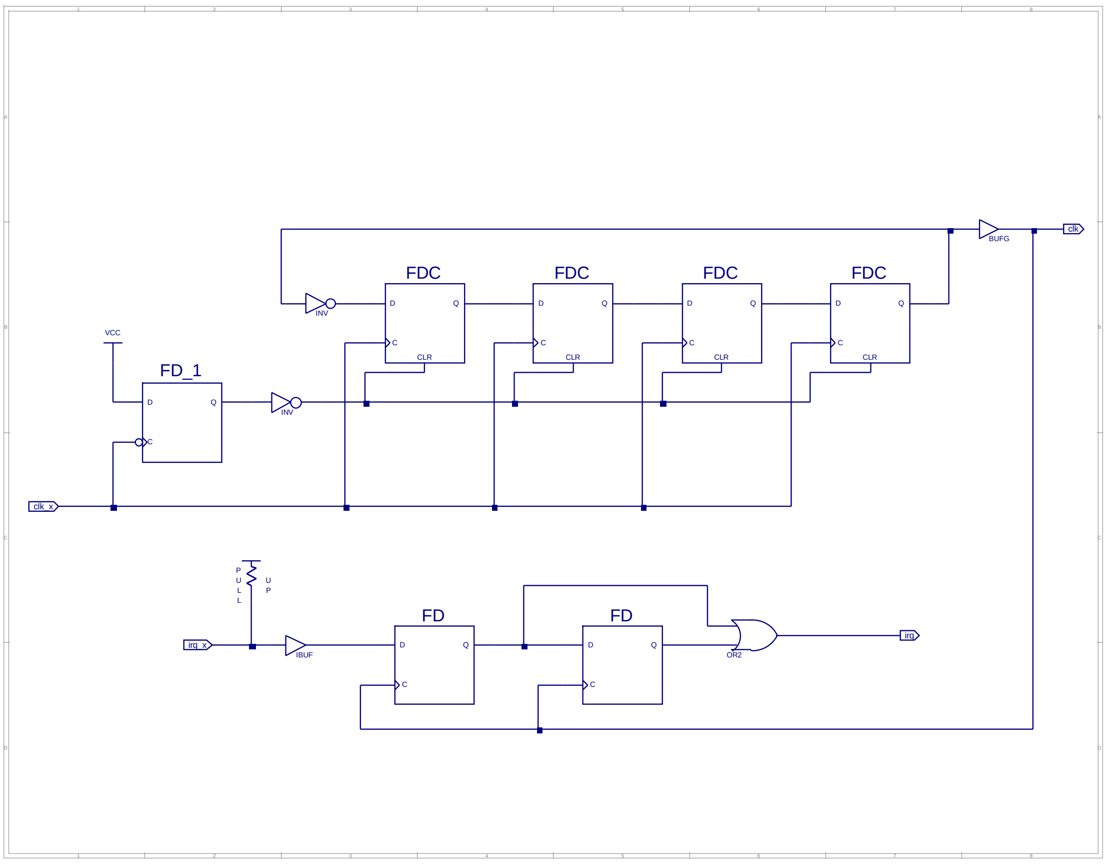
2.3.2 IO



2.3.3 MEM



2.3.4 CLK_IRQ



3. Assembler

XDELA is a very simple assembler. If you execute the assembler, you are asked for the name of the source code input file (enter only the filename without the extension which must be .mac). Also the file "mem.org" must be available in the current directory. The assembler generates four output files:

filename.bin	the binary code
filename.lst	the listing
mem.sch	the schematic for memory (based on mem.org with the binary code inserted)
mem.mem	a hex dump of the binary code to be used with Data2MEM to modify the FPGA bitstream

3.1 MPROZ Instruction Set

br	a	branch to address a)	FLAG = 0
bcc	a	same as br, use it for better readability after an add instruction	
bne	a	same as br, use it for better readability after a nadd/(n)or/(n)and instruction	
add	a,b,c	c = a+b	FLAG = carry
nadd	a,b,c	c = ~(a+b)	FLAG = zero
and	a,b,c	c = ~a and b	FLAG = zero
or	a,b,c	c = ~a or b	FLAG = zero
nand	a,b,c	c = ~a nand b	FLAG = zero
nor	a,b,c	c = ~a nor b	FLAG = zero

3.2 Directives:

dc.b	VALUE1,... ,VALUEn	:	declare byte value (8 bit)
dc.w	VALUE1,... ,VALUEn	:	declare word value (16 bit)
dc.l	VALUE1,... ,VALUEn	:	declare long value (32 bit

blk.b	VALUE1,VALUE2	:	execute VALUE1 times 'dc.b VALUE2'
blk.w	VALUE1,VALUE2	:	execute VALUE1 times 'dc.w VALUE2'
blk.l	VALUE1,VALUE2	:	execute VALUE1 times 'dc.l VALUE2'

: VALUE1 must be calculable in PASS1

blk.b	VALUE	:	Current Location Pointer is incremented by VALUE
blk.w	VALUE	:	Current Location Pointer is incremented by 2*VALUE
blk.l	VALUE	:	Current Location Pointer is incremented by 4*VALUE VALUE must be calculateable in PASS1

WARNING: blk.x (x=b,w,l) with only one parameter modifies only the Current Location Pointer @ and has the same effect as @ = @ + VALUE (*1, *2, *4). No bytes are written to the binary output file. Therefore blk.x must only be used at the end of a program.

odd	:	output a $00 if the Current Location Pointer is even
even	:	output a $00 if the Current Location Pointer is odd
even n	:	output $00 until the Current Location Pointer is a multiple of n
even n,m	:	output m until the Current Location Pointer is a multiple of n

even4	:	same as "even 4" (for comaptibility)
even16	:	same as "even 16" (for comaptibility)

if VALUE	:	the following code is only assembled if VALUE <> 0
else		(VALUE must be calculable in PASS1)
endif		

list	:	increment list level by 1
nolist	:	decrement list level by 1 (a listing is generated if list level >= 0)
label_block	:	defines a new scope for normal labels (in different label blocks you can use the same labels)

3.3 VALUE expressions:

Operators

highest priority

()	: brackets
! ~ -	: logical not, binary not, negative sign
* / \	: multiplication, division, modulo (2er complement)
+ -	: addition, subtraction
<< >>	: left shift, right shift
< > <= >=	: less, greater, less equal, greater equal
== !=	: equal, not equal
&	: binary AND
^	: binary XOR
|	: binary OR
&&	: logical AND (FALSE = 0; TRUE = all other
||	: logical OR TRUE as result = 1)

lowest priority

Operands

xyz	: labels
1234	: decimal numbers
$1234	: hex numbers
%1010	: binary numbers
'a' or "a"	: char byte constant
'ab' or "ab"	: char word constant
'abc' or "abc"	: char 3nyte constant
'abcd' or "abcd"	: char long constant

dc.b allows max. 100 char constants, e.g.
dc.b "this character ' also can be used to enclose the string"

3.4 labels

The first character must be a letter or @ or # or . or _ .
The following characters must be a letter or digit or @ or # or _ .

Labels which start with _ are local labels, i.e. there scope is restricted to the code between to normal labels.

A value is assigned to a label by using a : or = after the label.

abc: assigns the current location counter to abc
abc=VALUE assigns VALUE to abc

abc: and abc=@ has the same effect.
The scope of normal labels is restricted to the code between to label_block instructions.

If you use :: or == instead of : or =, the label is a global label which scope is the complete input file.

Labels which start with a @ can be repeatedly assigned a number (only usable for = assignment). The value must be calculable in PASS1. This labels are always global labels and doesn't start a new scope for normal labels.

If a normal label starts with a ".", no new scope for local labels is started.

The label with the single letter @ is a reserved label (the Current Location Pointer). Normally @ shouldn't be modified within the program (see remark to blk.b).

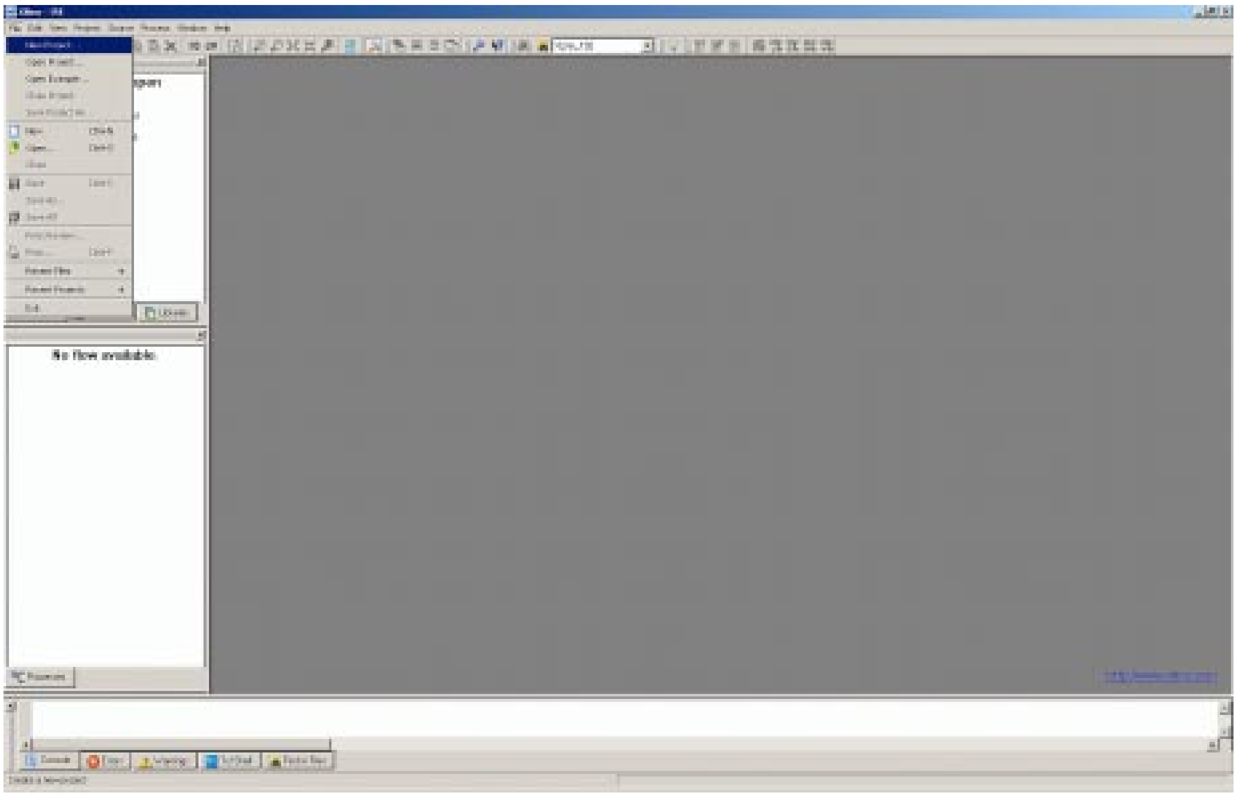
The label with the name @@ is a reserved label (the file pointer for the binary output file) and must not be modified.

4. Step by step for beginners

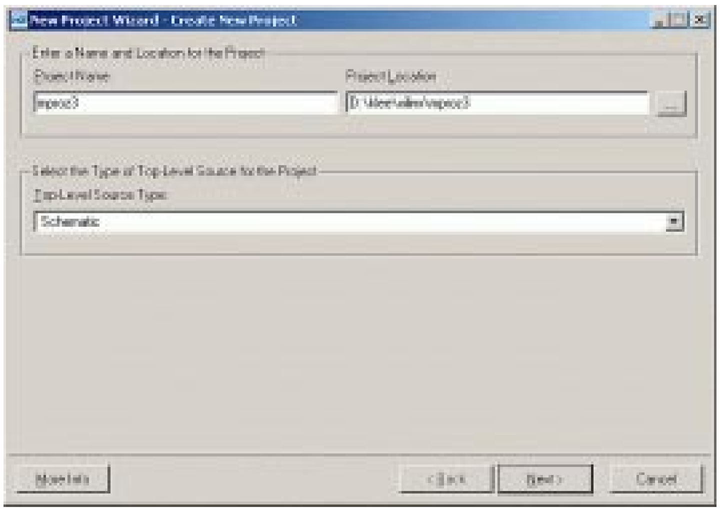
Download mproz3.zip and unzip it in a temporary directory. There should be two folders ("import" and "addon") and this pdf file.

If you don't have already installed the Xilinx software, download the free Webpack from Xilinx's Website (http://www.xilinx.com/ise/logic_design_prod/webpack.htm) and install it.

Start the program and create a new project:



Left click on "New Project..."



New Project Wizard - Create New Project

Enter a Name and Location for the Project

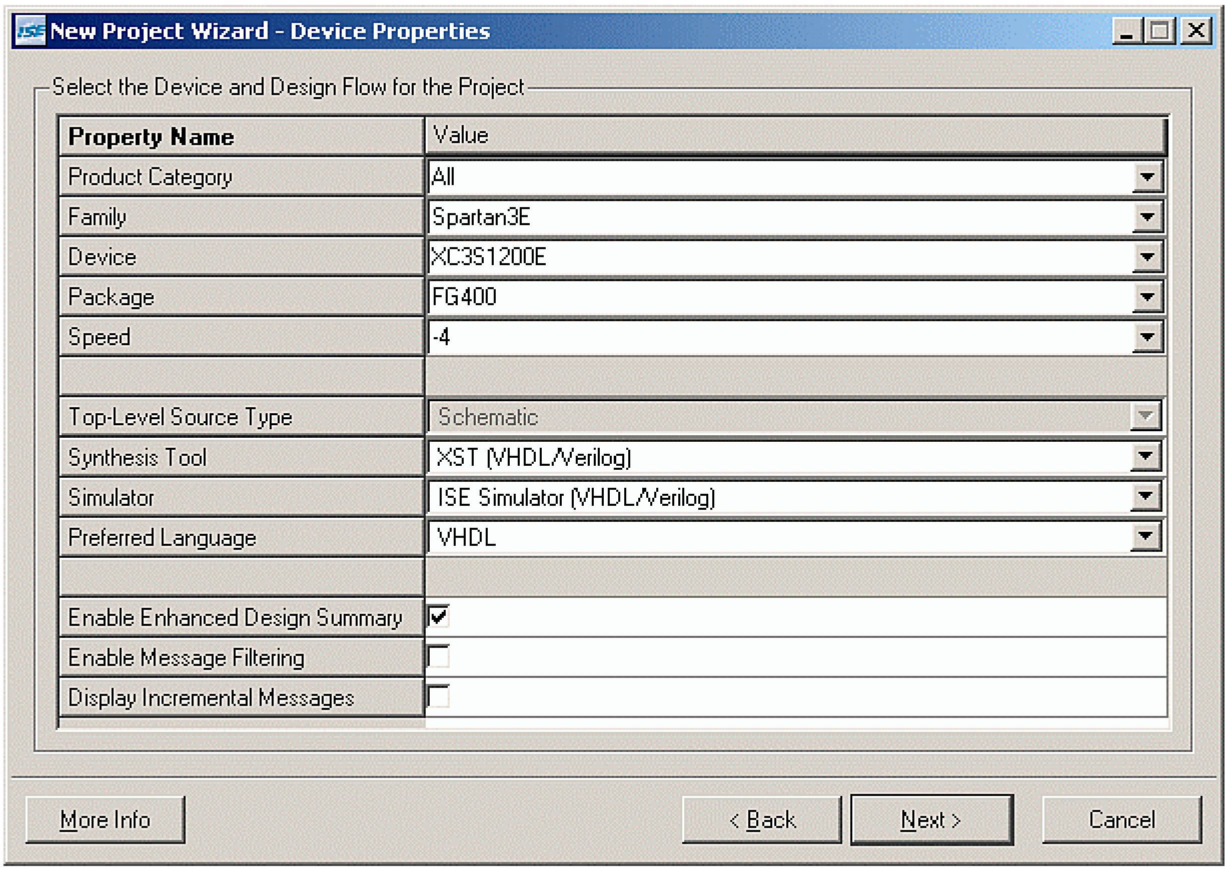
Project Name: mproz3

Project Location: D:\dee\xilinx\mproz3

Select the Type of Top-Level Source for the Project

Top-Level Source Type: Schematic

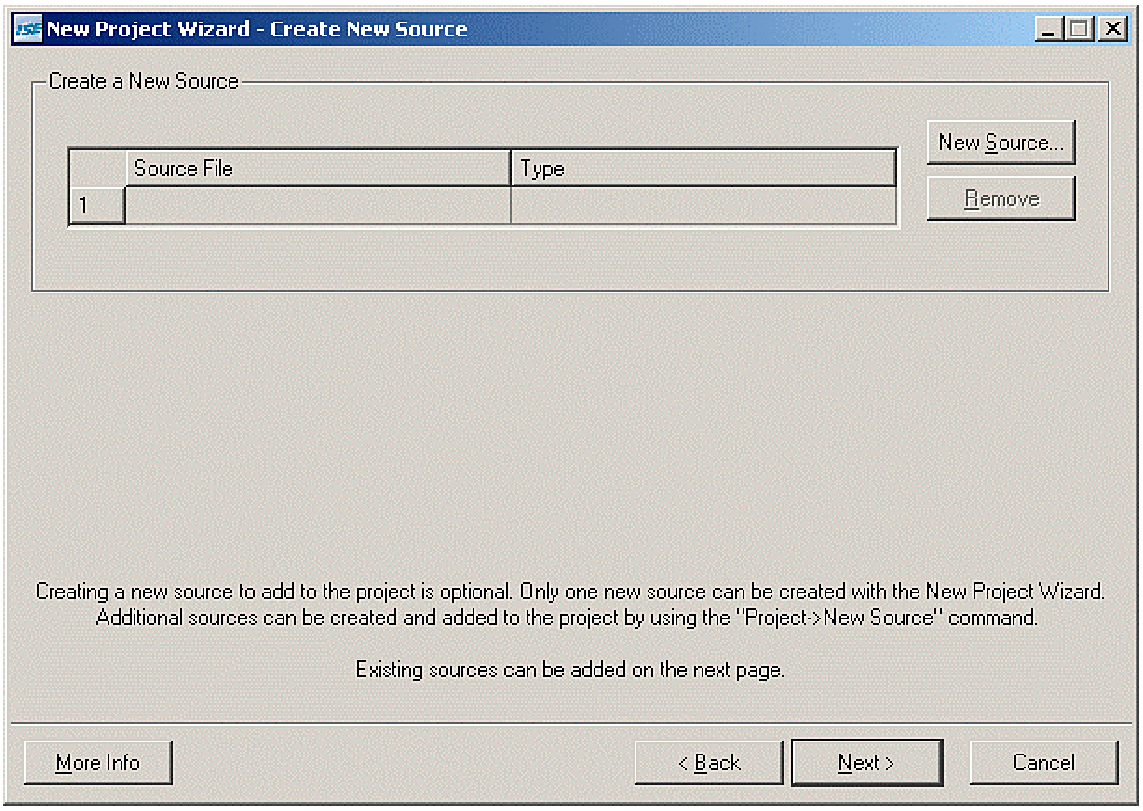
Enter Project Name and Location and click "Next"


New Project Wizard - Device Properties

Select the Device and Design Flow for the Project

Property Name	Value
Product Category	All
Family	Spartan3E
Device	XC3S1200E
Package	FG400
Speed	-4
Top-Level Source Type	Schematic
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISE Simulator (VHDL/Verilog)
Preferred Language	VHDL
Enable Enhanced Design Summary	✓
Enable Message Filtering	☐
Display Incremental Messages	☐

Make the above selections and click "Next".


New Project Wizard - Create New Source

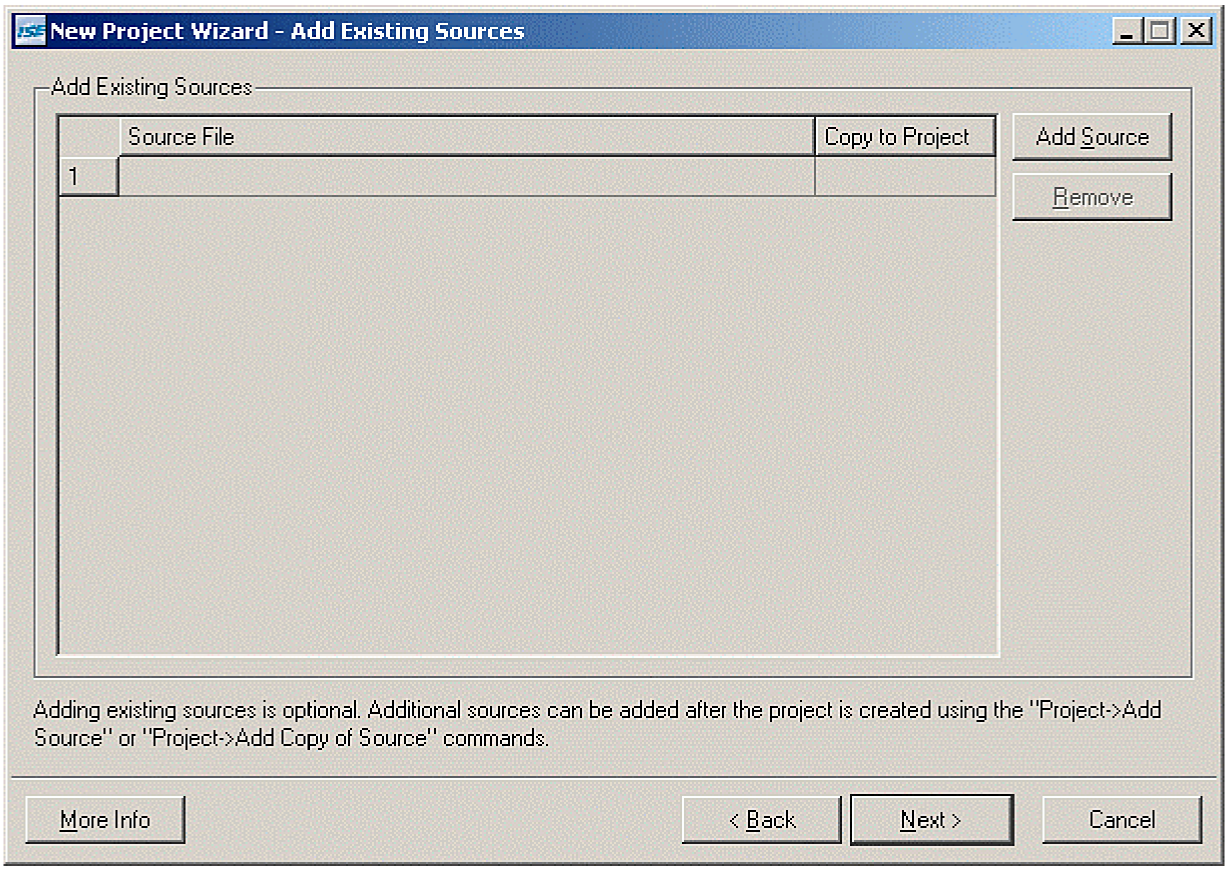
Create a New Source

	Source File	Type
1		

Creating a new source to add to the project is optional. Only one new source can be created with the New Project Wizard. Additional sources can be created and added to the project by using the "Project->New Source" command.

Existing sources can be added on the next page.

Don't create any new source. Click "Next"

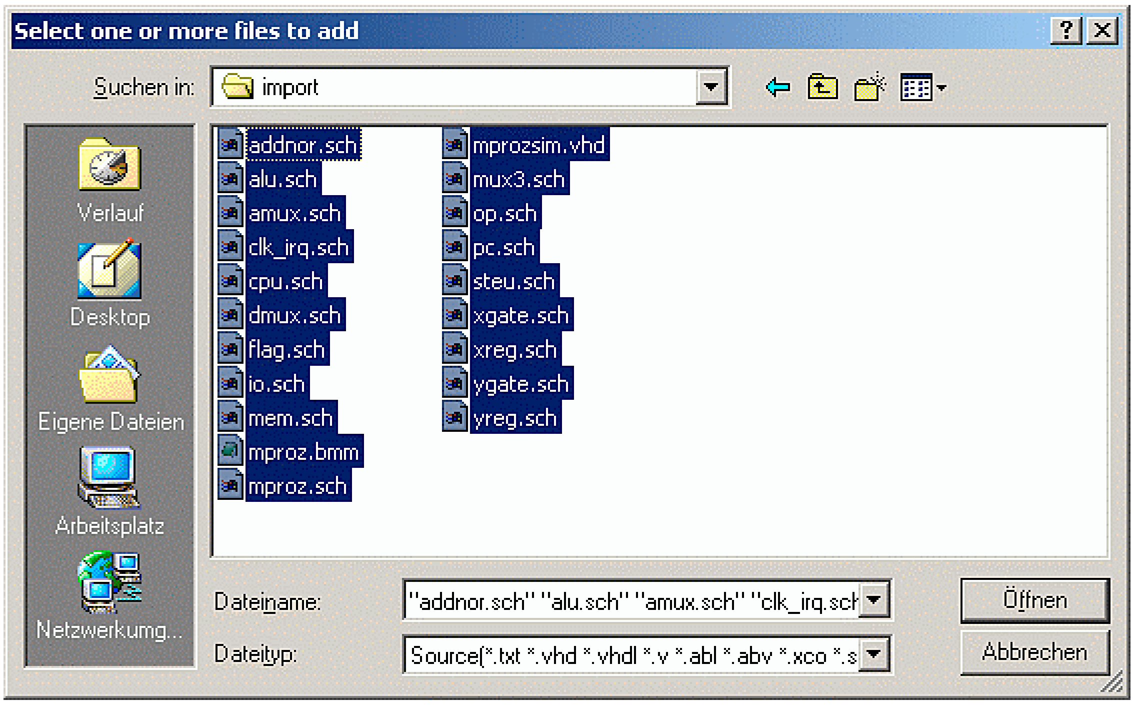

New Project Wizard - Add Existing Sources

Add Existing Sources

	Source File	Copy to Project
1		

Adding existing sources is optional. Additional sources can be added after the project is created using the "Project->Add Source" or "Project->Add Copy of Source" commands.

Click "Add Source"


Select one or more files to add

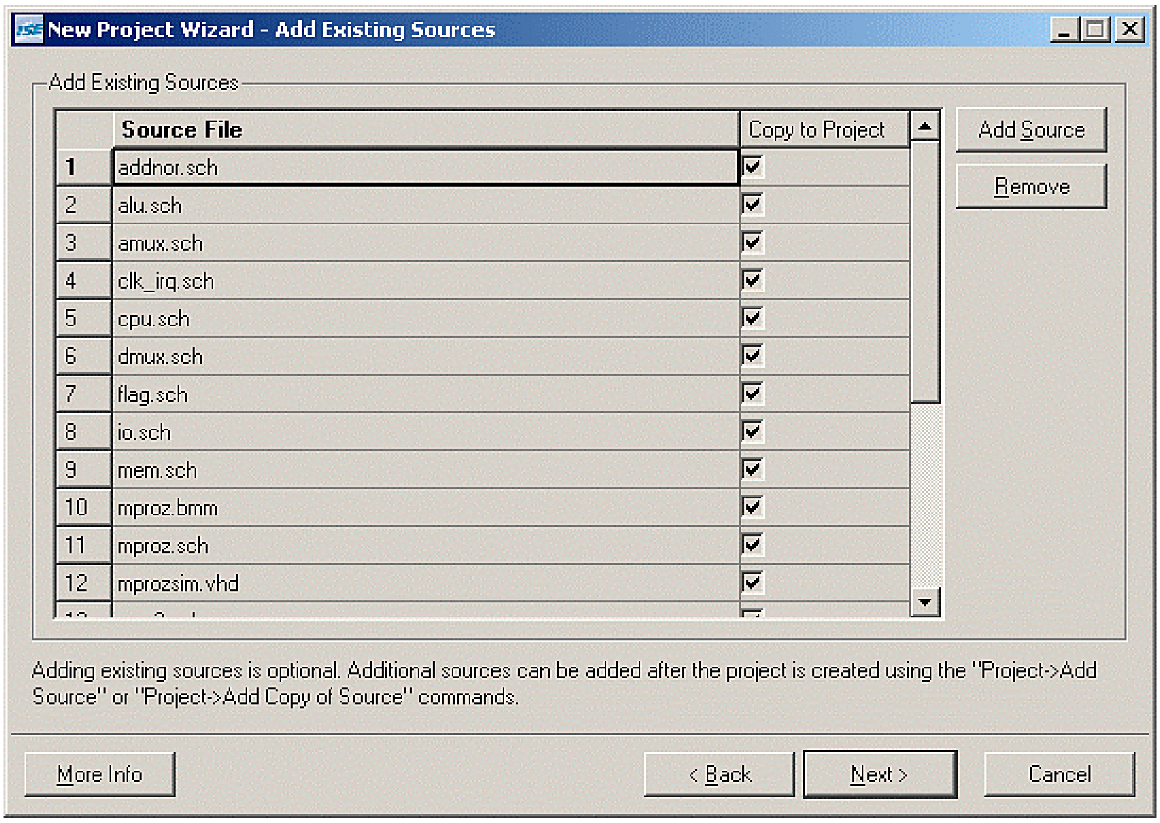
Suchen in: import

addnor.sch
alu.sch
amux.sch
clk_irq.sch
cpu.sch
dmux.sch
flag.sch
io.sch
mem.sch
mproz.bmm
mproz.sch
mprozsim.vhd
mux3.sch
op.sch
pc.sch
steu.sch
xgate.sch
xreg.sch
ygate.sch
yreg.sch

Dateiname: "addnor.sch" "alu.sch" "amux.sch" "clk_irq.sch

Dateityp: Source(*.txt *.vhd *.vhdl *.v *.abl *.abv *.xco *.s

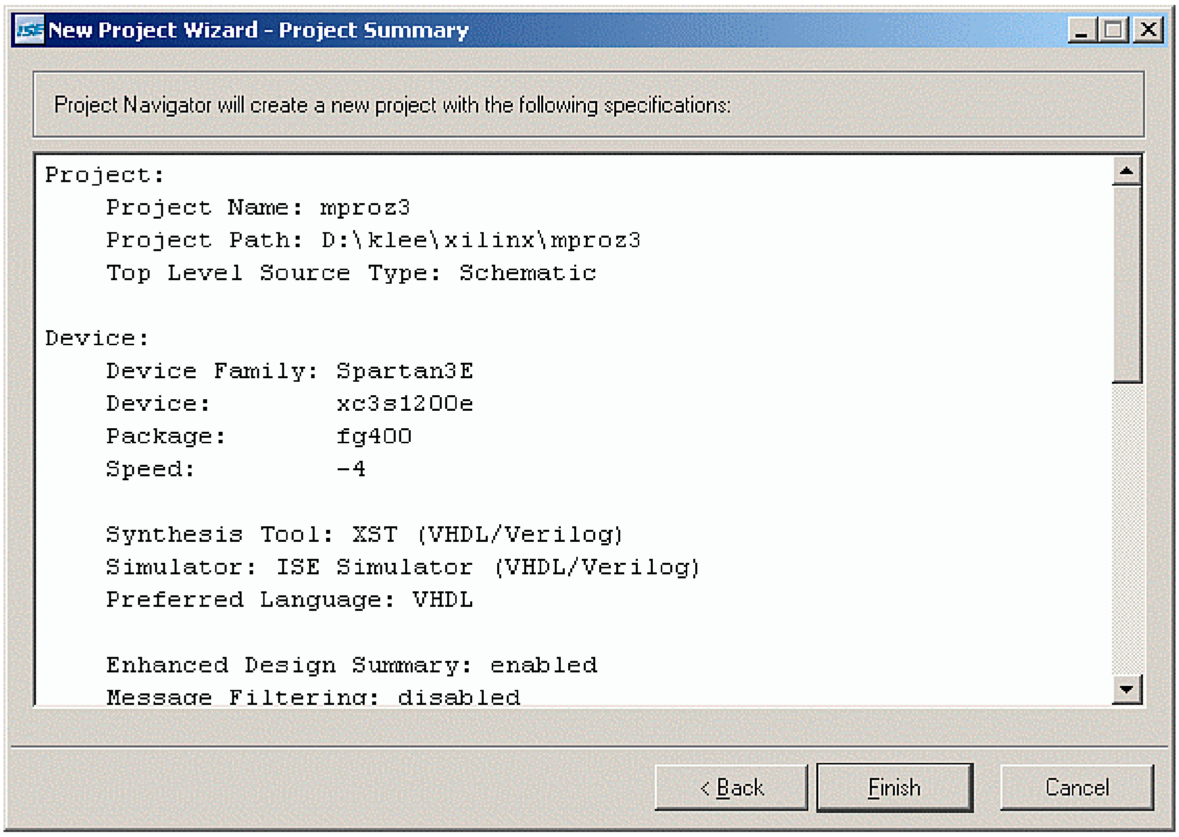
Select all files in the folder "imort" (from mproz3.zip). Be sure to not forget mproz.bmm or you will not be able to update the bit stream with a new software later. Click "Open".



	Source File	Copy to Project
1	addnor.sch	✓
2	alu.sch	✓
3	amux.sch	✓
4	clk_irq.sch	✓
5	cpu.sch	✓
6	dmux.sch	✓
7	flag.sch	✓
8	io.sch	✓
9	mem.sch	✓
10	mproz.bmm	✓
11	mproz.sch	✓
12	mprozsim.vhd	✓

Adding existing sources is optional. Additional sources can be added after the project is created using the "Project->Add Source" or "Project->Add Copy of Source" commands.

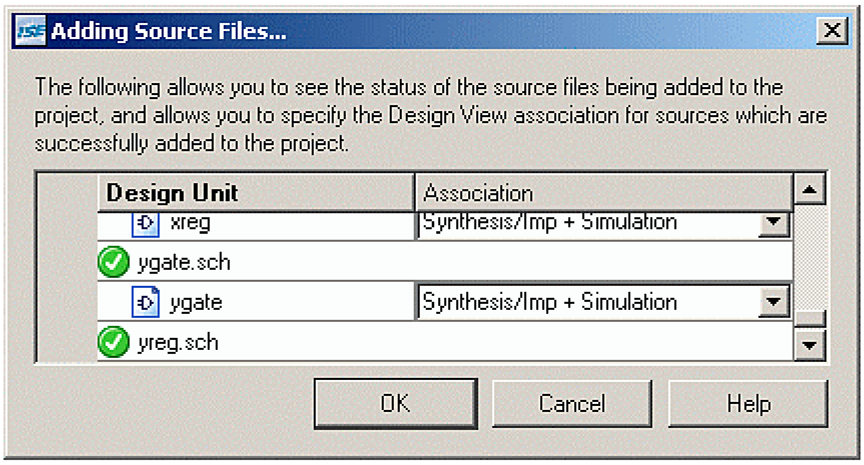
Click "Next"



Project Navigator will create a new project with the following specifications:

```
Project:
    Project Name: mproz3
    Project Path: D:\klee\xilinx\mproz3
    Top Level Source Type: Schematic

Device:
    Device Family: Spartan3E
    Device:        xc3s1200e
    Package:       fg400
    Speed:         -4

    Synthesis Tool: XST (VHDL/Verilog)
    Simulator: ISE Simulator (VHDL/Verilog)
    Preferred Language: VHDL

    Enhanced Design Summary: enabled
    Message Filtering: disabled
```

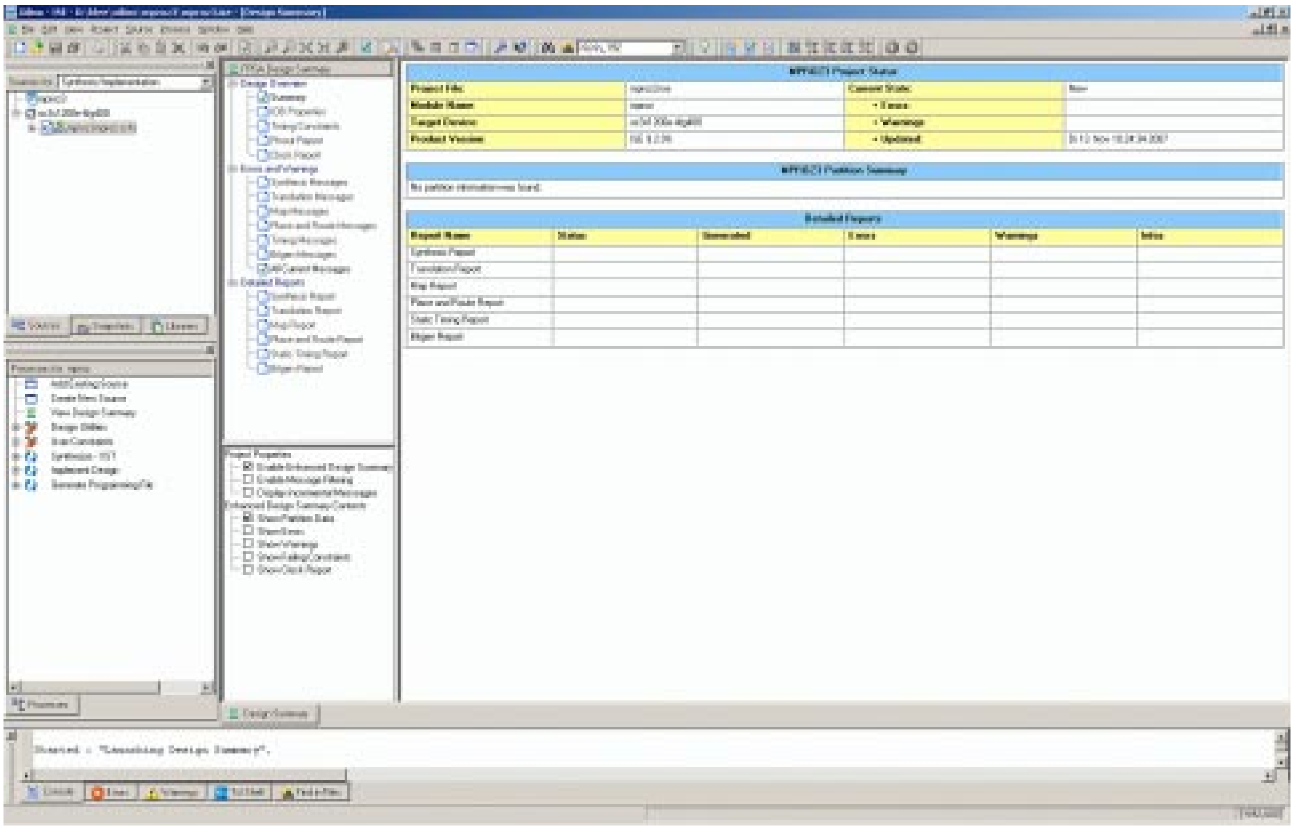
Click "Finish"


Adding Source Files...

The following allows you to see the status of the source files being added to the project, and allows you to specify the Design View association for sources which are successfully added to the project.

Design Unit	Association
xreg	Synthesis/Imp + Simulation
ygate.sch	
ygate	Synthesis/Imp + Simulation
yreg.sch	

Wait until all files are imported. Click "OK".



Project Status			
Project File:	[illegible]	Current State:	[illegible]
Module Name:	[illegible]	• Errors:	
Target Device:	[illegible]	• Warnings:	
Product Version:	[illegible]	• Updated:	[illegible]

Partition Summary

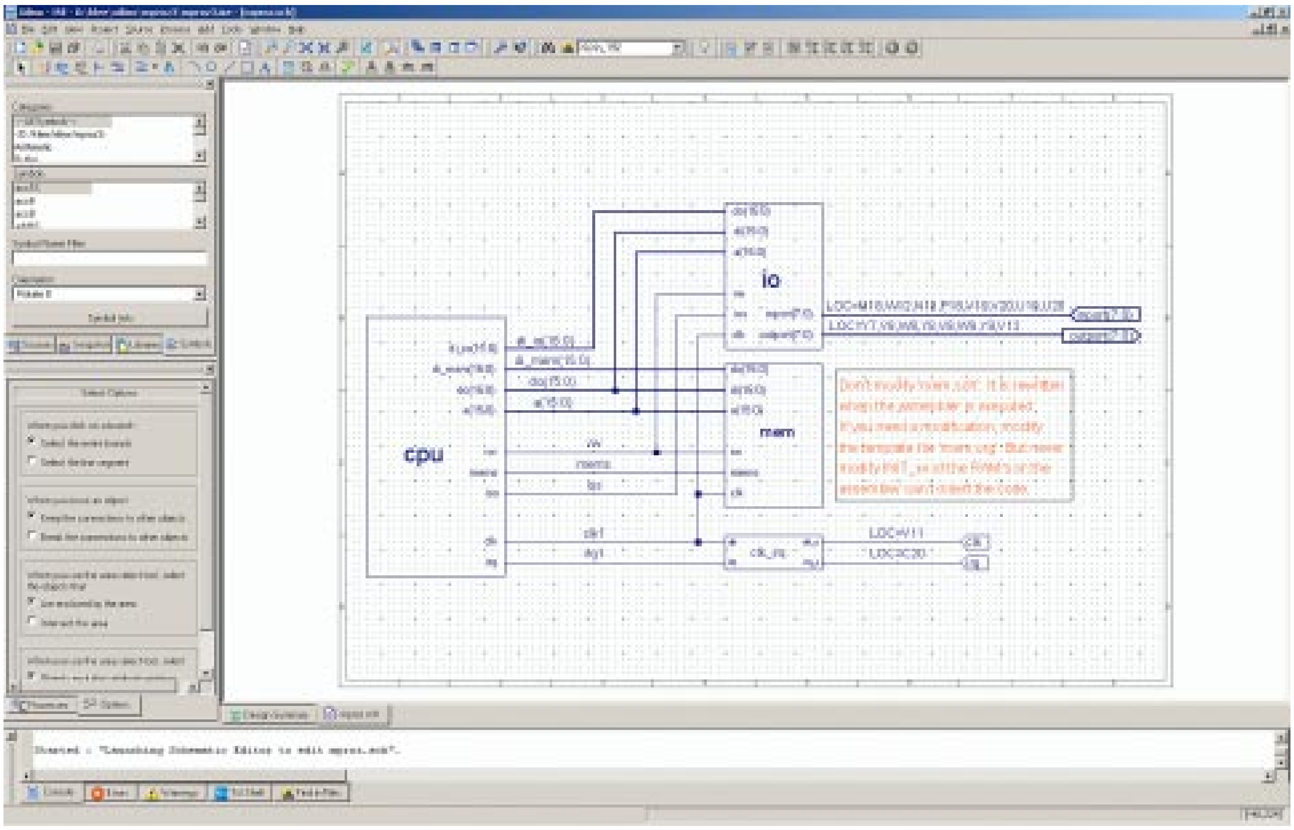
No partition information was found.

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report					
Translation Report					
Map Report					
Place and Route Report					
Static Timing Report					
Bitgen Report					

```
Started : "Launching Design Summary".
```

You should get the above screen**. Now copy all files in the folder "addon" (from mproz.zip) to the chosen Project directory.**

Double click "mproz.sch":

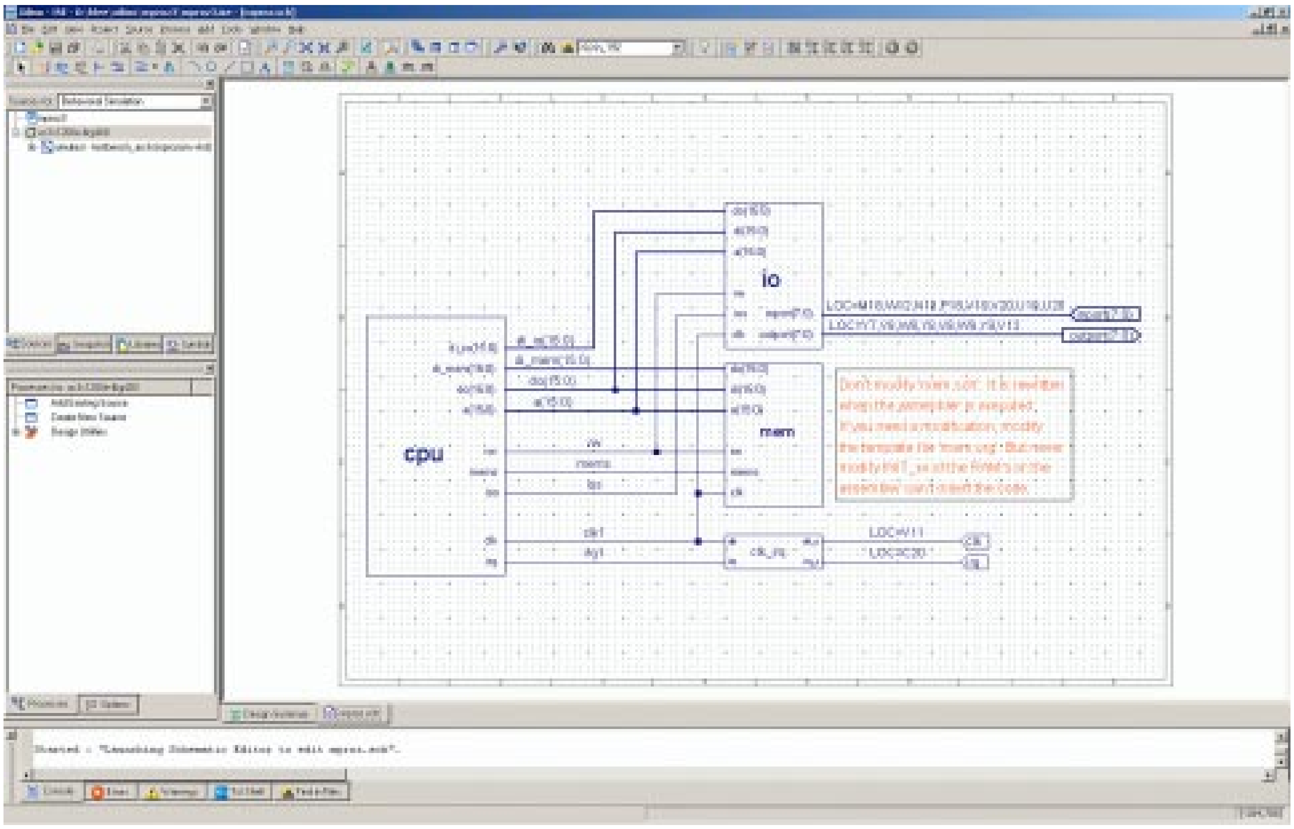


Don't modify 'mem.coe'. It is rewritten when the assembler is executed. If you need a modification, modify the template file 'mem.org'. But never modify INIT_xx of the RAMB's or the assembler can't insert the code.

```
Started : "Launching Schematic Editor to edit mproc.sch".
```

Now you can inspect (and modify) the schematics.

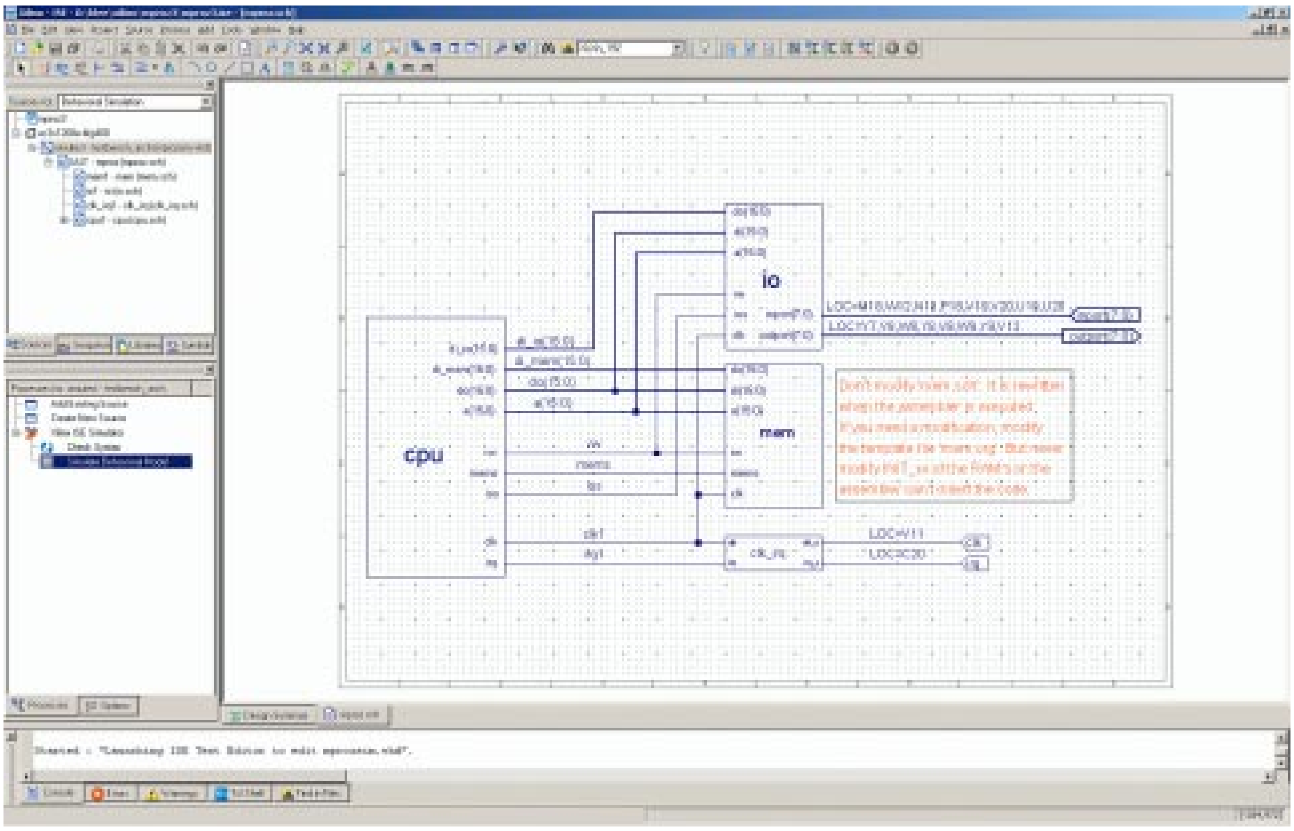
Select the tab "Sources" and "Behavioral Simulation" in the upper left Window:



Don't modify 'mem.coe'. It is rewritten when the assembler is executed. If you need a modification, modify the template file 'mem.org'. But never modify INIT_xx of the RAMB's or the assembler can't insert the code.

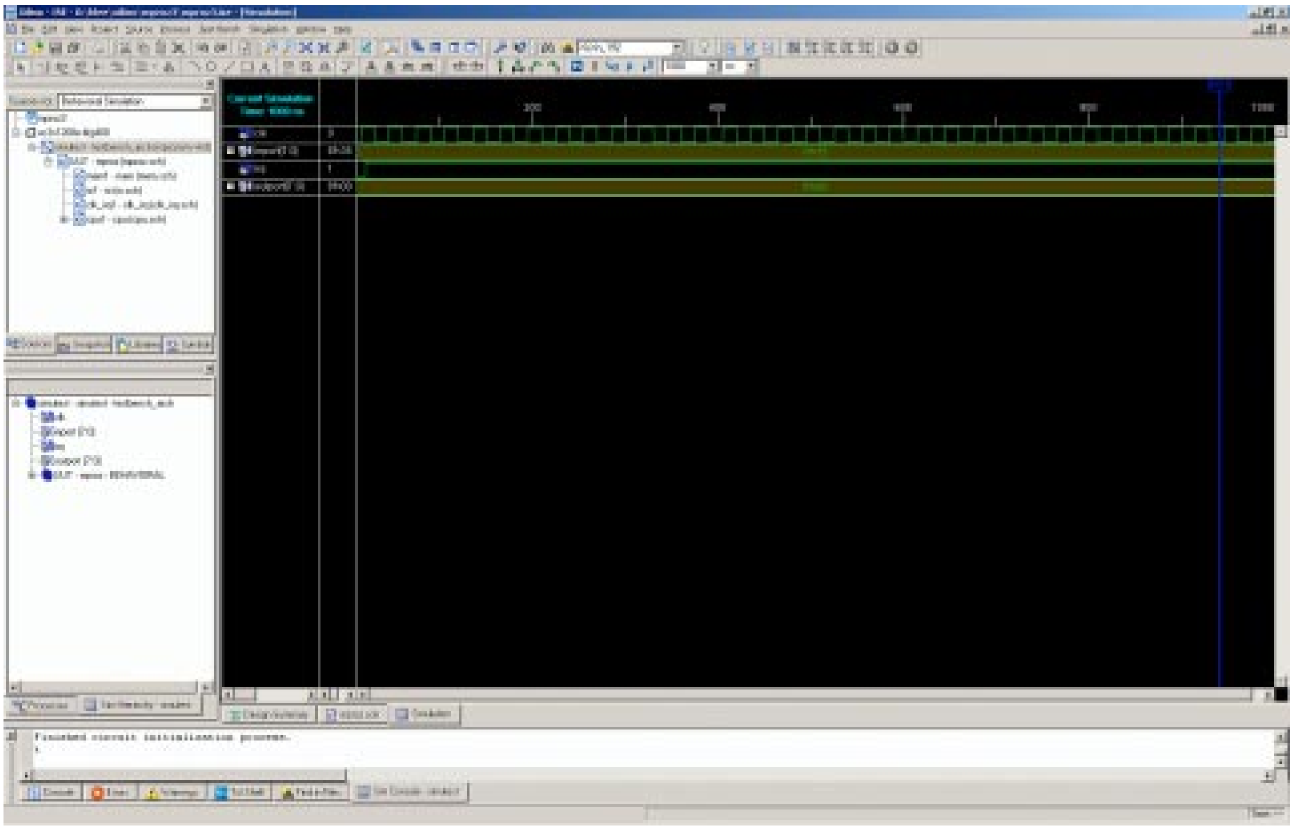
```
Started : "Launching Schematic Editor to edit mproc.sch".
```

Select "simutest - testbench_arch (mprozsim.vhd)" and double click "Simulate Behavioral Model":



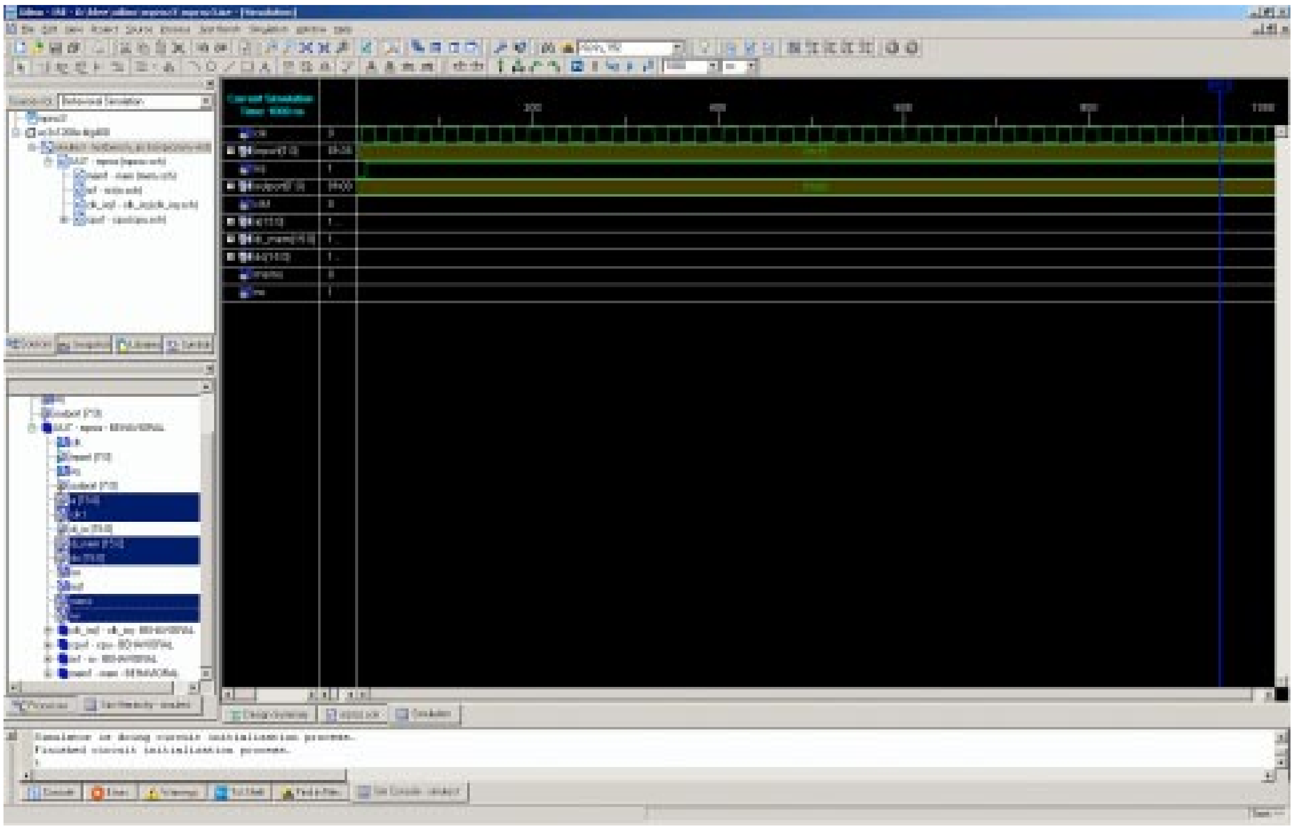
```
Started : "Launching ISE Text Editor to edit mprozsim.vhd".
```

You will get:



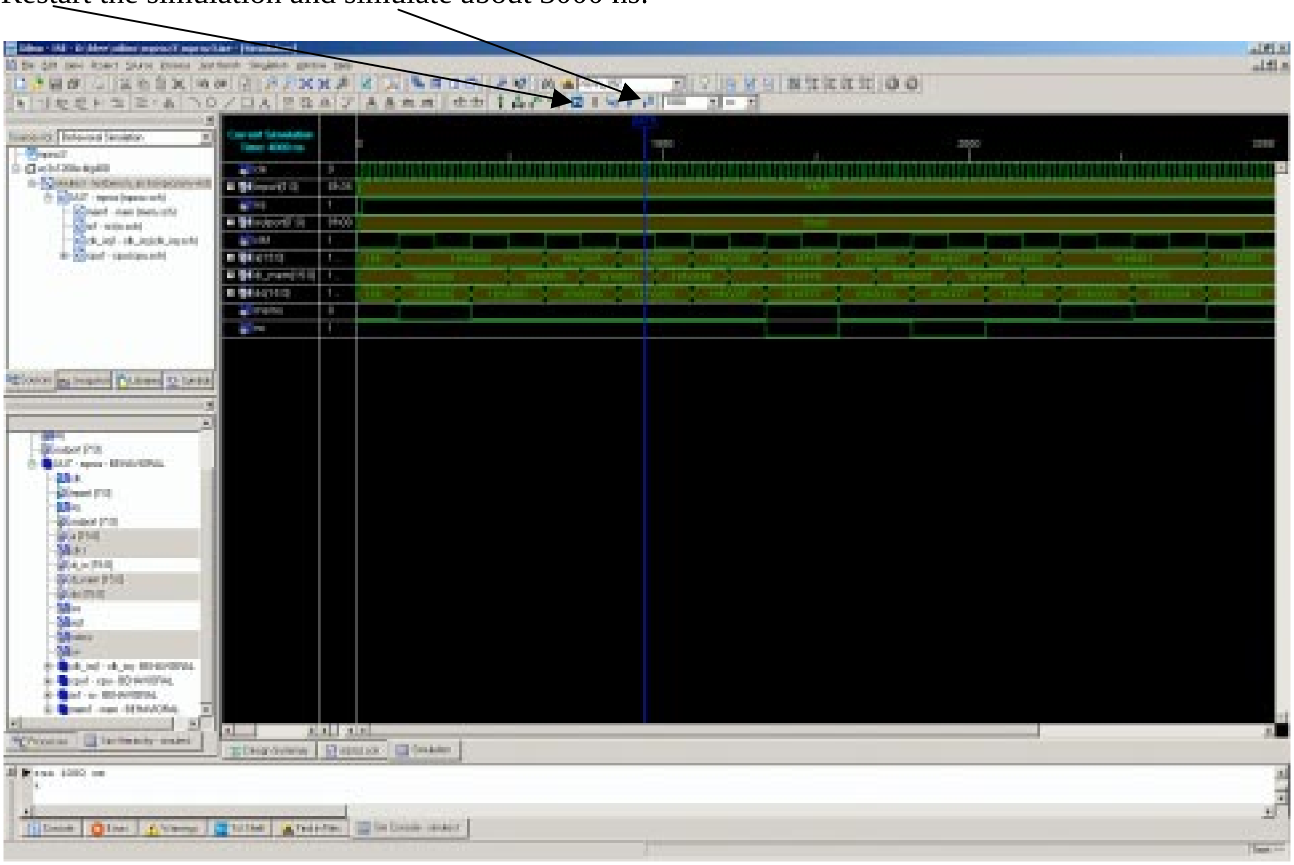
```
Finished circuit initialization process.
```

Add the signals clk1, a[15:0], di_mem[15:0], do[15:0], ms and rw by a right click on the signals.



```
Simulator is doing circuit initialization process.
Finished circuit initialization process.
```

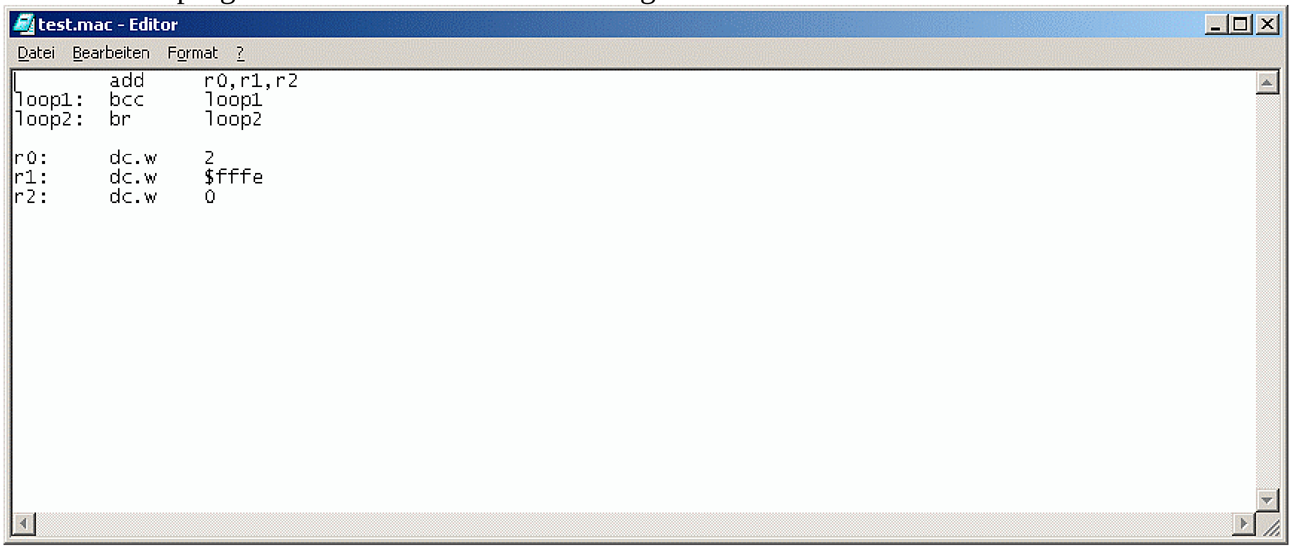
Restart the simulation and simulate about 3000 ns:



```
run 3000 ns
```

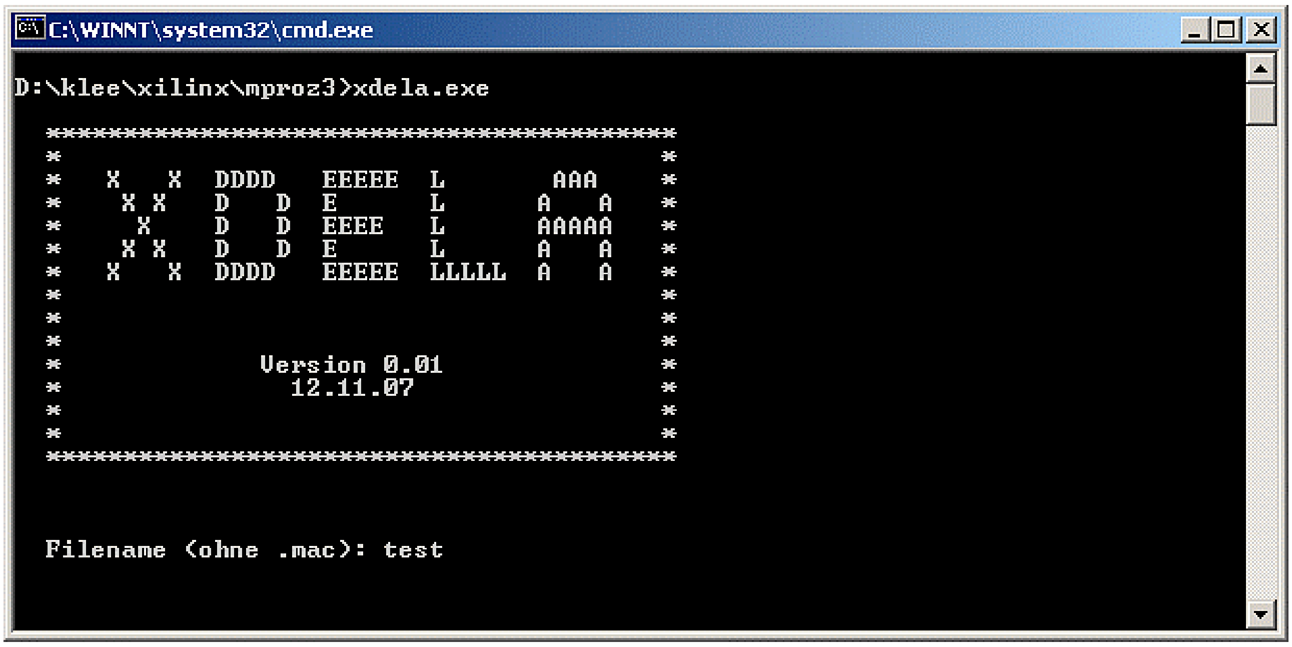
You see the simulation of the add and br instruction. Because there is no Carry, the processor loops in "loop1".

Now edit the program source "test.mac" and change the content of r0 from 1 to 2:



```
        add     r0,r1,r2
loop1:  bcc     loop1
loop2:  br      loop2

r0:     dc.w    2
r1:     dc.w    $fffe
r2:     dc.w    0
```

Save it and start XDELA (either by typing XDELA.EXE in a CMD window or double clicking XDELA.BAT within Windows Explorer:

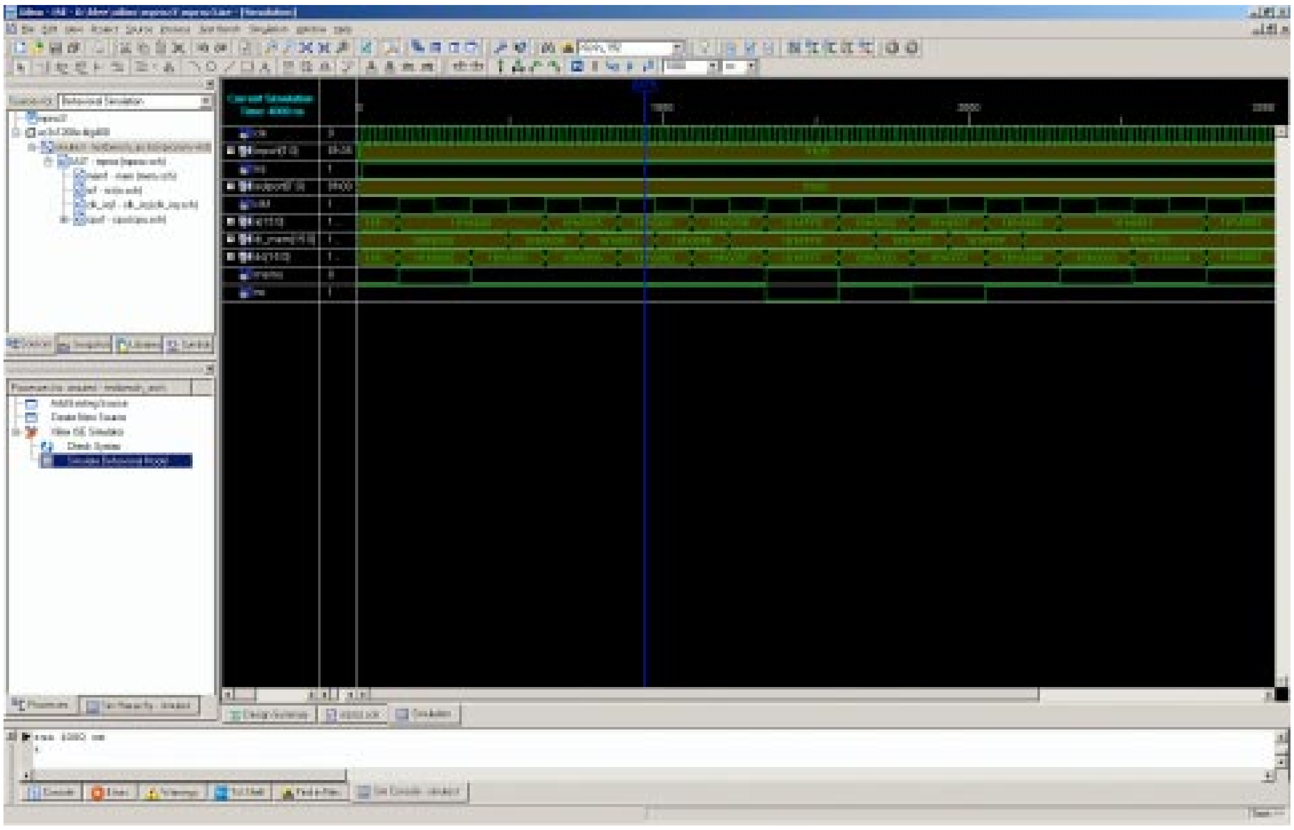


```
D:\klee\xilinx\mproz3>xdela.exe

  ******************************************
  *                                        *
  *   X   X  DDDD   EEEEE  L       AAA     *
  *    X X   D   D  E      L      A   A    *
  *     X    D   D  EEEE   L      AAAAA    *
  *    X X   D   D  E      L      A   A    *
  *   X   X  DDDD   EEEEE  LLLLL  A   A    *
  *                                        *
  *                                        *
  *                                        *
  *             Version 0.01               *
  *               12.11.07                 *
  *                                        *
  *                                        *
  ******************************************


  Filename (ohne .mac): test
```

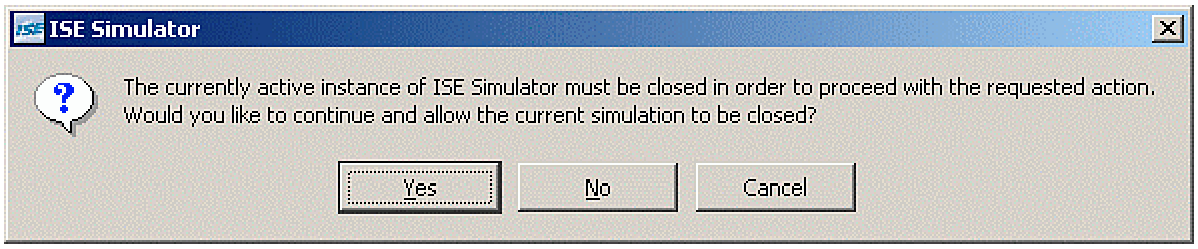
Enter test <cr>

XDELA will modify the schematic "mem.sch" so it now contains the new program.

Again double click "Simulate Behavioral Model"

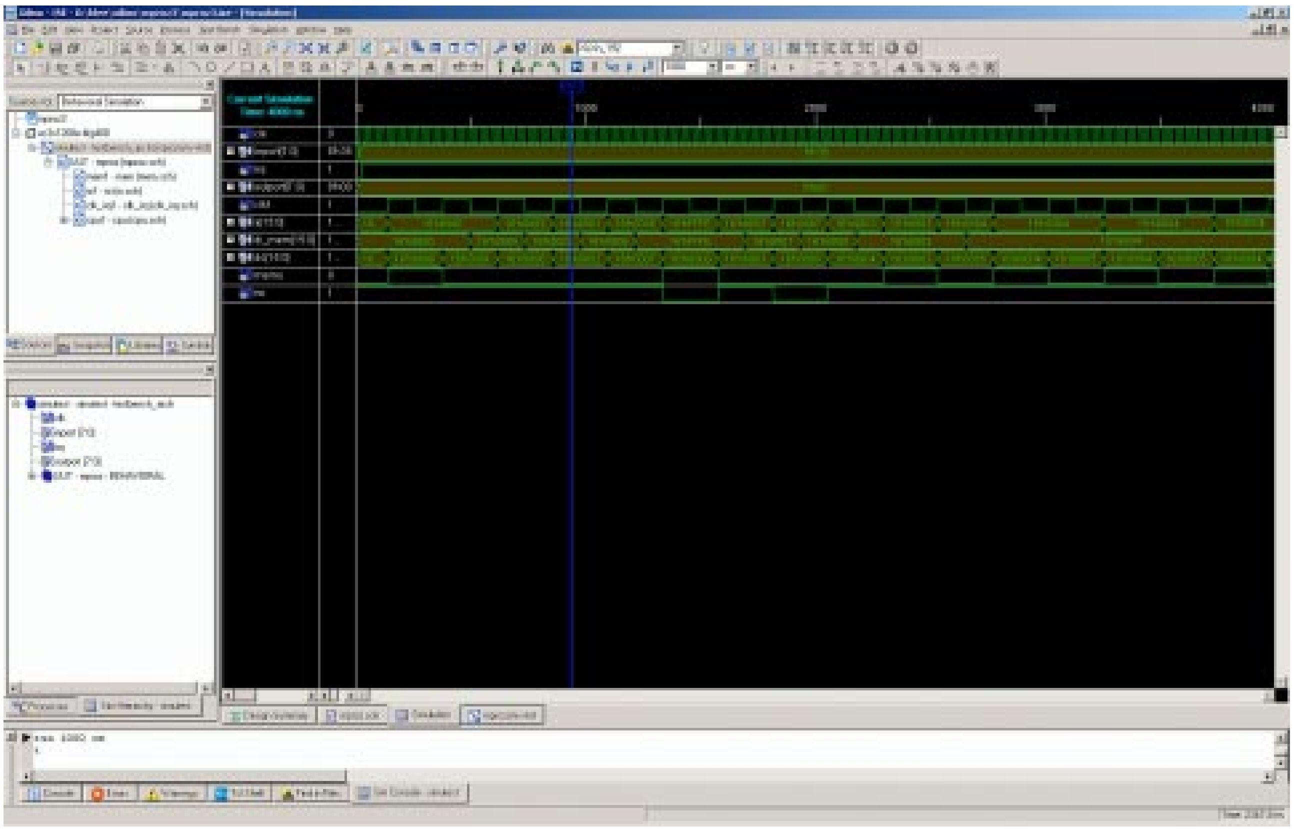


You will get:


ISE Simulator

The currently active instance of ISE Simulator must be closed in order to proceed with the requested action. Would you like to continue and allow the current simulation to be closed?

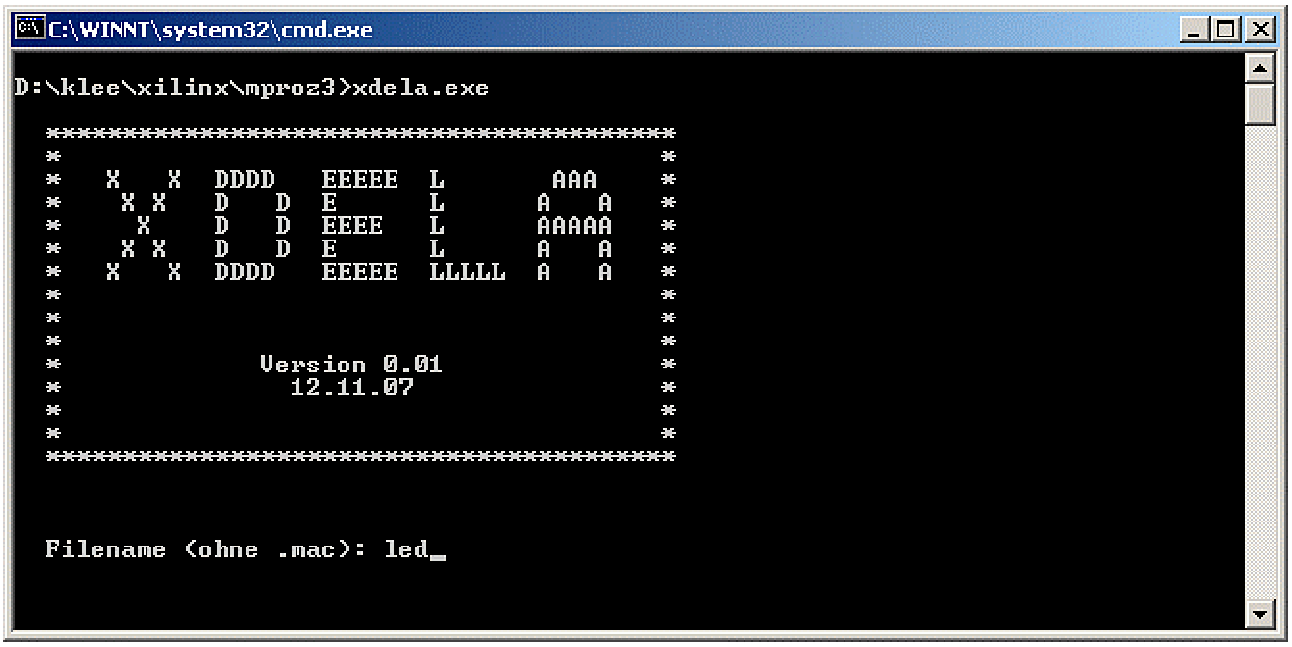
Click "Yes".



Because now the add will generate a Carry, the processor loops in "loop2".

How to load a design into the DARNAW1 board.

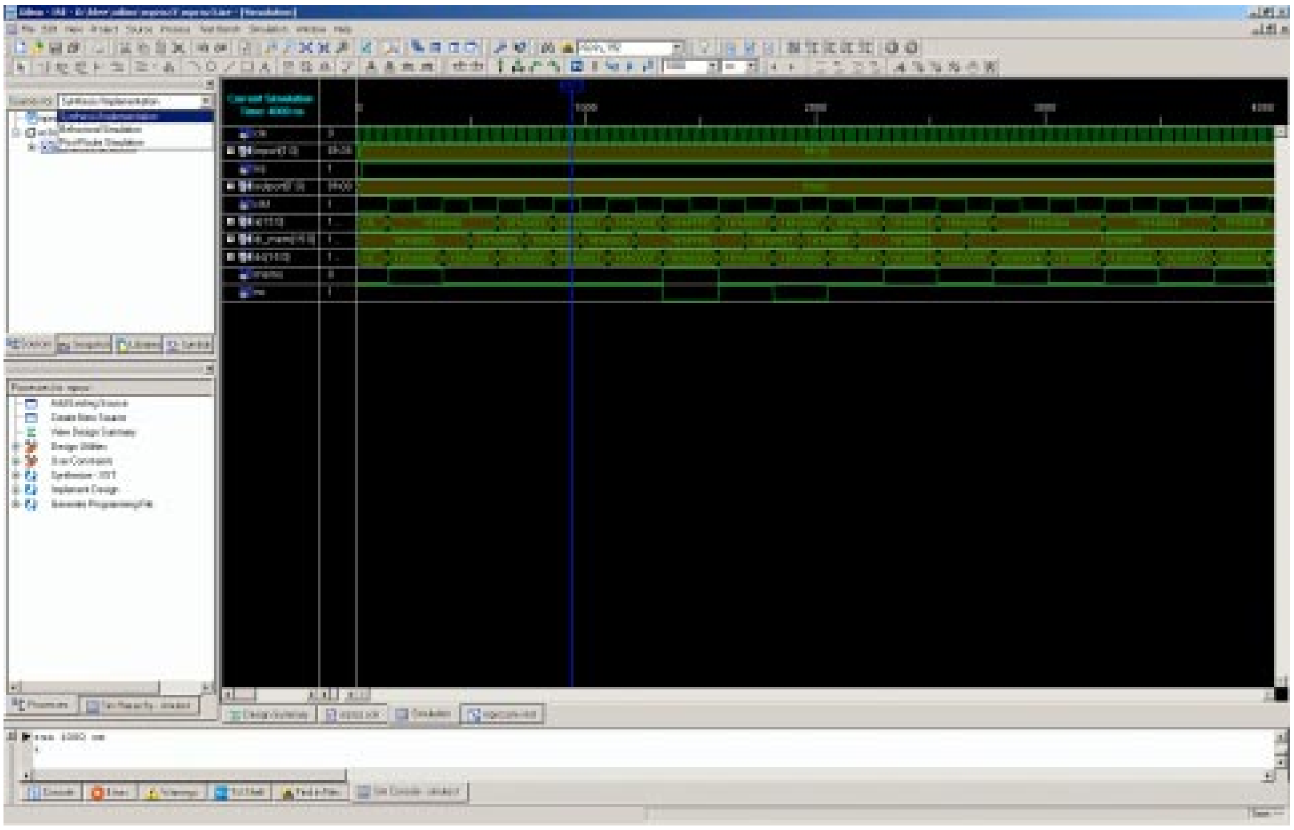
Assemble the LED demo program for the prototype board. Start XDELA:



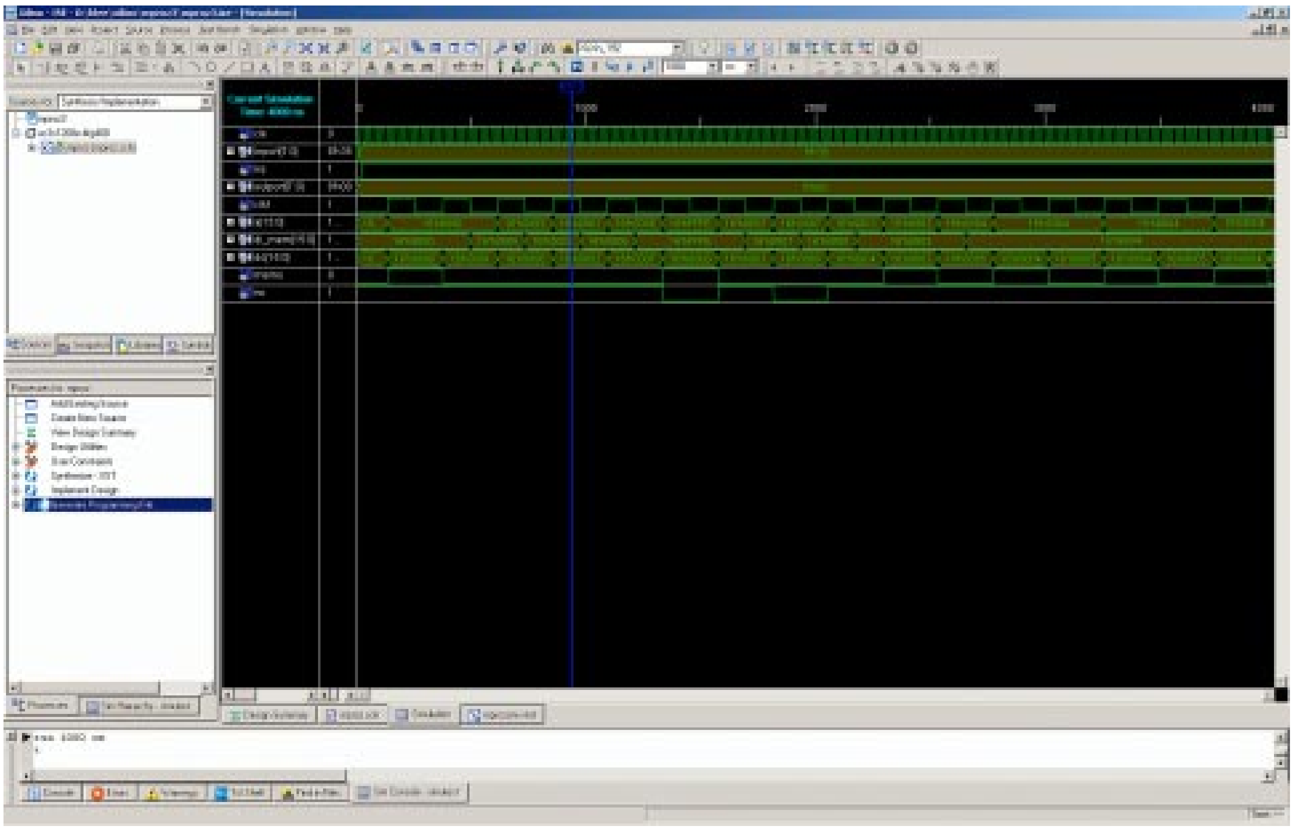
```
C:\WINNT\system32\cmd.exe

D:\klee\xilinx\mproz3>xdela.exe

  *****************************************
  *                                       *
  *   X   X  DDDD   EEEEE  L       AAA    *
  *    X X   D   D  E      L      A   A   *
  *     X    D   D  EEEE   L      AAAAA   *
  *    X X   D   D  E      L      A   A   *
  *   X   X  DDDD   EEEEE  LLLLL  A   A   *
  *                                       *
  *                                       *
  *                                       *
  *            Version 0.01               *
  *              12.11.07                 *
  *                                       *
  *                                       *
  *****************************************


  Filename (ohne .mac): led_
```

Enter led <cr>

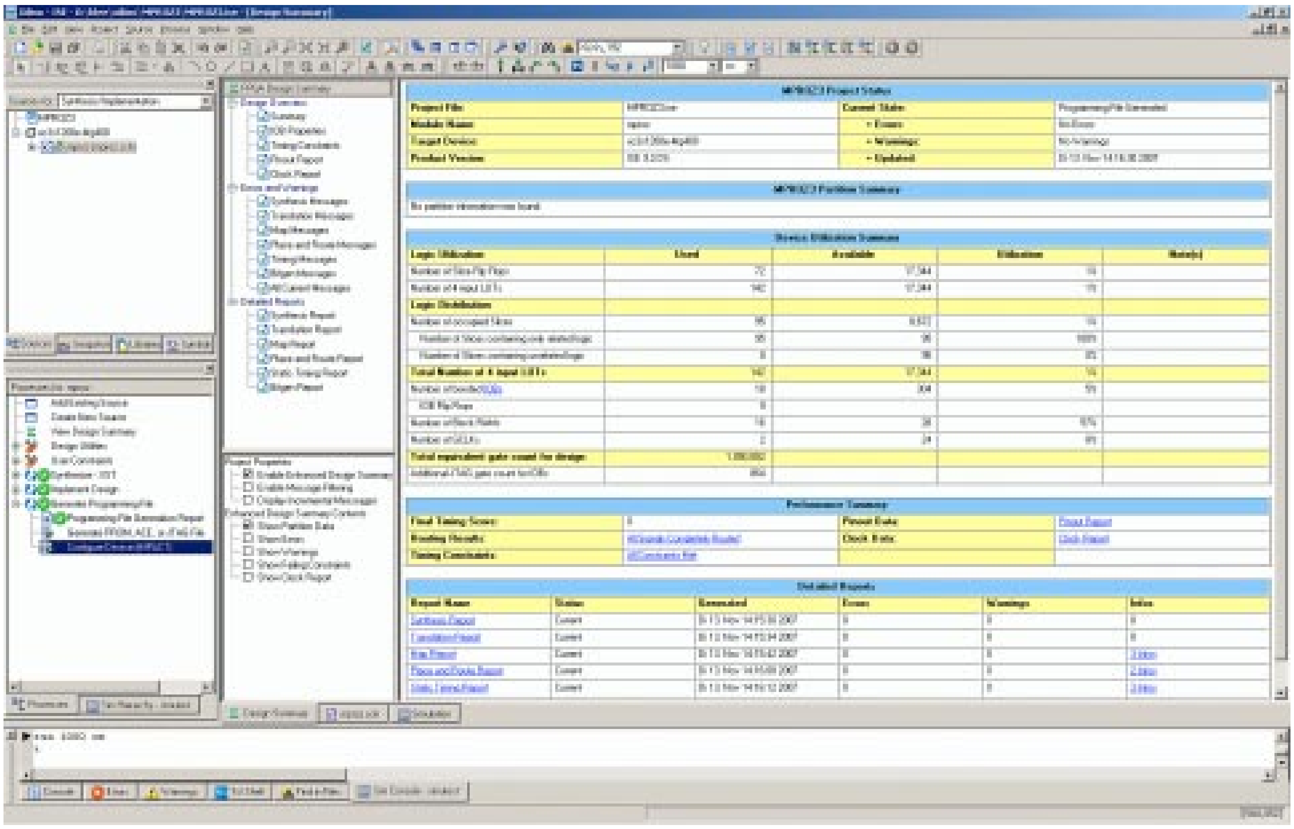
Select "Synthesis/Implementation" instead of "Behavioral Simulation":



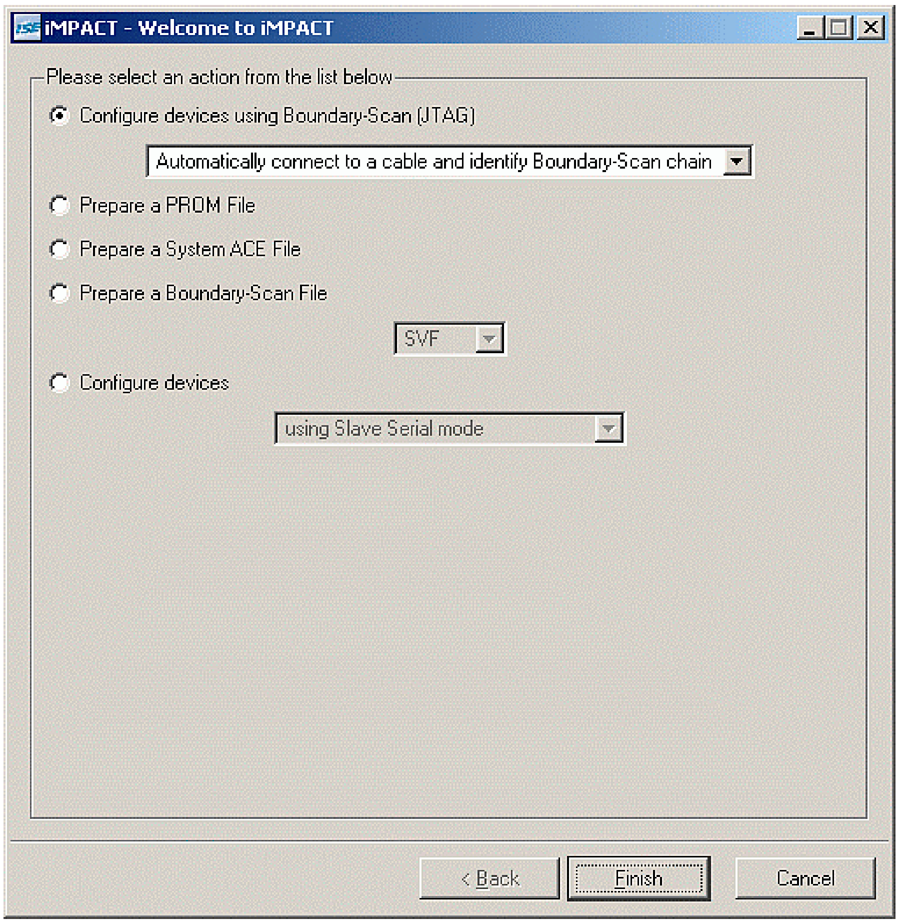
Double click on "Generate Programming File":



Select the tab "Design Summary".You should get:



Double click "Configure Device (iMPACT)".



iMPACT - Welcome to iMPACT

Please select an action from the list below

[● Configure devices using Boundary-Scan (JTAG)]

Automatically connect to a cable and identify Boundary-Scan chain

○ Prepare a PROM File

○ Prepare a System ACE File

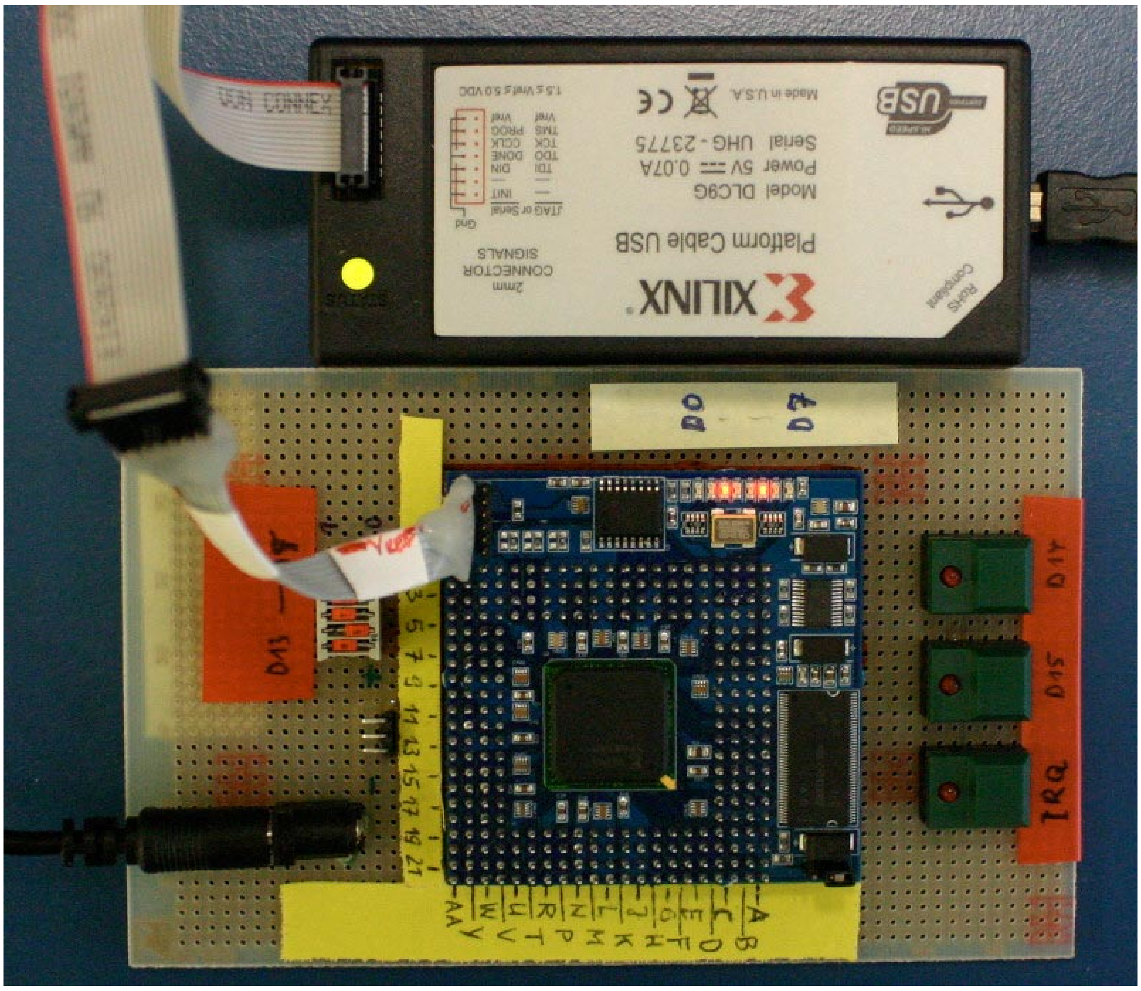
○ Prepare a Boundary-Scan File

SVF

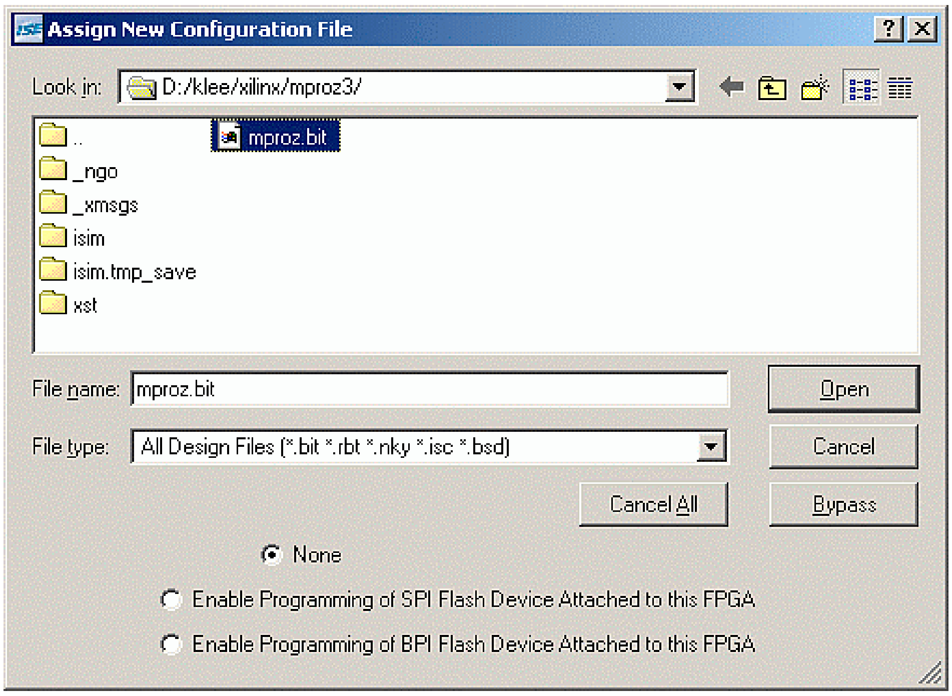
○ Configure devices

using Slave Serial mode

Connect the download cable to the prototype board an click "Finish"



With some PC's we had problems with the parallel port cable (the one provided with the DARNAW1 board and the XILINX one) so we used the USB download cable. If you use a Xilinx cable, you need an adapter cable because the pin ordering is different from the DARNAW1 download cable. Connect the cable to the outer connector and remove the jumper.


Assign New Configuration File

Look in: D:/klee/xilinx/mproz3/

..
_ngo
_xmsgs
isim
isim.tmp_save
xst
mproz.bit

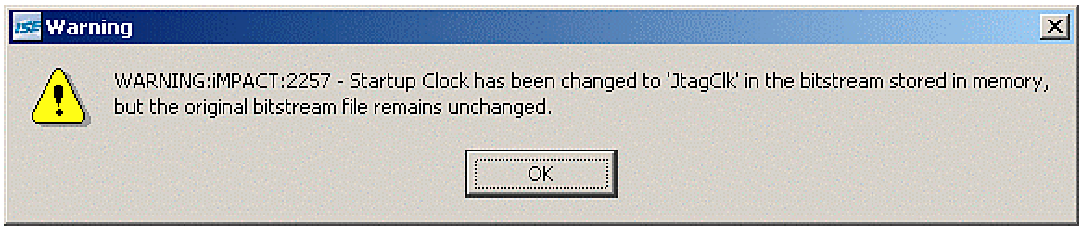
File name: mproz.bit

File type: All Design Files (*.bit *.rbt *.nky *.isc *.bsd)

Open
Cancel
Cancel All
Bypass

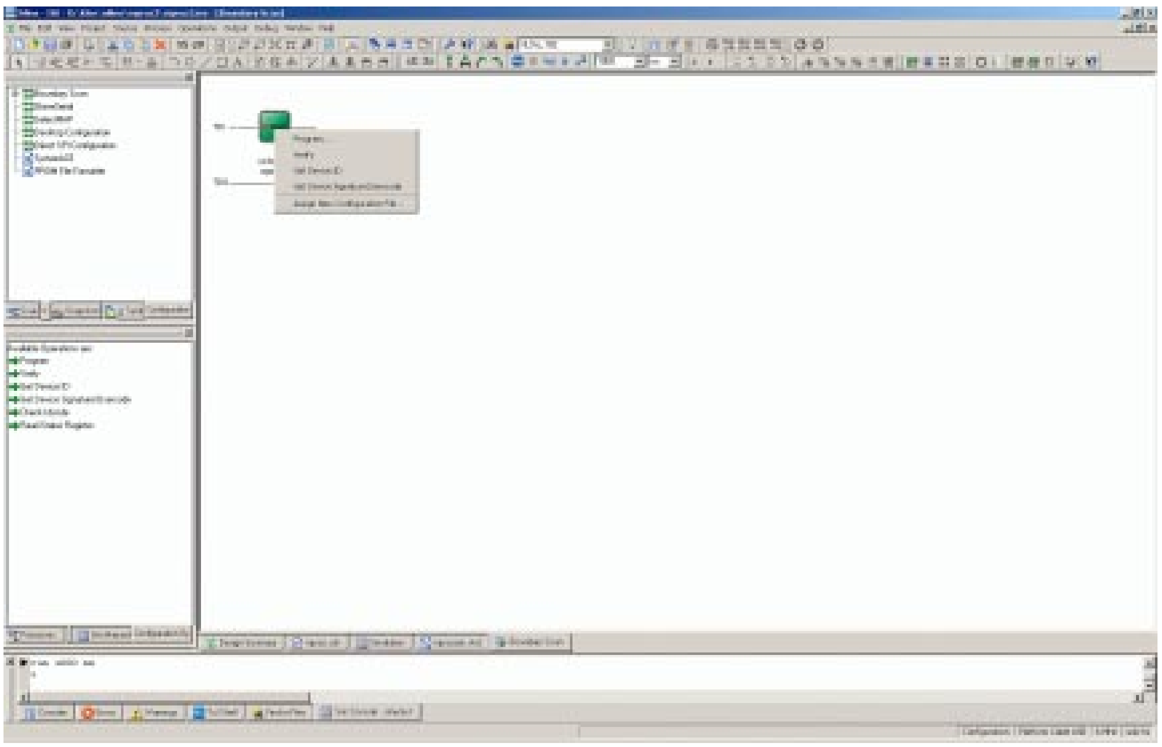
[✓ None]
Enable Programming of SPI Flash Device Attached to this FPGA
Enable Programming of BPI Flash Device Attached to this FPGA

Select "mproz.bit" and click "Open"

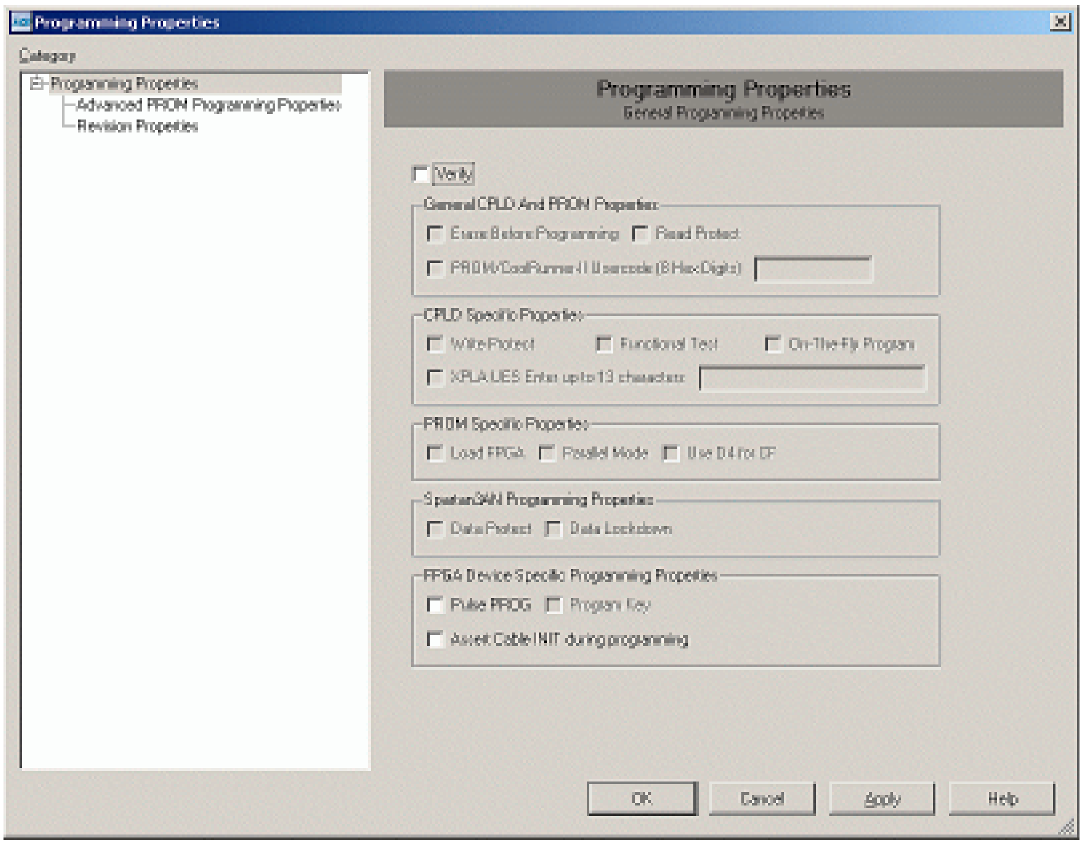


WARNING:iMPACT:2257 - Startup Clock has been changed to 'JtagClk' in the bitstream stored in memory, but the original bitstream file remains unchanged.

Click OK


Program...
Verify
Get Device ID
Get Device Signature/Usercode
Assign New Configuration File...
Program
Verify
Get Device ID
Get Device Signature/Usercode
Check Idcode
Read Status Register

Right click on the FPGA symbol and select "Program..."



Programming Properties
General Programming Properties

☐ Verify

General CPLD And PROM Properties
☐ Erase Before Programming ☐ Read Protect
☐ PROM/CoolRunner-II Usercode (8 Hex Digits)

CPLD Specific Properties
☐ Write Protect ☐ Functional Test ☐ On-The-Fly Program
☐ XPLA UES Enter up to 13 characters

PROM Specific Properties
☐ Load FPGA ☐ Parallel Mode ☐ Use D4 for CF

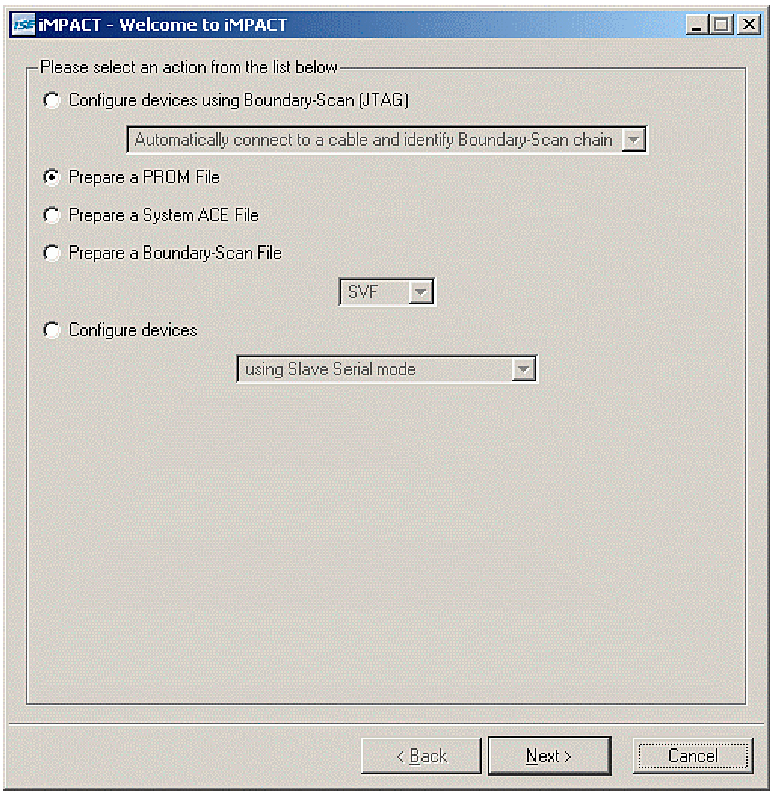
Spartan3AN Programming Properties
☐ Data Protect ☐ Data Lockdown

FPGA Device Specific Programming Properties
☐ Pulse PROG ☐ Program Key
☐ Assert Cable INIT during programming

Uncheck "Verify" (or you must generate the necessary files in a prior step) the and click "OK".

How to program the bit file into the Atmel flash memory on the DARNAW1 board

Close the "Boundary Scan" window and reopen it by a double click "Configure Device (iMPACT)".

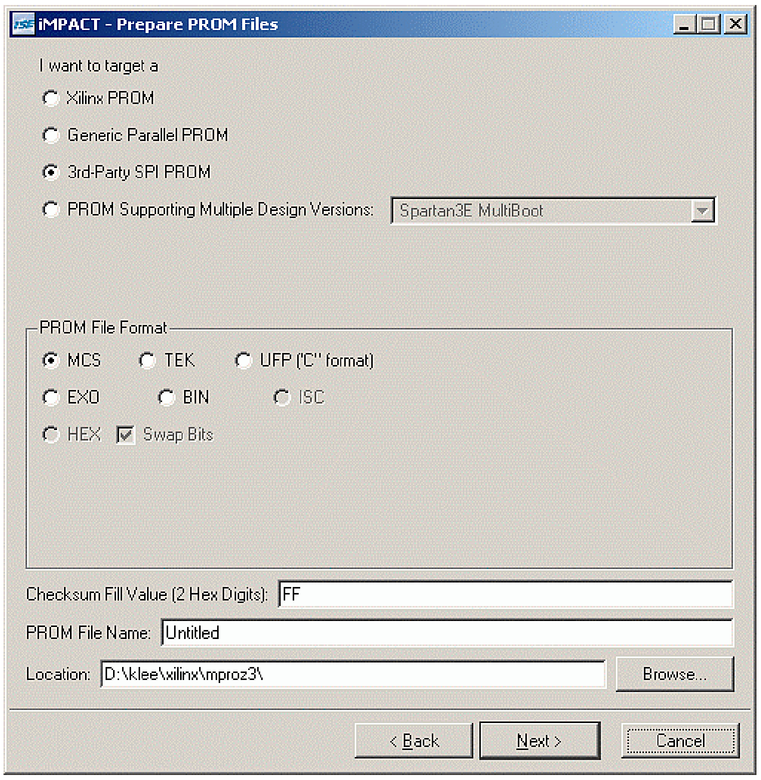


iMPACT - Welcome to iMPACT

Please select an action from the list below

- Configure devices using Boundary-Scan (JTAG) — Automatically connect to a cable and identify Boundary-Scan chain
- [✓ Prepare a PROM File]
- Prepare a System ACE File
- Prepare a Boundary-Scan File — SVF
- Configure devices — using Slave Serial mode

This time choose "Prepare a PROM file" and click "Next".



iMPACT - Prepare PROM Files

I want to target a

- Xilinx PROM
- Generic Parallel PROM
- [✓ 3rd-Party SPI PROM]
- PROM Supporting Multiple Design Versions: Spartan3E MultiBoot

PROM File Format

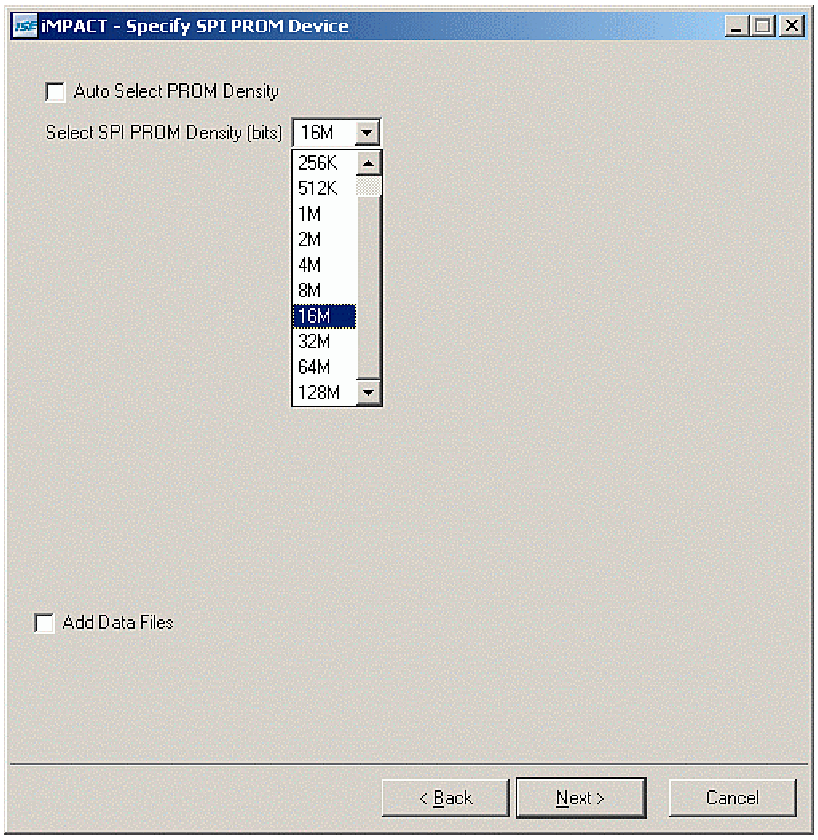
[✓ MCS] TEK UFP ("C" format) EXO BIN ISC HEX [✓ Swap Bits]

Checksum Fill Value (2 Hex Digits): FF

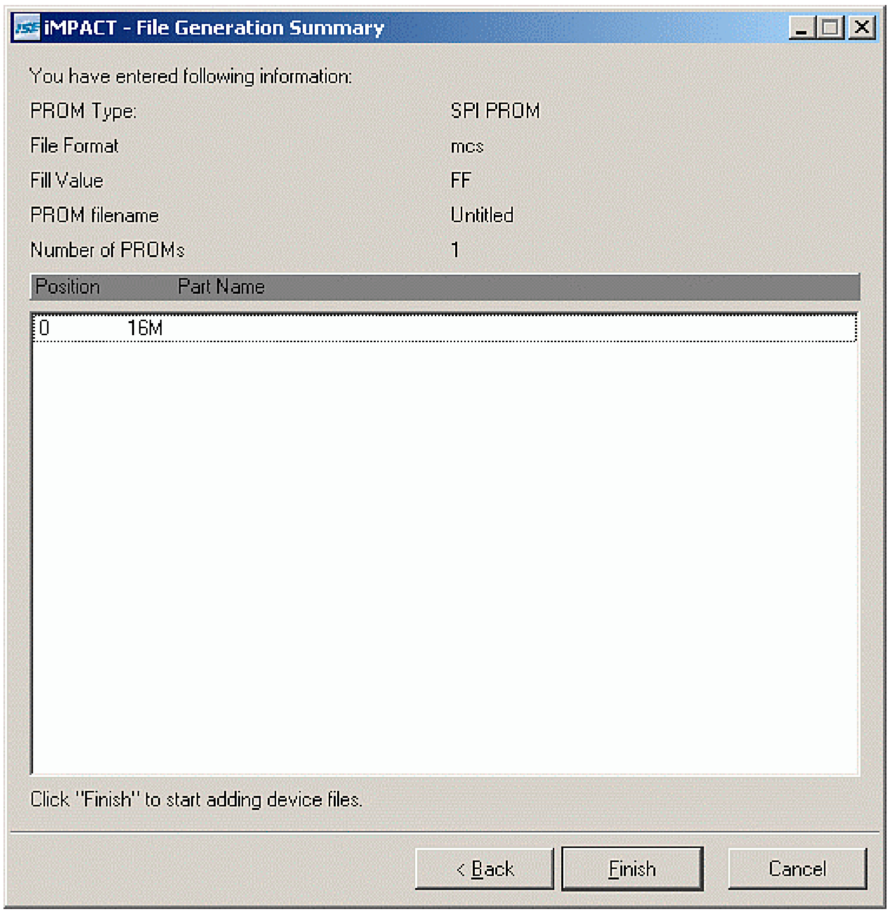
PROM File Name: Untitled

Location: D:\klee\xilinx\mproz3\

Select "3rd-Party SPI PROM" and click "Next". You may also enter a different name than "Untitled".



Select 16M and click "Next".



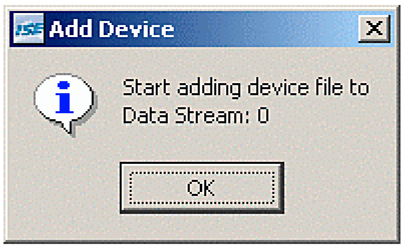
You have entered following information:

PROM Type:	SPI PROM
File Format	mcs
Fill Value	FF
PROM filename	Untitled
Number of PROMs	1

Position	Part Name
0	16M

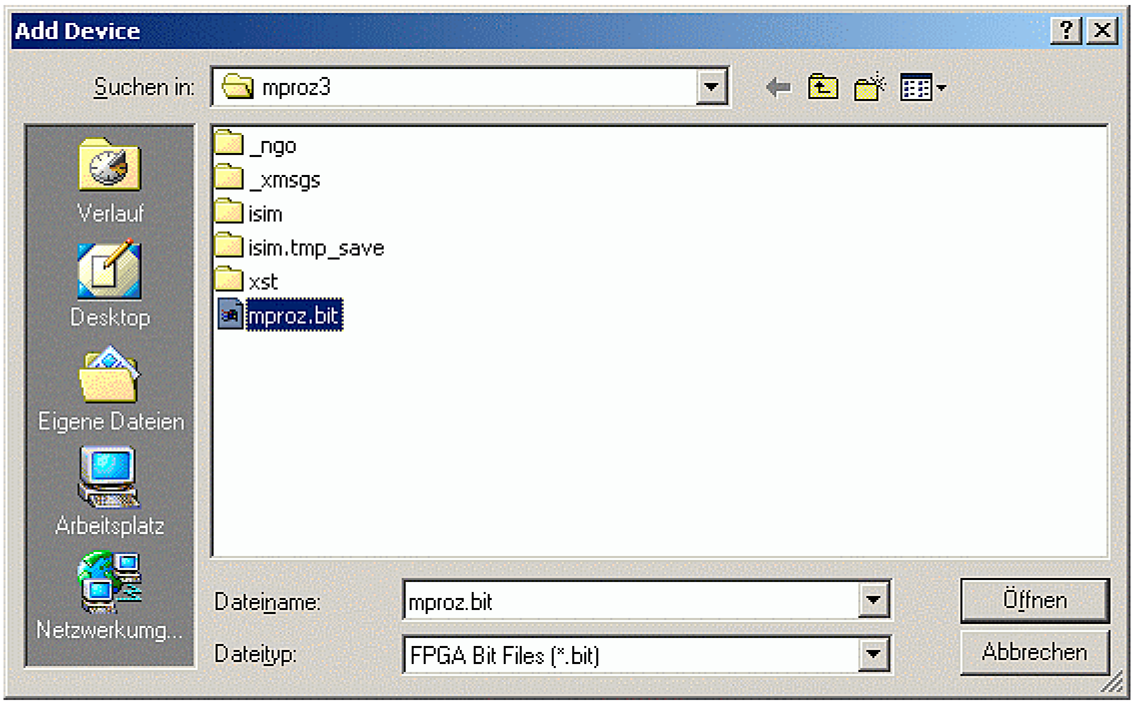
Click "Finish" to start adding device files.

Click "Finish".


Add Device

Start adding device file to
Data Stream: 0

Click "OK".

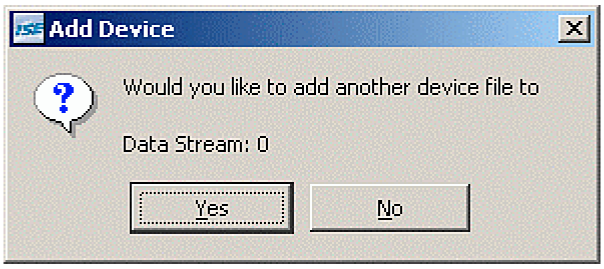

Add Device

Suchen in: mproz3

_ngo
_xmsgs
isim
isim.tmp_save
xst
mproz.bit

Dateiname: mproz.bit
Dateityp: FPGA Bit Files (*.bit)

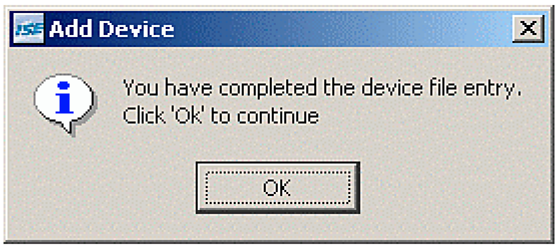
Select "mproz.bit" and click "Open".


Add Device

Would you like to add another device file to

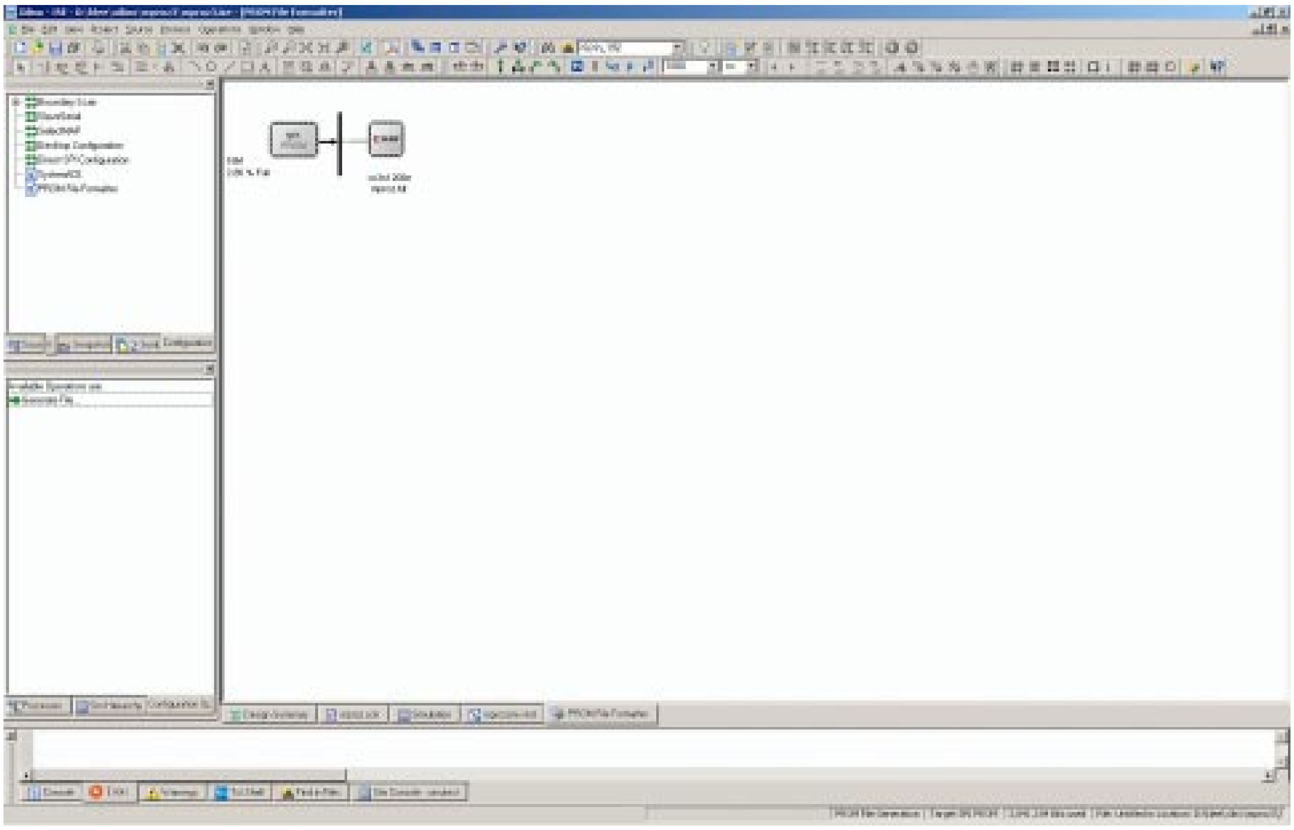
Data Stream: 0

Click "No".

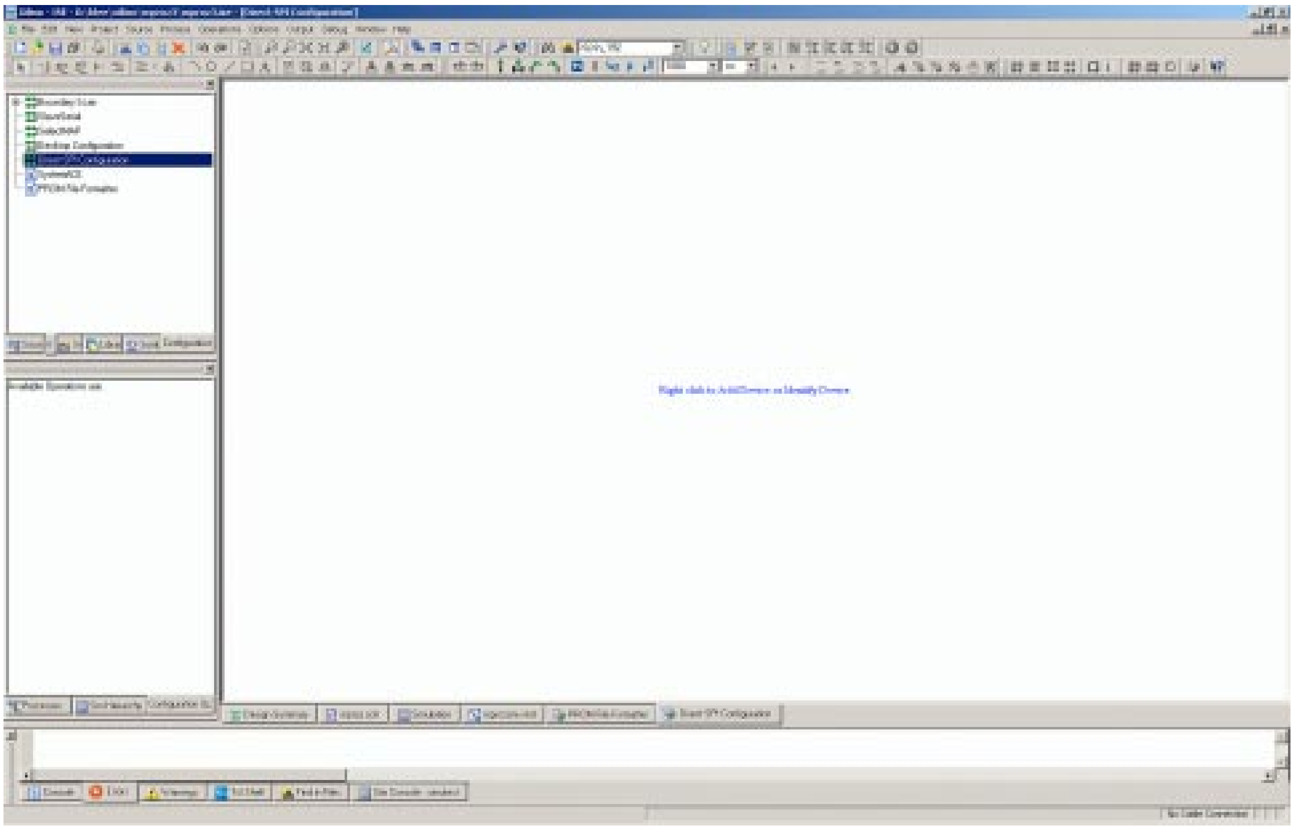

Add Device

You have completed the device file entry.
Click 'Ok' to continue

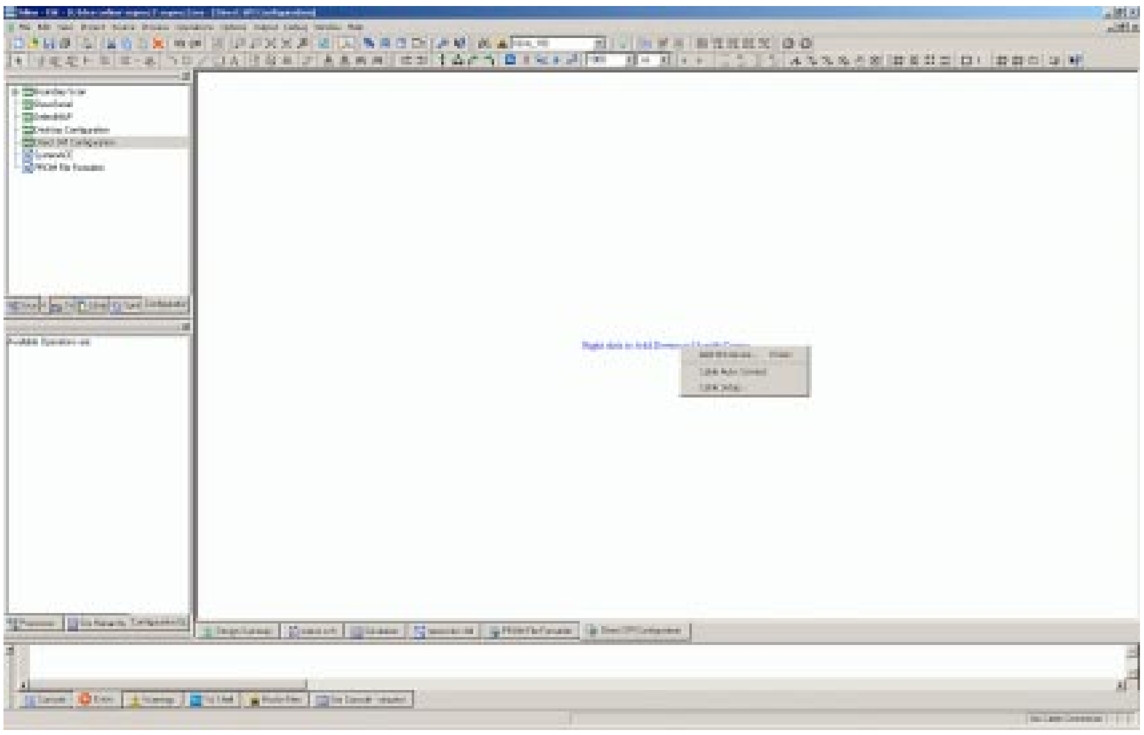
Click "OK".



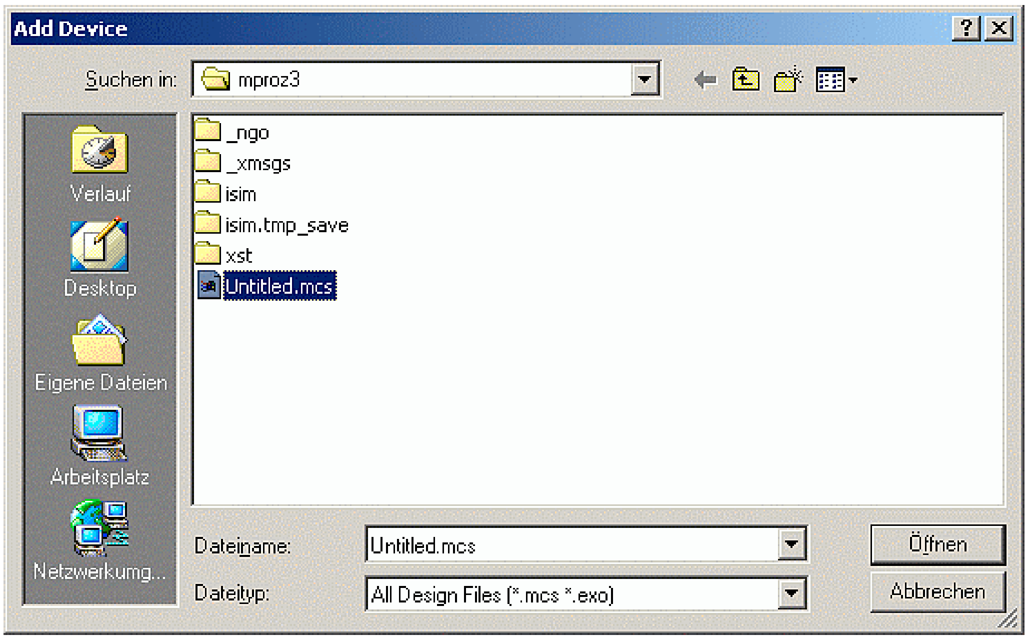
Double click "Generate File".



Double click "Direct SPI configuration".



Right click "Add SPI Device".



Add Device

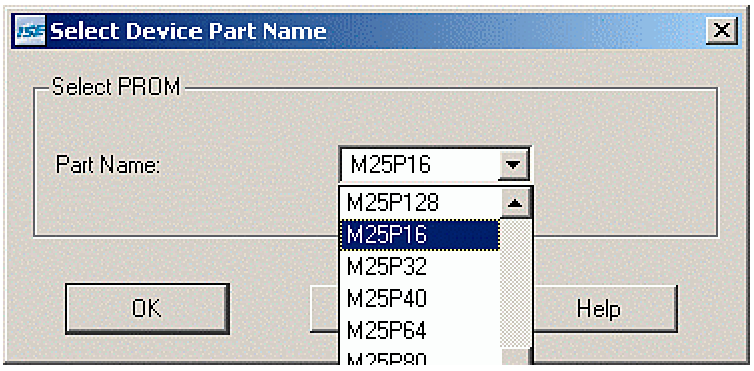
Suchen in: mproz3

_ngo
_xmsgs
isim
isim.tmp_save
xst
Untitled.mcs

Dateiname: Untitled.mcs

Dateityp: All Design Files (*.mcs *.exo)

Select "Untiteled.msc" (or which name you have chosen) and click "Open".



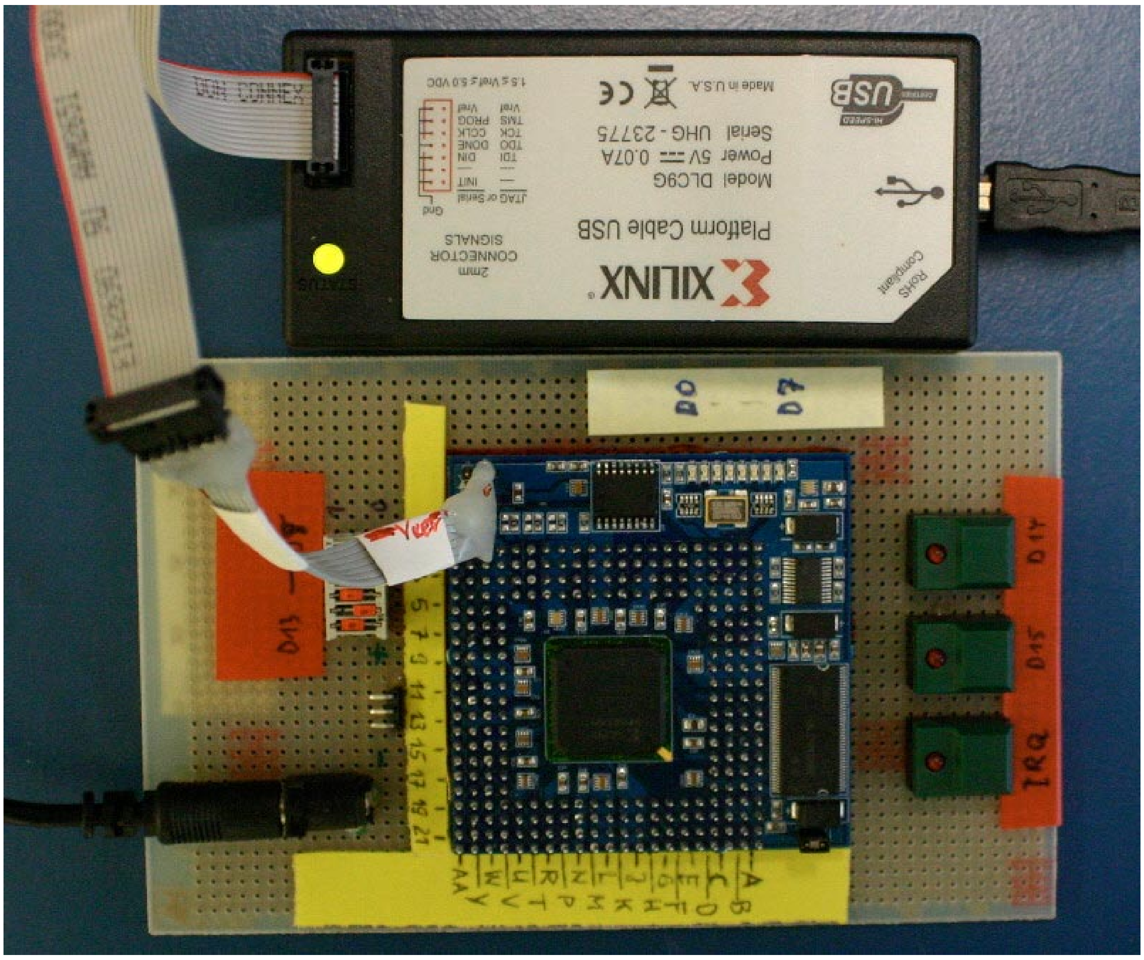
Select Device Part Name

Select PROM

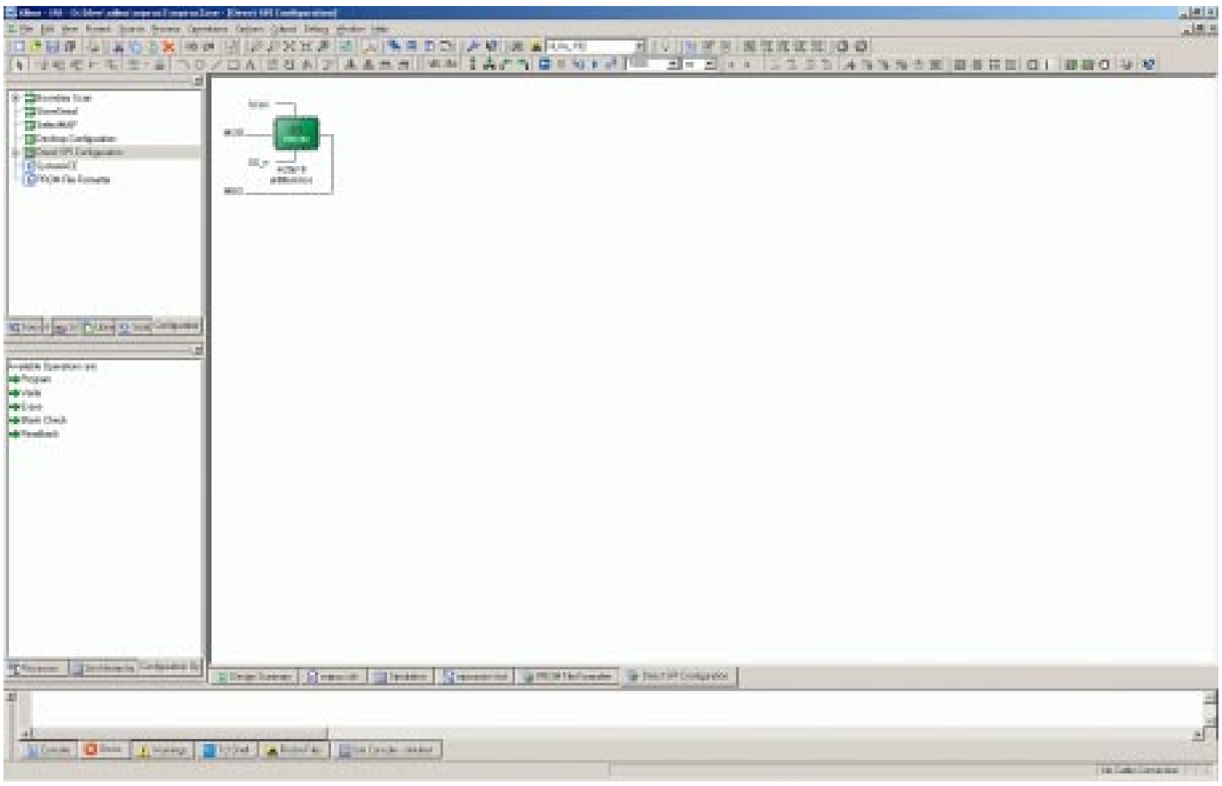
Part Name: M25P16

M25P128
M25P16
M25P32
M25P40
M25P64
M25P80

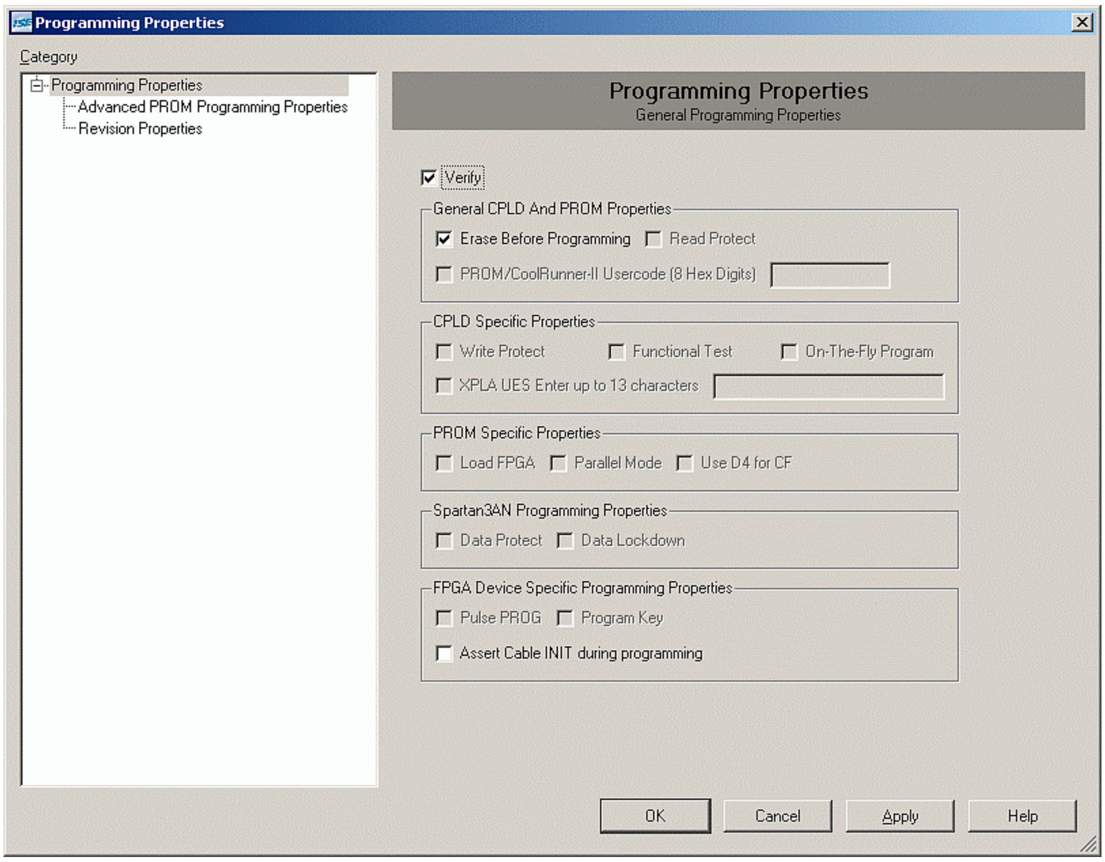
Select "M25P16" and click "OK".



Insert the jumper and connect the download cable to the inner connector.



Right click on the SPI PROM symbol and select "Program..".


Programming Properties

Category

- Programming Properties
  - Advanced PROM Programming Properties
  - Revision Properties

Programming Properties
General Programming Properties

[✓] Verify

General CPLD And PROM Properties
[✓] Erase Before Programming [ ] Read Protect
[ ] PROM/CoolRunner-II Usercode (8 Hex Digits)

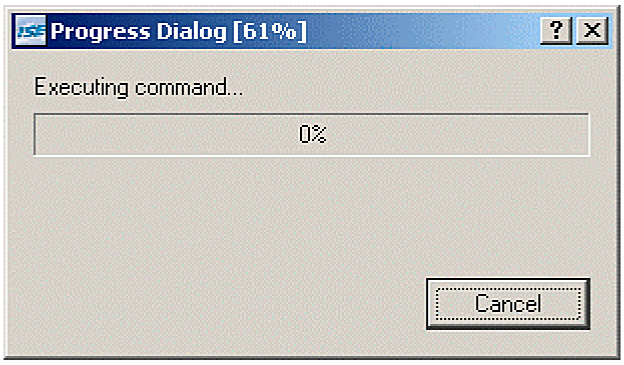
CPLD Specific Properties
[ ] Write Protect [ ] Functional Test [ ] On-The-Fly Program
[ ] XPLA UES Enter up to 13 characters

PROM Specific Properties
[ ] Load FPGA [ ] Parallel Mode [ ] Use D4 for CF

Spartan3AN Programming Properties
[ ] Data Protect [ ] Data Lockdown

FPGA Device Specific Programming Properties
[ ] Pulse PROG [ ] Program Key
[ ] Assert Cable INIT during programming

Click "OK".


Progress Dialog [61%]

Executing command...

0%

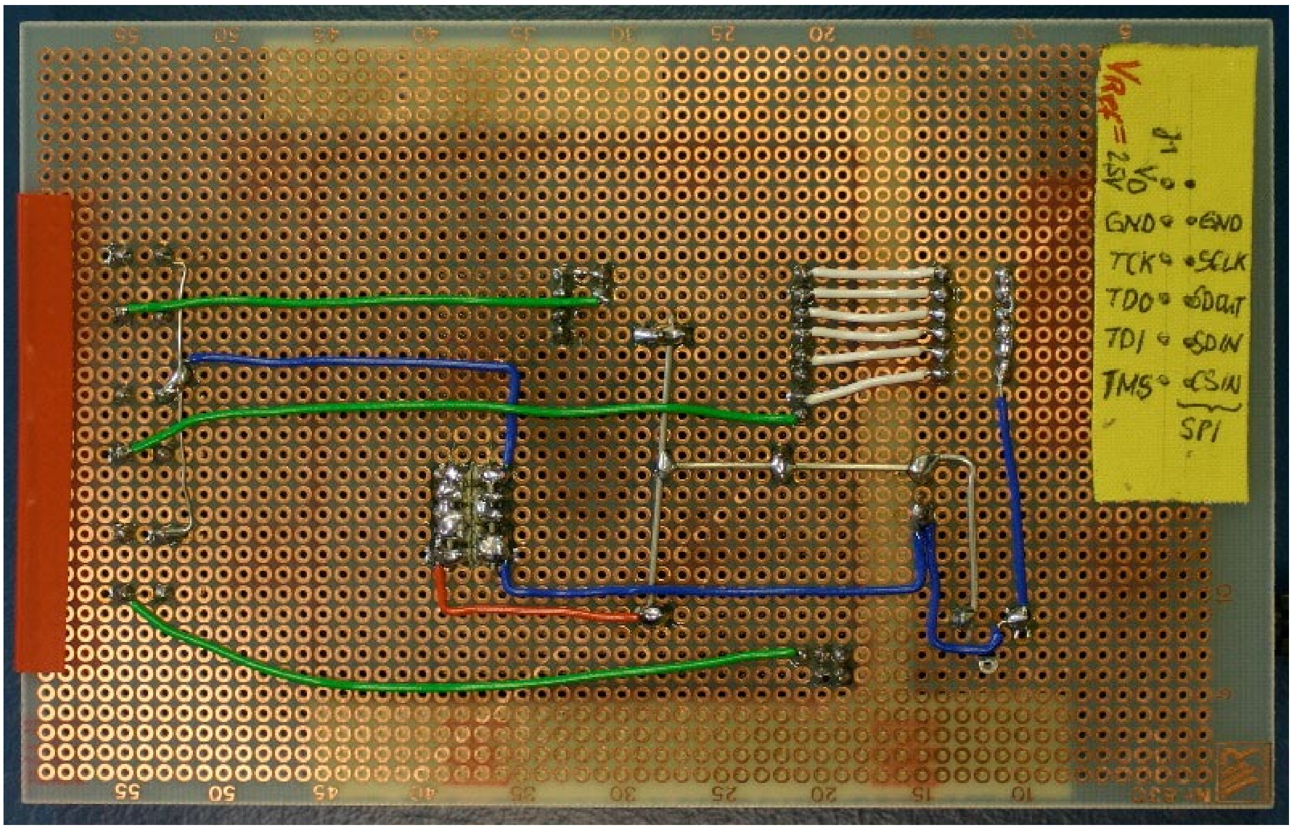
Wait until programming is finished, then remove the download cable and jumper.

5. Modifying the MPROZ program without re-implementing the design

- Modify the source code of LED.MAC and assemble it with XDELA.
- Execute UBIT.BAT. This will create the file mproz_rp.bit.
- Download this file instead of mproz.bit

6. Prototype Board

	GND	C11, C12, C13, C14, C15, D11, D12, D13, D14, D15	
	3.3V	A11, A12, A13, A14, A15, B11, B12, B13, B14, B15	
Switch	d8	W1	
	d9	W2	
	d10	W3	
	d11	W4	
	d12	W5	
	d13	W7	
	d14	J2	
	d15	W8	
	irq	W20	
		J1 - G1	(this is necessary because on the DARNAW1 board LED1 is connect to an input only FPGA pin)



7. Documentation DARNAW1

http://www.enterpoint.co.uk/moelbryn/darnaw1.html