

# **FPGA**

## **IDE Installation and Programming Reference**

**Paul Hamshire**



## Contents

FPGA Boards Overview .....	6
Board chipsets .....	6
Board power.....	6
Development software .....	6
Programming storage.....	7
FPGA Board Details .....	8
DK_START_GW2A-LV18PG256C8I7_V2.0 .....	8
Tang Nano 9K .....	9
Tang Nano 20K.....	10
Tang Primer 25k .....	11
Basys 3 .....	12
Nexys 4 DDR .....	13
Arty S7 50.....	14
Spartan 3e .....	15
Nexys Digital System Builder.....	16
Zybo.....	17
IceBreaker 1.03e .....	18
IceSugar.....	19
IDE Installation .....	20
Vivado 2018.2 .....	20
iCEcube2 .....	23
OSS CAD Suite .....	25
Xilinx ISE 14.7 .....	26
Gowin .....	28
Basys 3 (Vivado 2018.2).....	30
Introduction .....	30
Demo.....	30
Nexys 4 DDR (Vivado 2018.2).....	33
Introduction .....	33
Supplier example .....	33
Demo.....	33
Arty S7 (Vivado 2018.2).....	40
Introduction .....	40
Spartan 3e (Starter Board) (ISE 14.7) .....	41
Introduction .....	41

Supplier example (e3esk-startup).....	41
Building from source.....	43
Demo.....	44
Nexys (Digital System Builder) (ISE 14.7) .....	46
Introduction .....	46
Testing via Adept.....	46
NexysBIST binary .....	47
Building NexysBIST from source .....	48
Supplier example: Nexysdemo .....	50
Demo.....	56
Zybo (Vivado 2018.2).....	57
Introduction .....	57
Demo.....	58
Gowin DK-START (Gowin).....	59
Introduction .....	59
Demo.....	59
Tango Nano 9K (Gowin) .....	61
Demo.....	61
Tang Nano 20K (Gowin).....	64
Introduction .....	64
Demo.....	64
Tang Primer 25k (Gowin) .....	66
Introduction .....	66
Demo.....	66
Lattice ICEBreaker 1.0e (OSS CAD Suite) .....	70
Introduction .....	70
Demo.....	70
IceSugar (OSS CAD Suite).....	73
Introduction .....	73
Demo.....	73
Xilinx PicoBlaze.....	75
Introduction .....	75
KCPSM3 / 6 replacement opbasm .....	75
Build e3esk_startup.....	76
Demo program (KCPSM3 – Spartan 3e) .....	77
Demo program (KCPSM6 – Nexys 4 DDR) .....	79

Demo program (pauloBlaze – Nexys 4 DDR) .....	82
Appendix A: Demo program links .....	86
Example programs.....	86
Appendix B: Useful links .....	87
OSS CAS Suite.....	87
Trial project for an IceBreaker 1.0e .....	87
IceSugar.....	87
Sipeed Nano getting started guides .....	87
Tutorials.....	87
Digilent getting started guides .....	88
Digilent board wikis .....	88
Cable driver locations.....	88
Picoblaze .....	88
Appendix C: Zadig and USB Serial drivers .....	89
Reverting from the Zadig drivers .....	89
Change to another driver .....	89
Delete the Zadig driver .....	90
Seeing how the driver changes when installing Zadig.....	92
Ways to dump all PNP drivers.....	95
USB chips in use.....	95
Useful web sites .....	96

# FPGA Boards Overview

## Board chipsets

Manufacturer	Board	FPGA Manufacturer	FPGA
Gowin	DK_START_GW2A	Gowin	GW2A-LV18PG256
Sipeed	Tang Nano 9k	Gowin	GW1NR-LV9 QN88P
Sipeed	Tang Nano 20k	Gowin	GW2AR-18
Sipeed	Tang Primer 25k	Gowin	GW5A-LV25MG121
Digilent	Basys 3	Xilinx	XC7A35T-1CPG236C
Digilent	Nexsys 4 DDR	Xilinx	XC7A100T-1CSG324C
Digilent	Arty S7	Xilinx	XC7S50-CSGA324
Digilent	Spartan 3e	Xilinx	XC3S500E
Digilent	Nexys	Xilinx	XC3S400
Digilent	Zybo	Xilinx	
1BitSquared	Icebreaker 1.0e	Lattice	iCE40UP5k
MuseLab	IceSugar	Lattice	iCE40UP5k

## Board power

Board	USB	Other	Selection
DK_START_GW2A	N/A	5v centre +	N/A
Tang Nano 9k	USB C	N/A	N/A
Tang Nano 20k	USB C	N/A	N/A
Tang Primer 25k	USB C	N/A	N/A
Basys 3	Micro USB	External	Jumper
Nexsys 4 DDR	Micro USB	5v centre +	Jumper
Arty S7	USB C	5v centre +	Auto
Spartan 3e	N/A	5v centre +	N/A
Nexys	Mini USB	5v centre +	Jumper
Zybo	Mini USB	5v centre +	Jumper
Icebreaker 1.0e	Micro USB	N/A	N/A
IceSugar	USB C	N/A	N/A

## Development software

Board	Vendor		Open Source
	Design	Programmer	
DK_START_GW2A	Gowin IDE	Gowin Programmer	
Tang Nano 9k	Gowin IDE	Gowin Programmer	
Tang Nano 20k	Gowin IDE	Gowin Programmer	
Tang Primer 25k	Gowin IDE	Gowin Programmer	
Basys 3	Vivado 2018.2	n/a	
Nexsys 4 DDR	Vivado 2018.2	n/a	
Arty S7	Vivado 2018.2	n/a	
Spartan 3e	ISE 14.7	iMPACT	
Nexys	ISE 14.7	Adept	
Zybo	Vivado 2018.2	n/a	
Icebreaker 1.0e	iCEcube2 2020.12	Diamond Programmer	oss_cad_suite
IceSugar	iCEcube2 2020.12		oss_cad_suite

## Programming storage

	UBS Drive / SD Card	USB JTAG	Flash	Selection
DK_START_GW2A		Y	Y	Jumper
Tang Nano 9k		Y	Y	
Tang Nano 20k		Y	Y	
Tang Primer 25k		Y	Y	
Basys 3	USB Drive	Y	QSPI	
Nexsys 4 DDR	USB Drive / SD Card	Y	QSPI	Jumper
Arty S7		Y	QSPI	
Spartan 3e		Y	SPI	
Nexys		Y	Y	
Zybo	SD Card	Y	Y	Jumper
Icebreaker 1.0e		Y	Y	
IceSugar		N/A	Virtual disk	

# FPGA Board Details

DK\_START\_GW2A-LV18PG256C8I7\_V2.0



[https://www.gowinsemi.com/en/support/devkits\\_detail/31/](https://www.gowinsemi.com/en/support/devkits_detail/31/)

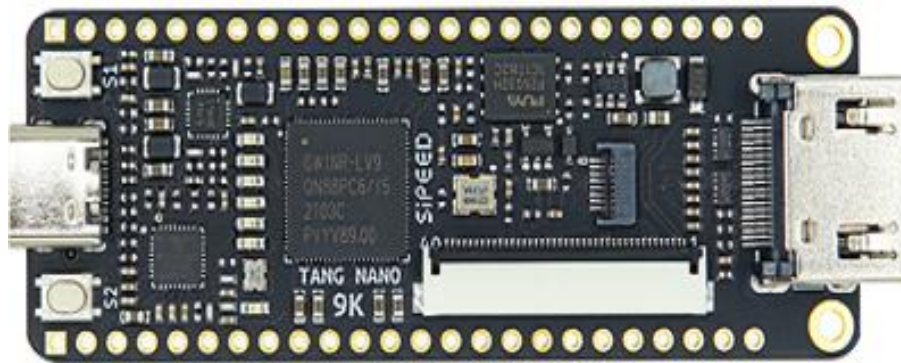
FPGA: GW2A-LV18PG256 C8/17

Datasheet: <https://cdn.gowinsemi.com.cn/DS102E.pdf>

Device	GW2A-18
LUT4s	20,736
Flip-Flops (FFs)	15,552
Shadow SRAM (SSRAM) Capacity (bits)	40K
Block SRAM (BSRAM) Capacity (bits)	828K
Number of BSRAMs	46
Multipliers (18 x 18 Multipliers)	48
Maxiumu PLLs	4
I/O Banks	8
Maxiumum GPIOs	384
Core Voltage	1.0V



## Tang Nano 9K



<https://wiki.sipeed.com/hardware/en/tang/Tang-Nano-9K/Nano-9K.html>

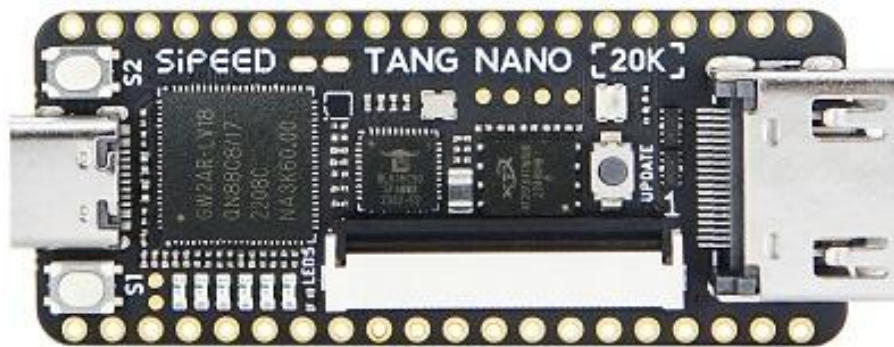
FPGA: GW1NR-LV9 QN88P

Datasheet: <https://cdn.gowinsemi.com.cn/DS117E.pdf>

Device	GW1NR-9
LUT4	8,640
Flip-Flop (REG)	6,480
Shadow SRAM (SSRAM) Capacity (bits)	16K
Block SRAM (BSRAM) Capacity (bits)	468K
Number of BSRAMs	26
User Flash (bits)	608K
SDR SRAM (bits)	64M
PS RAM (bits)	64M (QN88P)
NOR Flash (bits)	-
Multitpliers (18 x 18 Multipliers)	20
PLLs	2
I/O Banks	4
Maximum GPIOs	276
Core voltage (LV Version)	1.2V

Package	Device	Memory Type	Capacity	Width
GN88P	GW1NR-9	PSRAM	64M	16 bits

## Tang Nano 20K



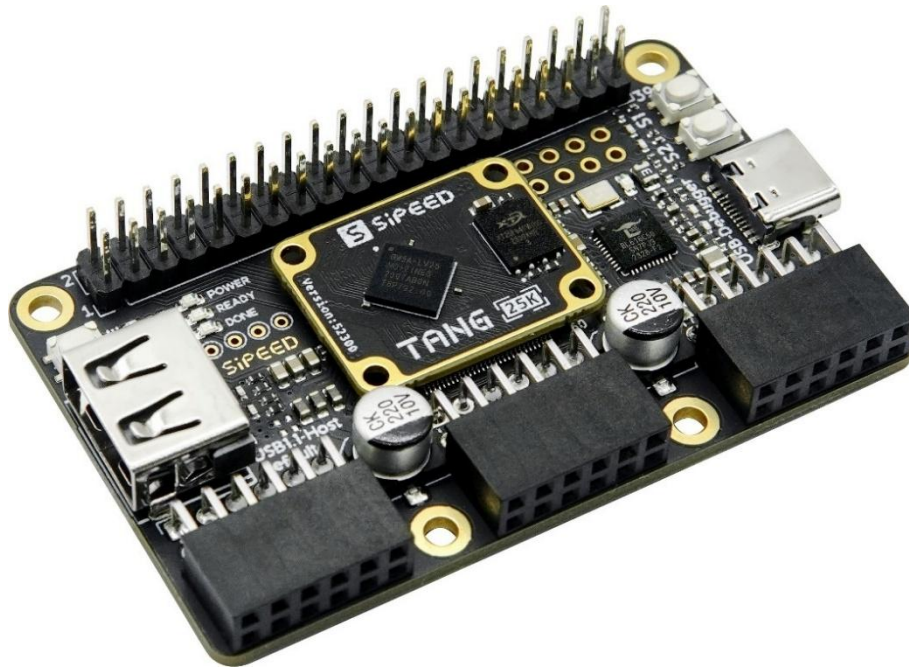
<https://wiki.sipeed.com/hardware/en/tang/tang-nano-20k/nano-20k.html>

FPGA: GW2AR-18 QN88 C8/17

Datasheet: <https://cdn.gowinsemi.com.cn/DS226E.pdf>

Device	GW2AR-18
LUT4s	20,736
Flip-Flops (FFs)	15,552
Shadow SRAM (SSRAM) Capacity (bits)	40K
Block SRAM (BSRAM) Capacity (bits)	828K
Number of BSRAMs	46
SDR SDRAM (bits)	64M
Multipliers (18 x 18 Multipliers)	48
Maximum PLLs	4
I/O Banks	8
Maximum GPIOs	384
Core Voltage	1.0V

## Tang Primer 25k



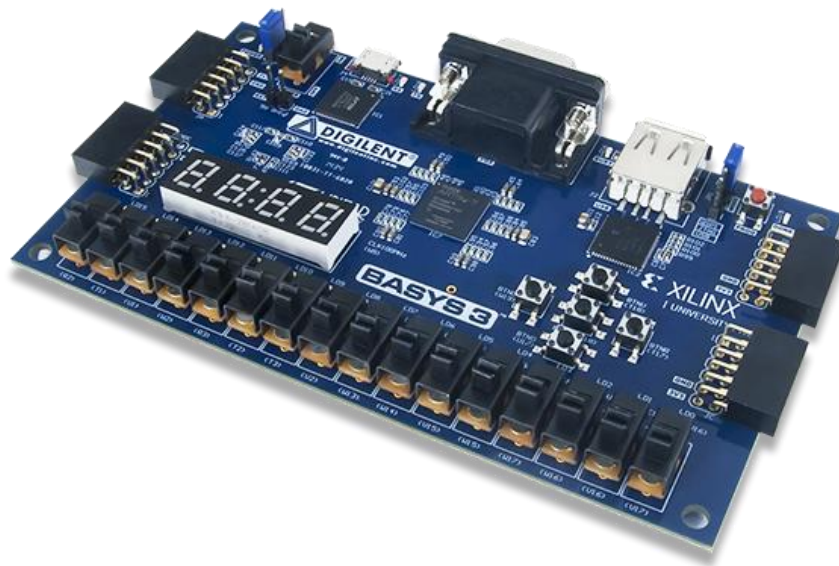
<https://wiki.sipeed.com/hardware/en/tang/tang-primer-25k/primer-25k.html>

FPGA: GW5A-LV25MG121

Datasheet: <https://cdn.gowinsemi.com.cn/DS1103E.pdf>

Device	GW5A-25
LUT4	23,040
Flip-Flop (REG)	23,040
Shadow SRAM (SSRAM) Capacity (bits)	180
Block SRAM (BSRAM) Capacity (bits)	1008
Number of BSRAMs	56
DSP (27 bit x 18 bit)	28
Maximum phase locked loop (PLLs)	6
Global Clocks	16
High-speed clocks	16
LCDS Gbps	1.25
DDR3 Mbps	1,066
MIPI D-PHY hard core	2.5 Gbps (RX/TX) 4 data lanes 1 clock lan
MIP C-PHY hardcore	-
ADC	1
Number of GPIO banks	8
Maximum number of GPIOs	239
Core voltage	0.9V / 1.0 V / 1.2V

## Basys 3



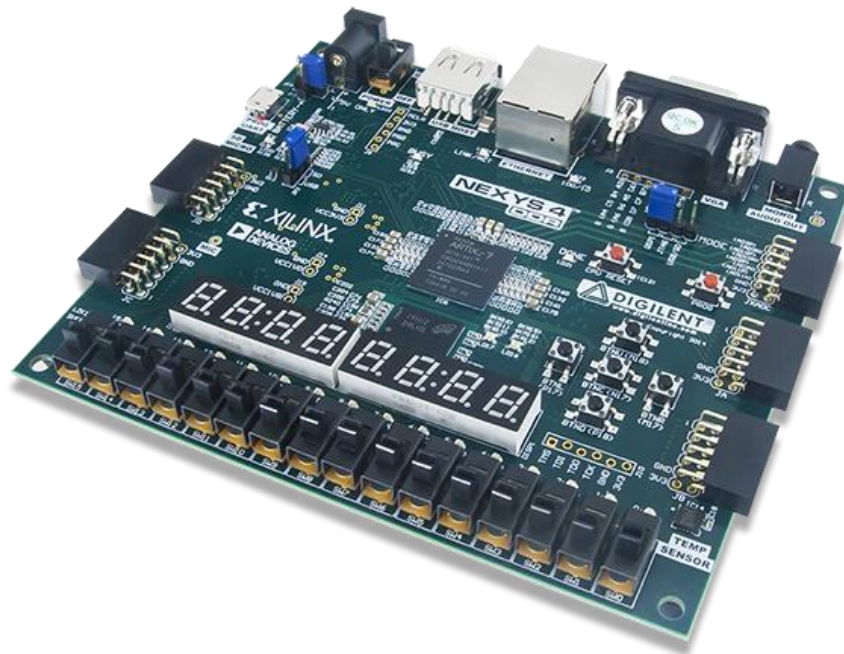
<https://digilent.com/reference/programmable-logic/basys-3/start>

FPGA: XC7A35T-1CPG236C

Datasheet: [https://docs.amd.com/v/u/en-US/ds180\\_7Series\\_Overview](https://docs.amd.com/v/u/en-US/ds180_7Series_Overview)

Device	XC7A35T
Logic cells	33,280
Configurable Logic Blocks (CLBs)	
Slices	5,200
Max distributed RAM (Kb)	400
DSP48E1 Slices	90
Block Ram Blocks	
18Kb	100
36Kb	50
Max (Kb)	1,800
CMTs	5
PCIe	1
GTPs	2
XADC Blocks	1
Total I/O Banks	5
Max User I/O	106

## Nexys 4 DDR



Web: <https://digilent.com/reference/programmable-logic/nexys-4-ddr/start>

Manual: <https://digilent.com/reference/programmable-logic/nexys-4-ddr/reference-manual>

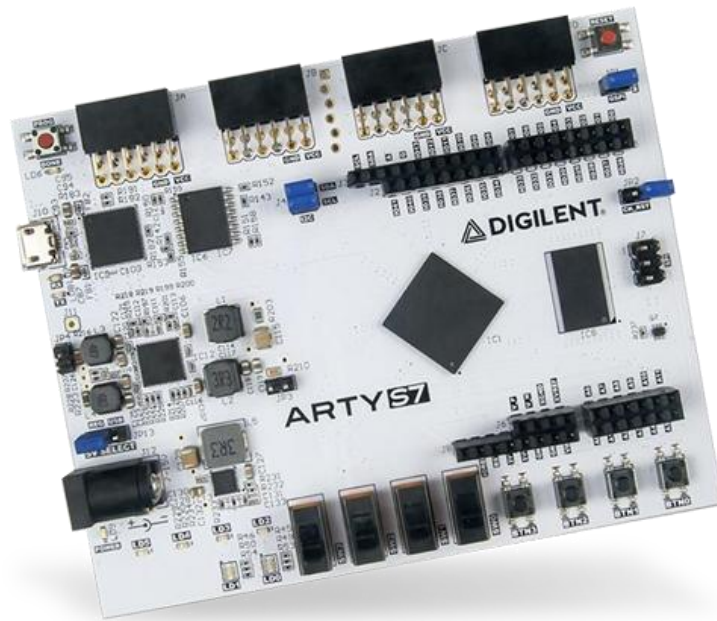
FPGA: XC7A35T-1CSG324C

Datasheet: [https://docs.amd.com/v/u/en-US/ds180\\_7Series\\_Overview](https://docs.amd.com/v/u/en-US/ds180_7Series_Overview)

Device	XC7A35T
Logic cells	33,280
Configurable Logic Blocks (CLBs)	
Slices	5,200
Max distributed RAM (Kb)	400
DSP48E1 Slices	90
Block Ram Blocks	
18Kb	100
36Kb	50
Max (Kb)	1,800
CMTs	5
PCIe	1
GTPs	0
XADC Blocks	1
Total I/O Banks	5
Max User I/O	210



## Arty S7 50



Web: <https://digilent.com/shop/arty-s7-spartan-7-fpga-development-board>

Manual: <https://digilent.com/reference/programmable-logic/arty-s7/reference-manual>

FPGA: Xilinx XC7S50-CSGA324

FPGA Datasheet: [https://docs.amd.com/v/u/en-US/ds180\\_7Series\\_Overview](https://docs.amd.com/v/u/en-US/ds180_7Series_Overview)

Device	XC7S50-CSGA324
Logic cells	52,160
Configurable Logic Blocks (CLBs)	
Slices	8,150
Max distributed RAM (Kb)	600
DSP48E1 Slices	120
Block Ram Blocks	
18Kb	150
36Kb	75
Max (Kb)	2,700
CMTs	5
PCIe	0
GTPs	0
XADC Blocks	1
Total I/O Banks	5
Max User I/O	250

## Spartan 3e



Web: <https://digilent.com/reference/programmable-logic/spartan-3e/start>

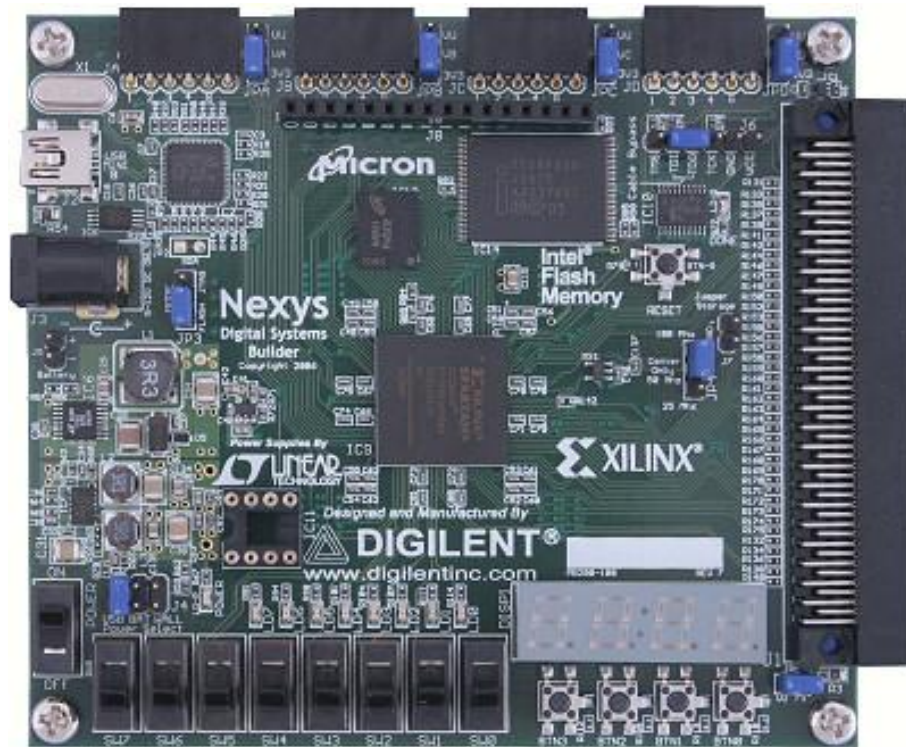
Manual: <https://digilent.com/reference/programmable-logic/spartan-3e/reference-manual>

FPGA: Xilinx XC3S500E

Datasheet: <https://docs.amd.com/v/u/en-US/ds312>

Device	XC3S500E
System gates	500K
Equivalent logic cells	10,476
CLB Array	
Rows	46
Columns	34
Total CLBs	1,164
Total slices	4,656
Distributed RAM bits	73K
Block RAM bits	360K
Dedicated multipliers	20
DCMs	4
Max user I/O	232
Max differential I/O pairs	92

## Nexys Digital System Builder



Web: <https://digilent.com/reference/nexys/nexys>

Manual: [https://digilent.com/reference/\\_media/nexys/nexys\\_rm.pdf](https://digilent.com/reference/_media/nexys/nexys_rm.pdf)

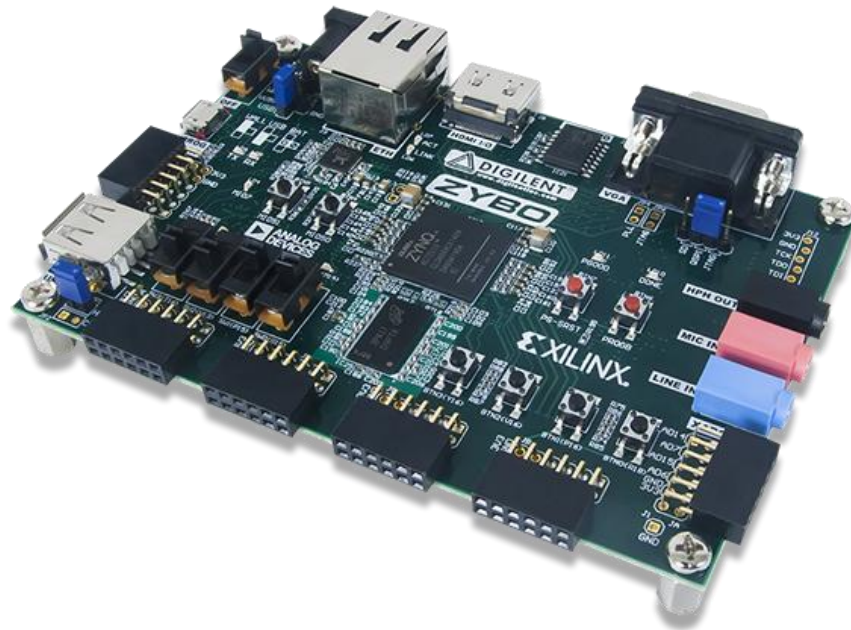
FPGA: Xilinx XC3S1000

FPGA Datasheet: <https://docs.amd.com/v/u/en-US/ds099>

Device	XC3S1000
System gates	1M
Equivalent logic cells	17,280
CLB Array	
Rows	48
Columns	40
Total CLBs	1,920
Total slices	7,680
Distributed RAM bits	120K
Block RAM bits	432K
Dedicated multipliers	24
DCMs	4
Max user I/O	391
Max differential I/O pairs	175



## Zybo



Web: <https://digilent.com/reference/programmable-logic/zybo/start>

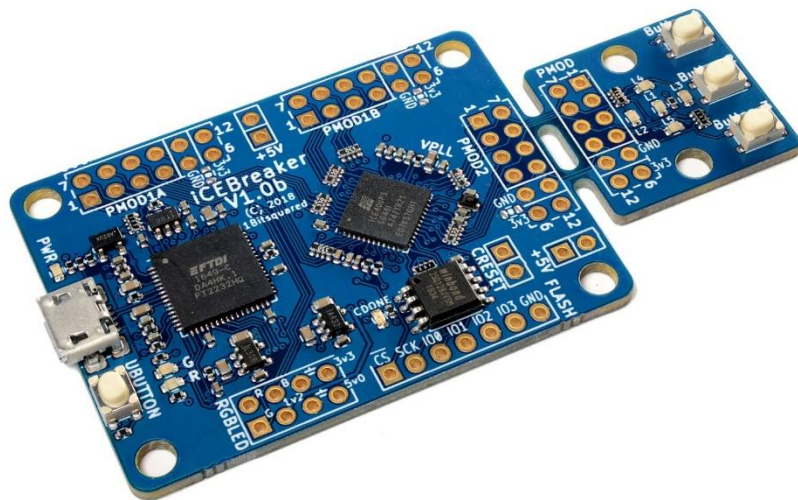
Manual: <https://digilent.com/reference/programmable-logic/zybo/reference-manual>

FPGA: XC7Z010-1CLG400C

FPGA Datasheet: <https://docs.amd.com/v/u/en-US/ds190-Zynq-7000-Overview>

Device	XC7Z010-1CLG400C
Xilinx 7 Series Equivalent	Artix 7
Programmable Logic Cells	28K
Look Up Tables (LUTs)	17,600
Flip-flops	35,200
Block RAM (# 36Kb blocks)	2.1Mb (60)
DSP Slices (18x25 MACCs)	80
Analog Mixed Signal (AMS / XADC)	2x12 bit MSPS ADCs

## IceBreaker 1.03e



Web: <https://www.crowdsupply.com/1bitsquared/icebreaker-fpga>

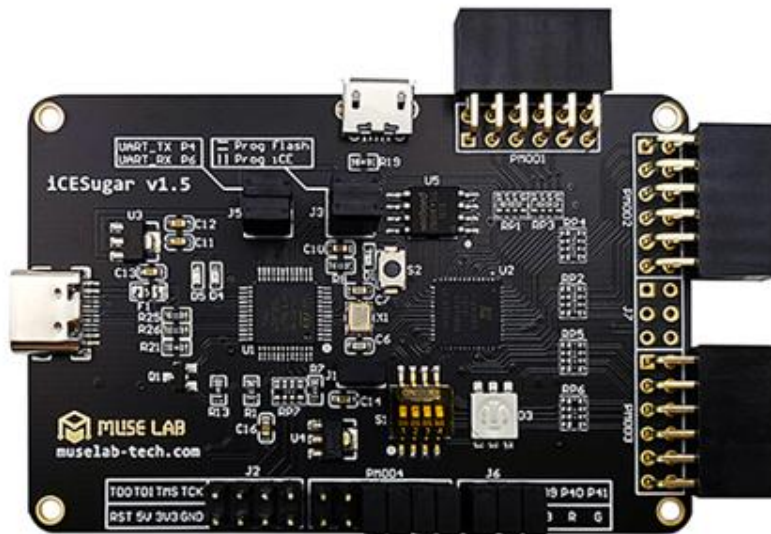
Web: <https://github.com/icebreaker-fpga>

FPGA: Lattice iCE40UP5k FPGA

<https://www.latticesemi.com/en/Products/FPGAandCPLD/iCE40UltraPlus>

Device	iCE40UP5k
Density LUTs	5280
NVCM	Yes
Static Current (uA)	75
EBR RAM (kbits)	120
SPRAM (kbits)	1024
PLL	1
I2C Core	2
SPI Core	2
Oscillator (10 kHz)	1
Oscillator (48 MHz)	1
24 mA Drive	3
500 mA Drive	-
16 x 16 Multiply & 32 bit Accumulator Blocks	8
PWM	Yes

## IceSugar



Web: <https://github.com/wuxx/icesugar>

FPGA: Lattice iCE40UP5k FPGA

<https://www.latticesemi.com/en/Products/FPGAandCPLD/iCE40UltraPlus>

Device	iCE40UP5k
Density LUTs	5280
NVCM	Yes
Static Current (uA)	75
EBR RAM (kbits)	120
SPRAM (kbits)	1024
PLL	1
I2C Core	2
SPI Core	2
Oscillator (10 kHz)	1
Oscillator (48 MHz)	1
24 mA Drive	3
500 mA Drive	-
16 x 16 Multiply & 32 bit Accumulator Blocks	8
PWM	Yes

# IDE Installation

## Vivado 2018.2

### Useful information

This is a smaller install than Vivado 2024.2 so is preferred, even if it is older and may have less capability.

This link has excellent instructions on installing and licensing:

<https://allaboutfpga.com/download-and-installation-of-vivado-design-suite-for-working-with-edge-artix-7-and-zynq-fpga-kits/>

Guide: <https://digilent.com/reference/programmable-logic/guides/installing-vivado-and-vitis>

### Files

Download page:

<https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html>

Vivado Design Suite - HLx Editions - 2018.2 Full Product Installation

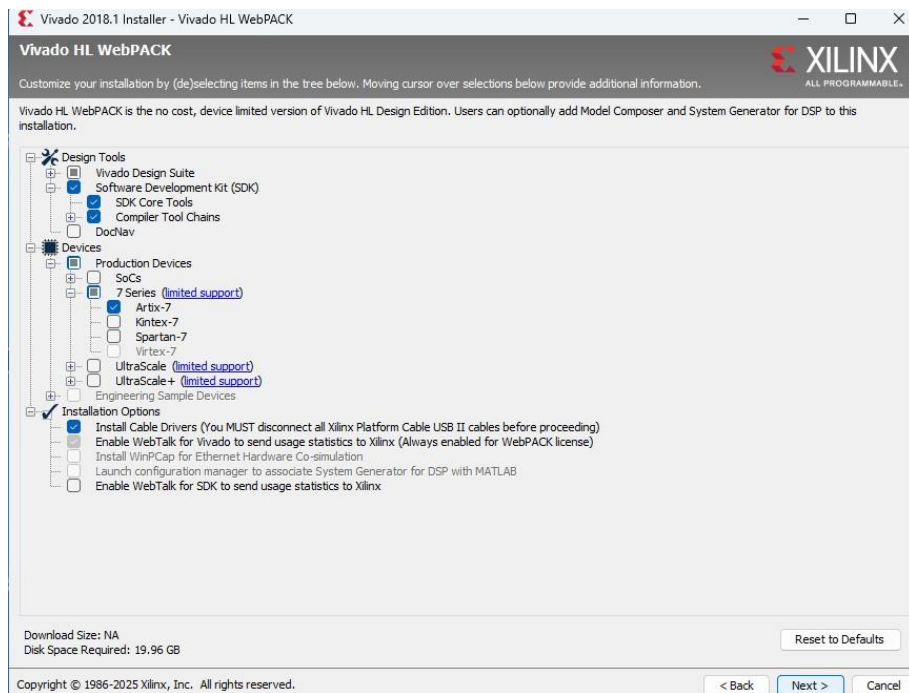
File: Vivado HLx 2018.2: All OS installer Single-File Download (TAR/GZIP - 17.11 GB)

### Install process

Extract from tar

Extract from zip

Run **setup.exe**

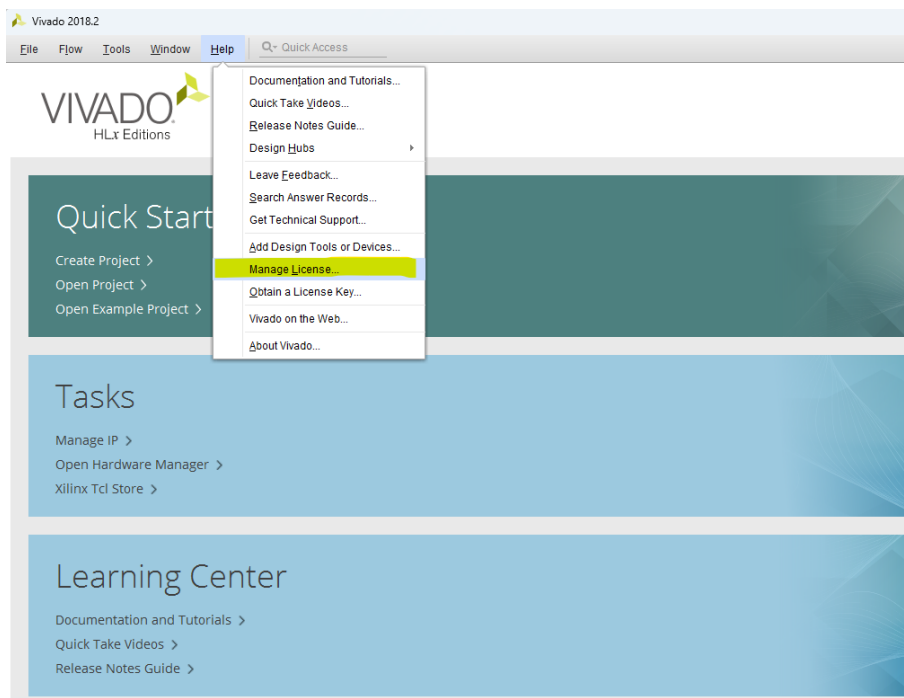


## License

Create an account on the AMD website

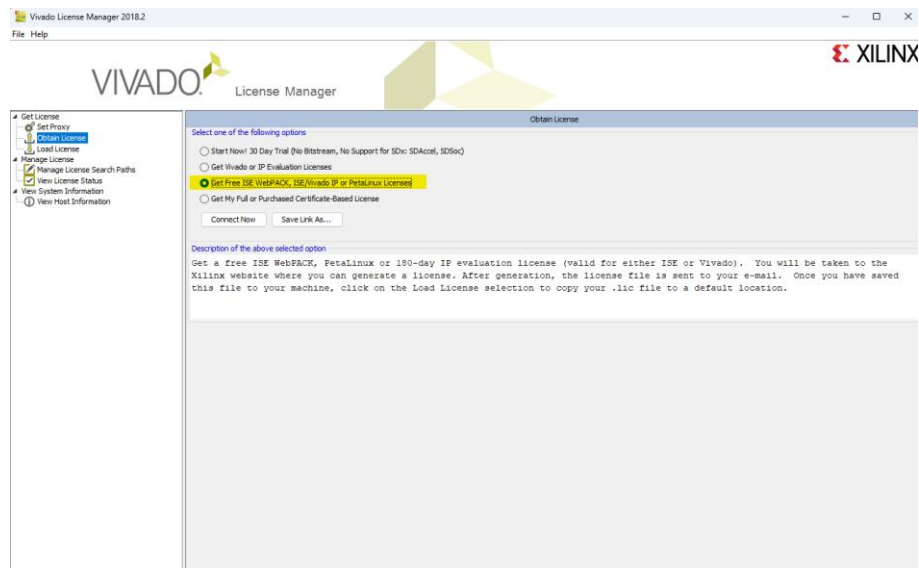
<https://login.amd.com/>

In Vivado, select “Help”, “Manage Licenses”



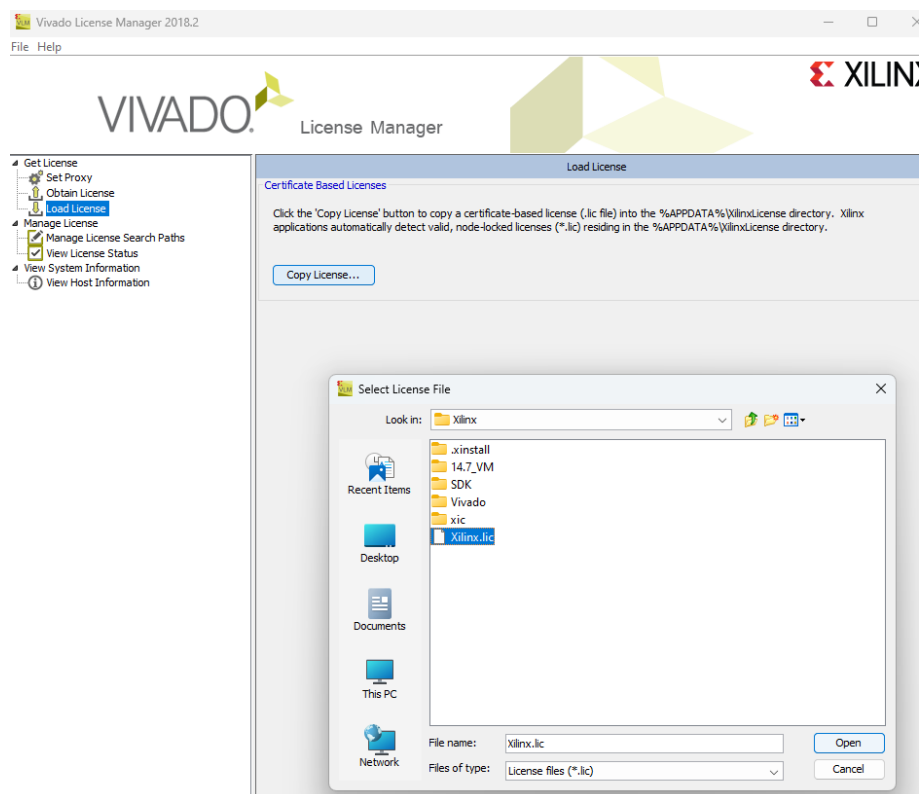
This opens the license manager.

Select “Obtain License” and then “Get Free ISE WebPACK, ISE/Vivado IP or PetaLinux Licenses”



Then a license key will be emailed to you. Copy to a location of your choice.

Final step is to load the license



## iCEcube2

### Useful information

You will need iCECube 2 and Diamond programmer, or OSS CAS Suite.

### Files

#### iCEcube2

<https://www.latticesemi.com/icecube2>

#### Diamond programmer

<https://www.latticesemi.com/diamond>

Select on: Programmer Standalone 3.14 64-bit for Windows

### License

You will need a 'makers community' licence

<https://www.latticesemi.com/icecube2>

Email [lic\\_admn@latticesemi.com](mailto:lic_admn@latticesemi.com) and with the title 'New iCEcube2 License Request'

Then you will receive form to complete for your MAC address

<add your message here if any>  
If you are Hobbyist/Enthusiast, please fill out this form.  
MAC address for license generation:  
License Type (Node-locked or Floating)  
iCE Device Used:  
Country:  
  
For Start-up or community educators that will provide license to their users, please fill out this form.  
Company name:  
Company address:  
Your Product Name:  
No. of Licenses Needed:

Like this

If you are Hobbyist/Enthusiast, please fill out this form.  
MAC address for license generation: XX-XX-XX-XX-XX-XX  
License Type (Node-locked or Floating) Node locked  
iCE Device Used: iCE40UP5K  
Country: United Kingdom

You will get a serial number via email

<https://www.latticesemi.com/Support/Licensing/DiamondAndiCEcube2SoftwareLicensing/DiamondSub>

Paste the serial number and a .dat file will be emailed

**Subscription Licensing Page**

As a customer you can create your own license request using the form below.  
Please make sure all information is correct then click "Save". For triple redundant server, enter the 3 NIC separated by commas

Save Cancel : Software Registration

Registration Details << >>

ID:

First Name:

Last Name:

Company Name:

Email:

Serial Number:

Please click to confirm the serial number is valid before submitting  
**Check SN**

Required for renewal license only

Previous Serial Number:

Previous Expiration Date:

Number of Registrations:

Product:

Sale License Type:

License File Type:

(MAC Address: Ethernet Adapter Local Area Connection) - Do not use any spaces or dashes.

NIC:

Once you have the .dat file, save it somewhere

Then run the License Manager:

c:\lsc\iCEcube2.2020.12\LicenseSetup.exe

To import the license file



# OSS CAD Suite

## Useful information

You will need iCECube 2 and Diamond programmer, or OSS CAS Suite.

## Files

<https://github.com/YosysHQ/oss-cad-suite-build>

<https://github.com/YosysHQ/oss-cad-suite-build/releases>

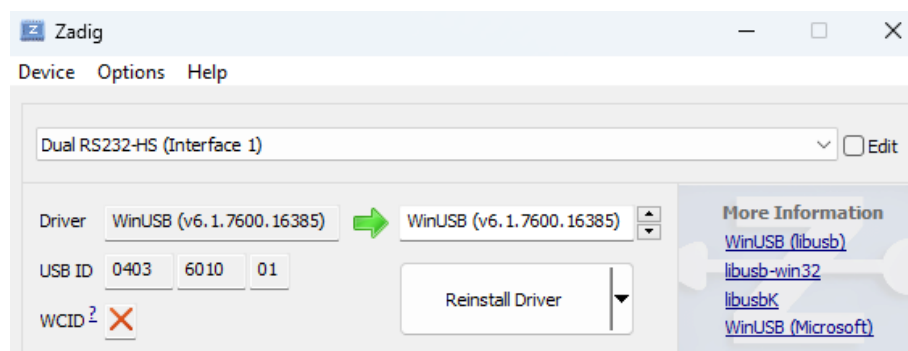
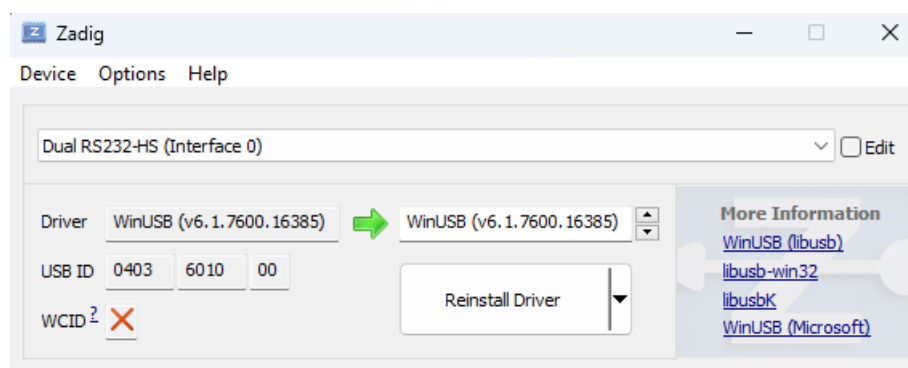
Download: [oss-cad-suite-windows-x64-20250315.exe](#)

Extract to a location you choose (c:\oss-cad-suite)

## Zadig

IceBreaker and ICESugar use USB ID 0403 6010 00

Use Zadig (<https://zadig.akeo.ie/>) to change the driver for the FTDI link to WinUSB (Inteface 0 and Interface 1) (from Serial CDC)



## Xilinx ISE 14.7

### Useful information

ISE 14.7 on Windows 11 runs in Oracle Virtualbox. You need the image file and Virtualbox.

For a Nexys board you also need the Adept programmer

Spartan 3e uses the iMPACT programmer in the Virtualbox image.

### Files

#### ISE 14.7

<https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive-ise.html>

#### Oracle Virtualbox

<https://www.virtualbox.org/wiki/Downloads>

#### Digilent Adept (for Nexys Digital System Builder)

<https://digilent.com/shop/software/digilent-adept/>

Adept System v2.27.9-2 for Windows (Retired)

### Installation

Install VirtualBox

Unzip Xilinx\_ISE\_14.7\_Win10\_14.7\_VM\_0213\_1.zip

Edit this file

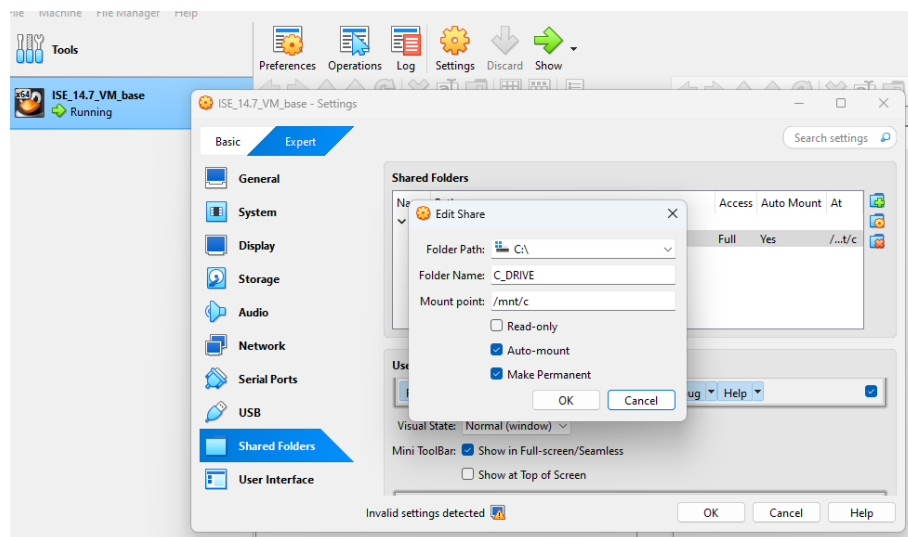
```
ISE
14.7\Xilinx_ISE_14.7_Win10_14.7_VM_0213_1\bin\validate_virtualization_enabled.ps1
```

Add this at the start:

```
# My changes
                                "Virtualization is enabled"
                                $host.SetShouldExit(0)
                                exit
# My changes end
```

Run **xsetup.exe**

In VM, select Settings/Shared Folders and set up a file mount to c:



## License

No license file is required for the VM image

# Gowin

## Useful information

The Gowin software after v1.9.9 gets reported as containing a virus.








For these boards, v1.9.9 works fine

Sipeed wiki: <https://wiki.sipeed.com/hardware/en/tang/Tang-Nano-Doc/install-the-ide.html>

## Files

Download page: [https://www.gowinsemi.com/en/support/download\\_eda/](https://www.gowinsemi.com/en/support/download_eda/)

### For v1.9.9

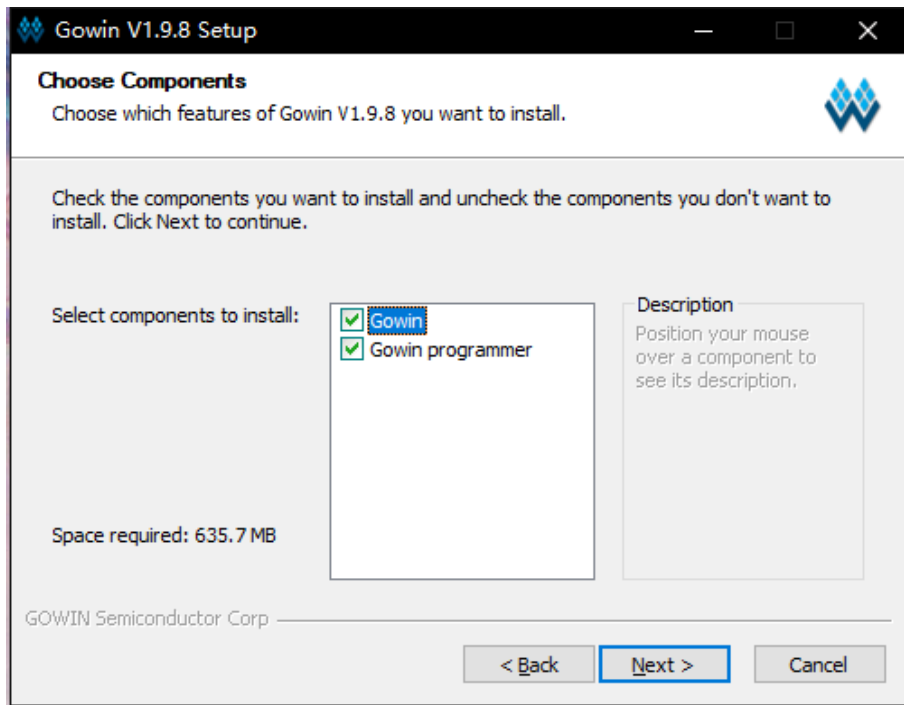
Software for Windows	Software for Linux
Windows Software	
Gowin V1.9.11.01 Education (Windows x86)	 Click Here
Gowin V1.9.11.01 (Windows x64)	 Click Here
Gowin V1.9.11 (Windows x64)	 Click Here
Gowin patch01 V1.9.10.03 (Windows x64)	 Download
Gowin V1.9.10.03 Software patch01 Release Note	 Download
Gowin_V1.9.9 (Windows x64) Achieved TUV certification for ISO26262 & IEC61508	 Click Here
Gowin V1.9.9 Software Release Note	 Download

Or via this link:

[https://www.gowinsemi.com/en/document/main/database/14/?support\\_search=&type=category&order=ASC&page=4](https://www.gowinsemi.com/en/document/main/database/14/?support_search=&type=category&order=ASC&page=4)

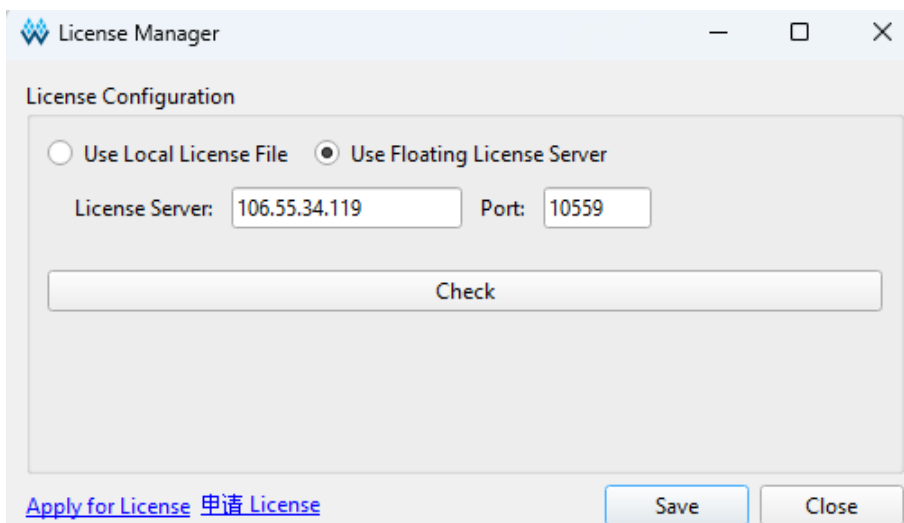
## Installation

Install Gowin and Gowin Programmer



## License

As per the Sipeed wiki, these details work for a floating license



ip: 106.55.34.119

port: 10559

# Basys 3 (Vivado 2018.2)

## Introduction

Based on code here: <https://www.fpga4student.com/2017/09/seven-segment-led-display-controller-basys3-fpga.html>

## Demo

Create Project

In the wizard, select the folder

Select RTL project

**Verilog file: digits.v**

```
module Seven_Segment(
    input clock, // 100 MHz clock
    input reset,
    output reg [3:0] anode,
    output reg [6:0] led
);

reg [26:0] counter;
reg [15:0] number;
reg [3:0] digit;
reg [19:0] refresh_counter;
wire [1:0] which_digit;

always @(posedge clock or posedge reset)
begin
    if (reset == 1)
        begin
            counter <= 0;
            number <= 0;
        end
    else
        begin
            if (counter >= 99999999) begin
                number <= number + 1;
                counter <= 0;
            end
            else
                counter <= counter + 1;
        end
    end

always @(posedge clock or posedge reset)
begin
    if (reset == 1)
        refresh_counter <= 0;
    else
        refresh_counter <= refresh_counter + 1;
    end

assign which_digit = refresh_counter[19:18];

always @(*)
begin
    case (which_digit)
        2'b00:
            begin
```

```

        anode = 4'b0111;
        digit = number[15:12];
    end
    2'b01:
    begin
        anode = 4'b1011;
        digit = number[11:8];
    end
    2'b10:
    begin
        anode = 4'b1101;
        digit = number[7:4];
    end
    2'b11:
    begin
        anode = 4'b1110;
        digit = number[3:0];
    end
endcase
end

always @(*)
begin
    case(digit)
        4'b0000: led = 7'b0000001; // "0"
        4'b0001: led = 7'b1001111; // "1"
        4'b0010: led = 7'b0010010; // "2"
        4'b0011: led = 7'b0000110; // "3"
        4'b0100: led = 7'b1001100; // "4"
        4'b0101: led = 7'b0100100; // "5"
        4'b0110: led = 7'b0100000; // "6"
        4'b0111: led = 7'b0001111; // "7"
        4'b1000: led = 7'b0000000; // "8"
        4'b1001: led = 7'b0000100; // "9"
        4'b1010: led = 7'b0001000; // "A"
        4'b1011: led = 7'b1100000; // "B"
        4'b1100: led = 7'b0110001; // "C"
        4'b1101: led = 7'b1000010; // "D"
        4'b1110: led = 7'b0110000; // "E"
        4'b1111: led = 7'b0111000; // "F"
        default: led = 7'b1111111; // " "
    endcase
end
endmodule

//      --A--
//      F      B
//      --G--
//      E      C
//      --D--
//
//      ABCDEFG
//      7'b0000001

```

### Constraints file: digits.cst

```

# Clock signal
set_property -dict { PACKAGE_PIN W5 IOSTANDARD LVCMOS33 } [get_ports clock]
set_property -dict { PACKAGE_PIN R2 IOSTANDARD LVCMOS33 } [get_ports reset]

#seven-segment LED display
set_property -dict { PACKAGE_PIN W7 IOSTANDARD LVCMOS33 } [get_ports {led[6]}]
set_property -dict { PACKAGE_PIN W6 IOSTANDARD LVCMOS33 } [get_ports {led[5]}]
set_property -dict { PACKAGE_PIN U8 IOSTANDARD LVCMOS33 } [get_ports {led[4]}]

```

```

set_property -dict { PACKAGE_PIN V8 IOSTANDARD LVCMOS33 } [get_ports {led[3]}]
set_property -dict { PACKAGE_PIN U5 IOSTANDARD LVCMOS33 } [get_ports {led[2]}]
set_property -dict { PACKAGE_PIN V5 IOSTANDARD LVCMOS33 } [get_ports {led[1]}]
set_property -dict { PACKAGE_PIN U7 IOSTANDARD LVCMOS33 } [get_ports {led[0]}]
set_property -dict { PACKAGE_PIN U2 IOSTANDARD LVCMOS33 } [get_ports {anode[0]}]
set_property -dict { PACKAGE_PIN U4 IOSTANDARD LVCMOS33 } [get_ports {anode[1]}]
set_property -dict { PACKAGE_PIN V4 IOSTANDARD LVCMOS33 } [get_ports {anode[2]}]
set_property -dict { PACKAGE_PIN W4 IOSTANDARD LVCMOS33 } [get_ports {anode[3]}]

```

Select part

Family: Artix-7

Package: cpg236

Speed grade: -1

Select: xc7a35tcpg236-1

Create project

Run Synthesis

Run Implementation

Generate bitstream

Under “Open hardware manager”

Open target – auto connect

Program device – xc7a35t\_0

Click Program



# Nexys 4 DDR (Vivado 2018.2)

## Introduction

<https://digilent.com/reference/programmable-logic/nexys-4-ddr/start>

## Supplier example

Example program here:

<https://github.com/Digilent/Nexys-4-DDR-OOB>

Build

Download zip file (release)

<https://github.com/Digilent/Nexys-4-DDR-OOB/releases/tag/v2018.2-1>

Extract

Open vivado\_proj\Nexys-4-DDR-OOB.xpr

Product family:	Artix-7
Project part:	Nexys4 DDR (xc7a100tcsg324-1)

Device: xc7a100tcsg324-1

Run Synthesis

Run Implementation

Generate bitstream

## Demo

Another example program here:

[https://digilent.com/reference/\\_media/nexys4-ddr/nexys4-ddr\\_sw\\_demo.zip](https://digilent.com/reference/_media/nexys4-ddr/nexys4-ddr_sw_demo.zip)

Slightly amended below to use a clock input.

**sw\_led.v**

```
`timescale 1ns / 1ps

module sw_led (
    input          CLK100MHZ,
    input [15:0]    SW,        // Slide switch inputs
    output reg [15:0] LED      // LED outputs
);

    always @(posedge CLK100MHZ)
```

```

begin
    LED <= SW;
end
endmodule

```

#### nexys-4-DDR.xdc

```

## Clock signal
set_property -dict { PACKAGE_PIN E3      IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ }];
#IO_L12P_T1_MRCC_35 Sch=clk100mhz
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports
{CLK100MHZ}];

## Switches
set_property -dict { PACKAGE_PIN J15      IOSTANDARD LVCMOS33 } [get_ports { SW[0] }];
set_property -dict { PACKAGE_PIN L16      IOSTANDARD LVCMOS33 } [get_ports { SW[1] }];
set_property -dict { PACKAGE_PIN M13      IOSTANDARD LVCMOS33 } [get_ports { SW[2] }];
set_property -dict { PACKAGE_PIN R15      IOSTANDARD LVCMOS33 } [get_ports { SW[3] }];
set_property -dict { PACKAGE_PIN R17      IOSTANDARD LVCMOS33 } [get_ports { SW[4] }];
set_property -dict { PACKAGE_PIN T18      IOSTANDARD LVCMOS33 } [get_ports { SW[5] }];
set_property -dict { PACKAGE_PIN U18      IOSTANDARD LVCMOS33 } [get_ports { SW[6] }];
set_property -dict { PACKAGE_PIN R13      IOSTANDARD LVCMOS33 } [get_ports { SW[7] }];
set_property -dict { PACKAGE_PIN T8       IOSTANDARD LVCMOS18 } [get_ports { SW[8] }];
set_property -dict { PACKAGE_PIN U8       IOSTANDARD LVCMOS18 } [get_ports { SW[9] }];
set_property -dict { PACKAGE_PIN R16      IOSTANDARD LVCMOS33 } [get_ports { SW[10] }];
set_property -dict { PACKAGE_PIN T13      IOSTANDARD LVCMOS33 } [get_ports { SW[11] }];
set_property -dict { PACKAGE_PIN H6       IOSTANDARD LVCMOS33 } [get_ports { SW[12] }];
set_property -dict { PACKAGE_PIN U12      IOSTANDARD LVCMOS33 } [get_ports { SW[13] }];
set_property -dict { PACKAGE_PIN U11      IOSTANDARD LVCMOS33 } [get_ports { SW[14] }];
set_property -dict { PACKAGE_PIN V10      IOSTANDARD LVCMOS33 } [get_ports { SW[15] }];

## LEDs
set_property -dict { PACKAGE_PIN H17      IOSTANDARD LVCMOS33 } [get_ports { LED[0] }];
set_property -dict { PACKAGE_PIN K15      IOSTANDARD LVCMOS33 } [get_ports { LED[1] }];
set_property -dict { PACKAGE_PIN J13      IOSTANDARD LVCMOS33 } [get_ports { LED[2] }];
set_property -dict { PACKAGE_PIN N14      IOSTANDARD LVCMOS33 } [get_ports { LED[3] }];
set_property -dict { PACKAGE_PIN R18      IOSTANDARD LVCMOS33 } [get_ports { LED[4] }];
set_property -dict { PACKAGE_PIN V17      IOSTANDARD LVCMOS33 } [get_ports { LED[5] }];
set_property -dict { PACKAGE_PIN U17      IOSTANDARD LVCMOS33 } [get_ports { LED[6] }];
set_property -dict { PACKAGE_PIN U16      IOSTANDARD LVCMOS33 } [get_ports { LED[7] }];
set_property -dict { PACKAGE_PIN V16      IOSTANDARD LVCMOS33 } [get_ports { LED[8] }];
set_property -dict { PACKAGE_PIN T15      IOSTANDARD LVCMOS33 } [get_ports { LED[9] }];
set_property -dict { PACKAGE_PIN U14      IOSTANDARD LVCMOS33 } [get_ports { LED[10] }];
set_property -dict { PACKAGE_PIN T16      IOSTANDARD LVCMOS33 } [get_ports { LED[11] }];
set_property -dict { PACKAGE_PIN V15      IOSTANDARD LVCMOS33 } [get_ports { LED[12] }];
set_property -dict { PACKAGE_PIN V14      IOSTANDARD LVCMOS33 } [get_ports { LED[13] }];
set_property -dict { PACKAGE_PIN V12      IOSTANDARD LVCMOS33 } [get_ports { LED[14] }];
set_property -dict { PACKAGE_PIN V11      IOSTANDARD LVCMOS33 } [get_ports { LED[15] }];

```

Create new project

Select directory location

Project Type is RTL project

Add sources / Add Files select **sw\_led.v**

Add constraints / Add Files select **nexys-4-DDR.xdc**

Dafult part is xc7a100tcs9324-1

Product family: Artix-7  
Project part: Nexys4 DDR (xc7a100tcsq324-1)

Run Synthesis

Run Implementation

Generate bitstream

To program via JTAG

The screenshot displays the Xilinx IDE interface. On the left is the 'Flow Navigator' pane with a tree view containing sections like PROJECT MANAGER, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, SYNTHESIS, IMPLEMENTATION, and PROGRAM AND DEBUG. The 'PROGRAM AND DEBUG' section is selected, showing options like 'Generate Bitstream' and 'Open Hardware Manager'. The main workspace is divided into three panes. The top pane is the 'HARDWARE MANAGER' for 'localhost/xilinx\_tcf/Digilent/210292742279...', showing a message 'There are no debug cores.' and a table of hardware components. The middle pane is the 'Properties' pane, currently empty with the text 'Select an object to see properties'. The bottom pane is the 'Tcl Console', showing a series of INFO messages from Labtools regarding hardware target opening and device refresh.

**Flow Navigator**

- PROJECT MANAGER
  - Settings
  - Add Sources
  - Language Templates
  - IP Catalog
- IP INTEGRATOR
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- SIMULATION
  - Run Simulation
- RTL ANALYSIS
  - Open Elaborated Design
- SYNTHESIS
  - Run Synthesis
  - Open Synthesized Design
- IMPLEMENTATION
  - Run Implementation
  - Open Implemented Design
- PROGRAM AND DEBUG
  - Generate Bitstream
  - Open Hardware Manager
    - Open Target
    - Program Device
    - Add Configuration Memory Device

**HARDWARE MANAGER - localhost/xilinx\_tcf/Digilent/210292742279/**

There are no debug cores. [Program device](#) [Refresh device](#)

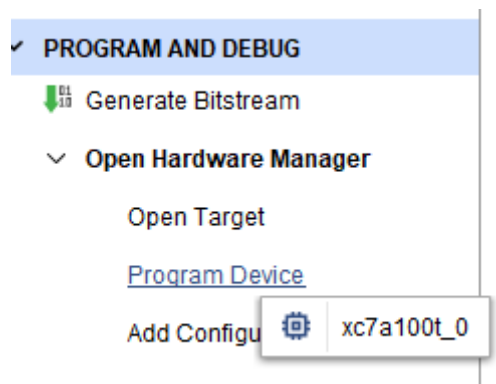
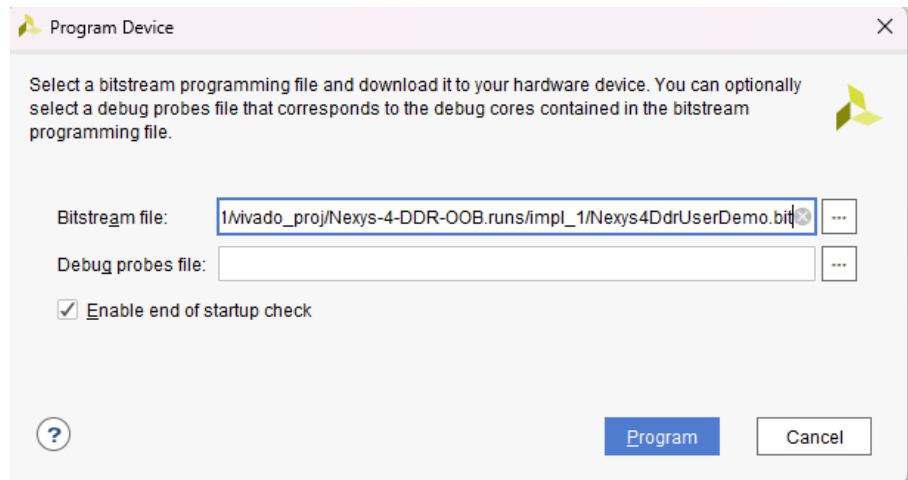
Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/2102927422...	Open
xc7a100t_0 (1)	Not programr
XADC (System Monitor)	

**Properties**

Select an object to see properties

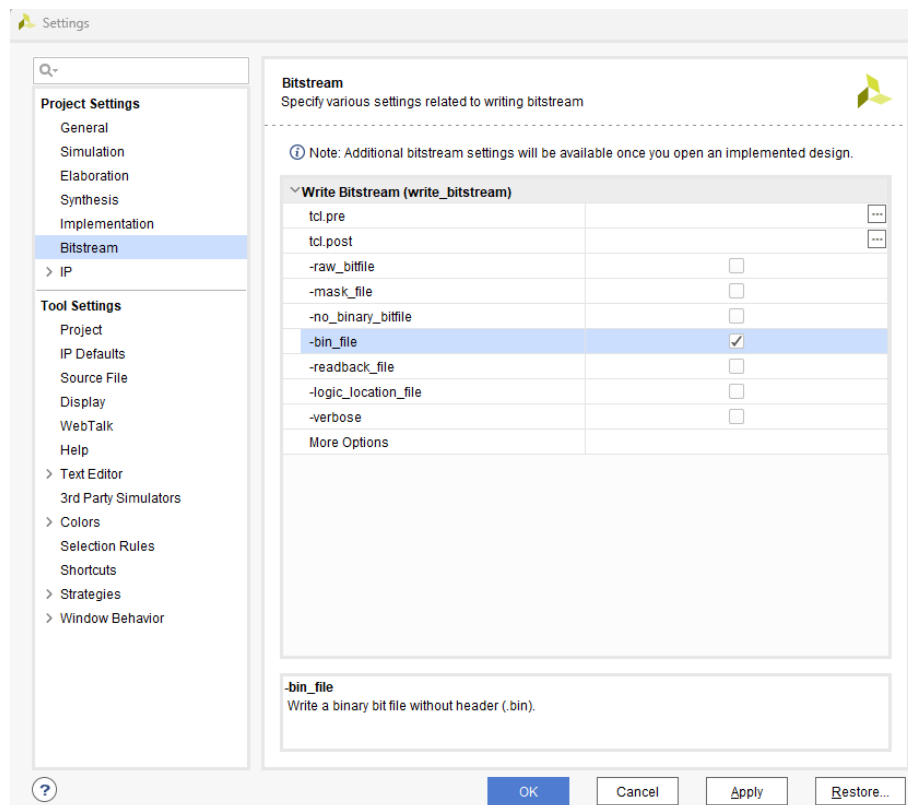
**Tcl Console**

```
INFO: [Labtoolstcl 44-466] Opening hw_target loca
set_property PROGRAM.FILE {C:/FPGA/DemoPrograms/I
current_hw_device [get_hw_devices xc7a100t_0]
refresh_hw_device -update_hw_probes false [linder
INFO: [Labtools 27-1435] Device xc7a100t (JTAG de
```

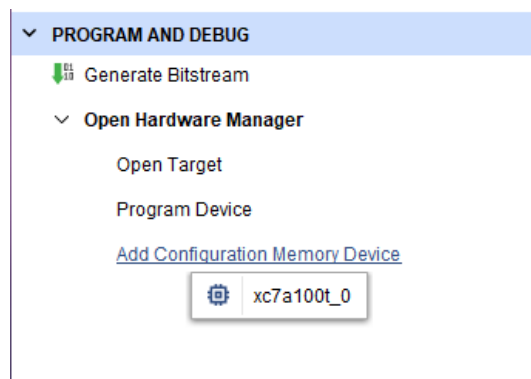


To program the flash

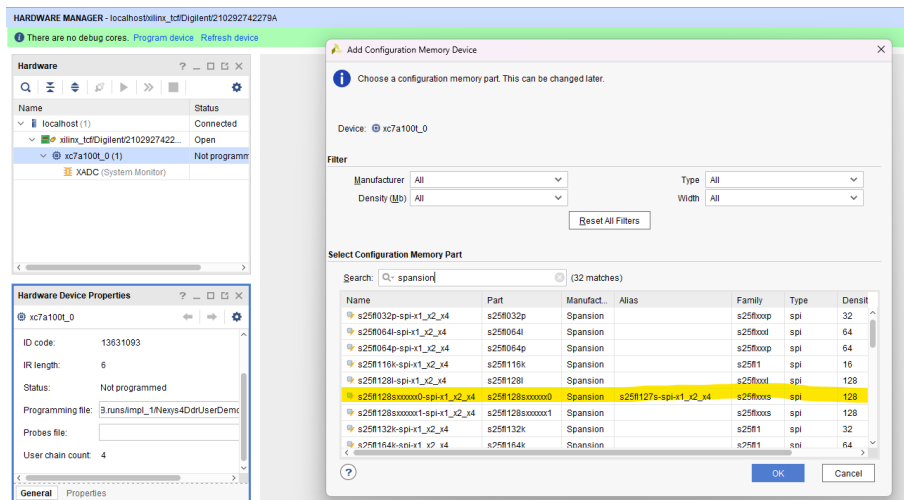
You require a **.bin** file, so update the project settings



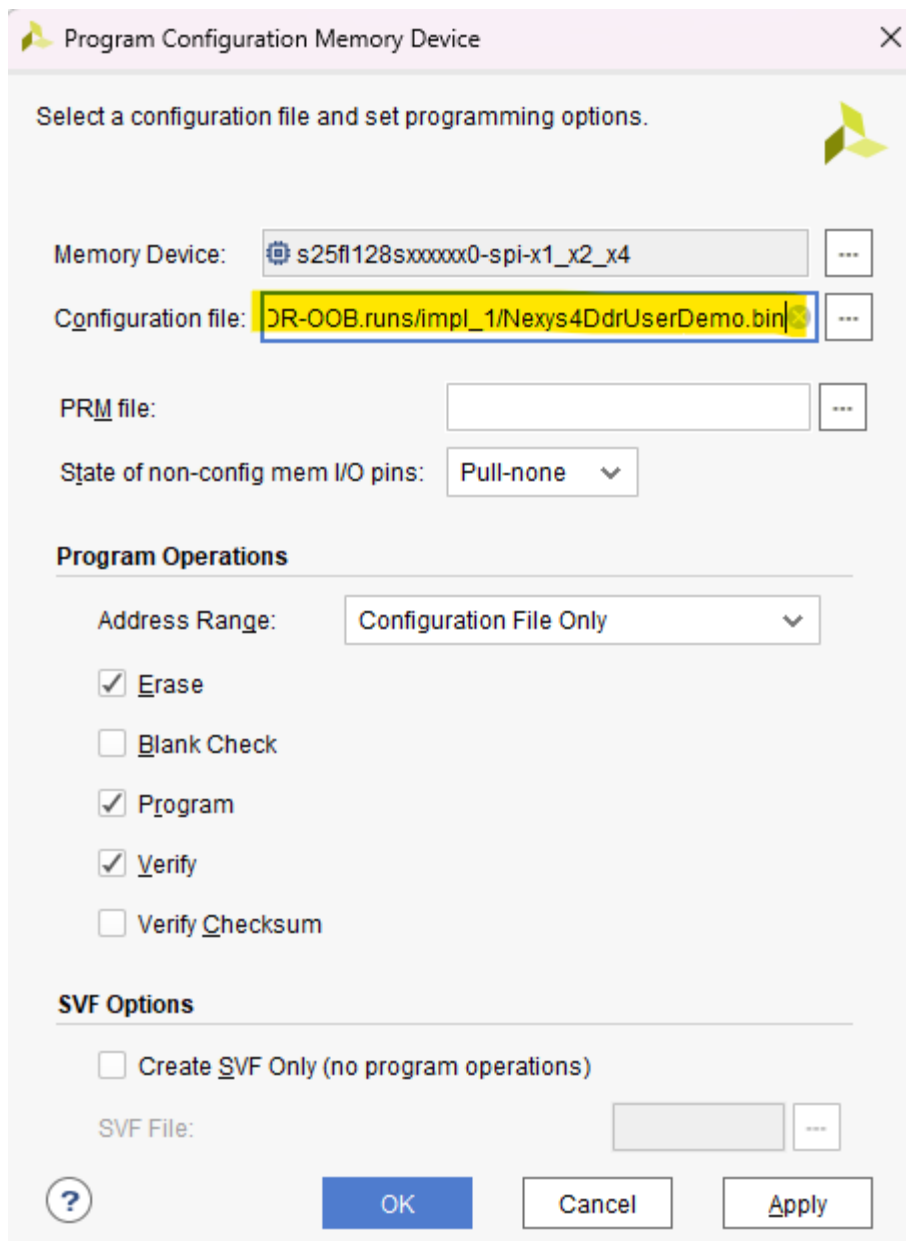
Then add the flash device



Type in 'Spansion' and select the 128 bit device



Then program the device



Press PROG

On USB Drive

Copy the **.bit** file to the root of the UBS drive

Set the SD / USB jumper to USB

Set the JTAG jumper to USB

Insert into the Nexys 4 board

Turn on and press PROG

# Arty S7 (Vivado 2018.2)

## Introduction

XXXXXXXXXX

XXXXXXXXXX

XXXXXXXXXX



# Spartan 3e (Starter Board) (ISE 14.7)

## Introduction

[https://digilent.com/reference/\\_media/reference/programmable-logic/spartan-3e/s3estarter\\_ug.pdf](https://digilent.com/reference/_media/reference/programmable-logic/spartan-3e/s3estarter_ug.pdf)

<https://digilent.com/reference/programmable-logic/spartan-3e/start>

## Supplier example (e3esk-startup)

### Files required

The startup program can be obtained from web.archive.org as below

[https://web.archive.org/web/20090323004714/http://www.xilinx.com/products/boards/s3estarter/reference\\_designs.htm](https://web.archive.org/web/20090323004714/http://www.xilinx.com/products/boards/s3estarter/reference_designs.htm)

[https://web.archive.org/web/20090323004714/http://www.xilinx.com/products/boards/s3estarter/files/s3esk\\_startup.zip](https://web.archive.org/web/20090323004714/http://www.xilinx.com/products/boards/s3estarter/files/s3esk_startup.zip)

To compile it yourself you also need the PicoBlaze source (KCPSM3) from here:

<https://www.amd.com/en/products/adaptive-socs-and-fpgas/intellectual-property/picoblaze.html#tabs-0ab3ca386e-item-e36b215777-tab>

And you may also need the default CPLD design (in case the board doesn't start from Flash), found here:

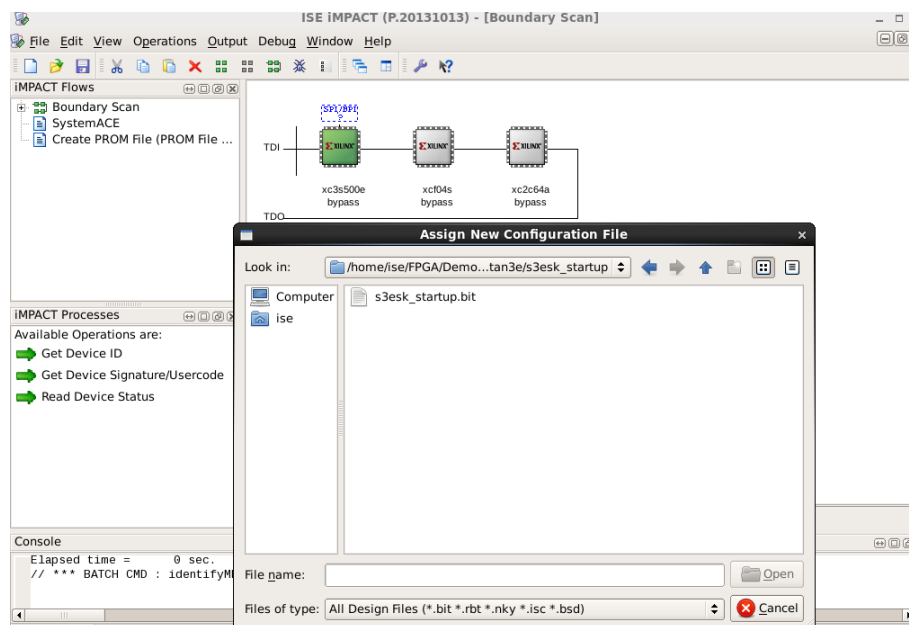
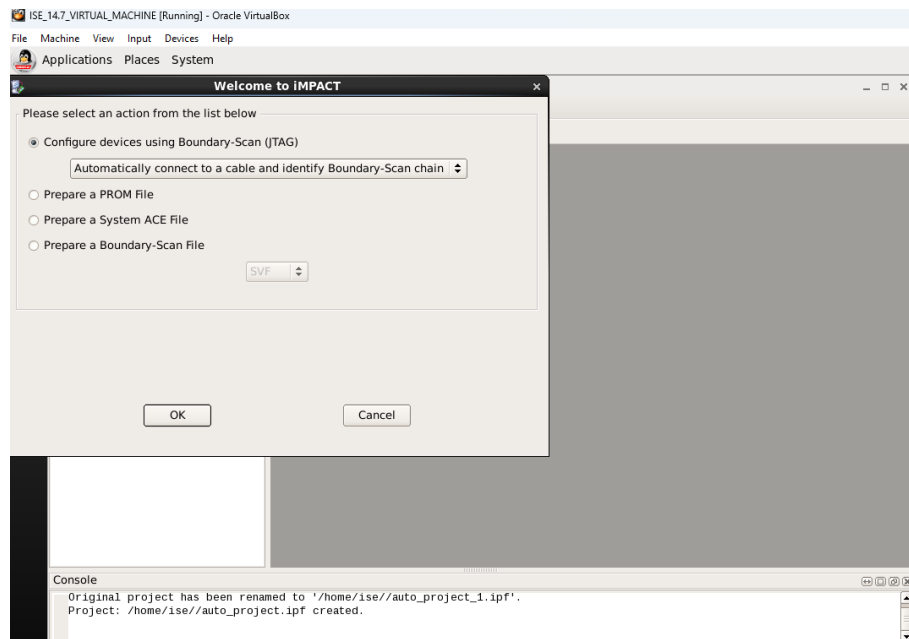
[https://web.archive.org/web/20090323004714/http://www.xilinx.com/products/boards/s3estarter/reference\\_designs.htm](https://web.archive.org/web/20090323004714/http://www.xilinx.com/products/boards/s3estarter/reference_designs.htm)

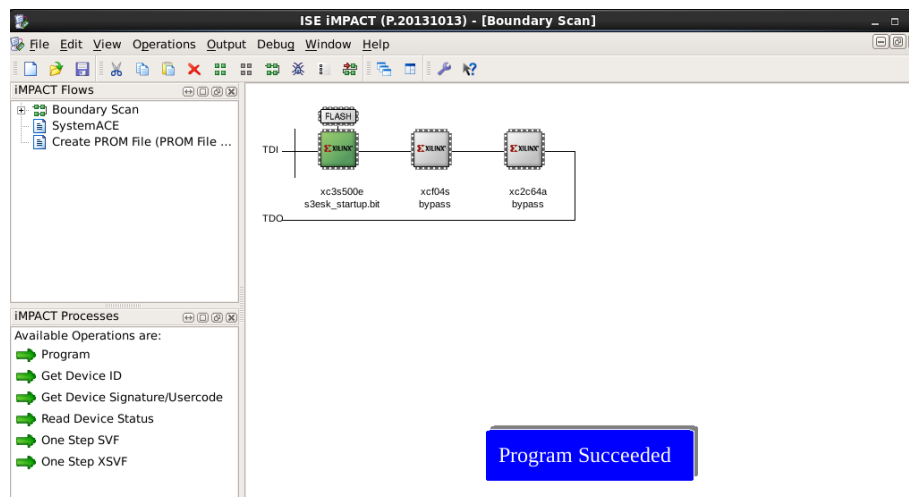
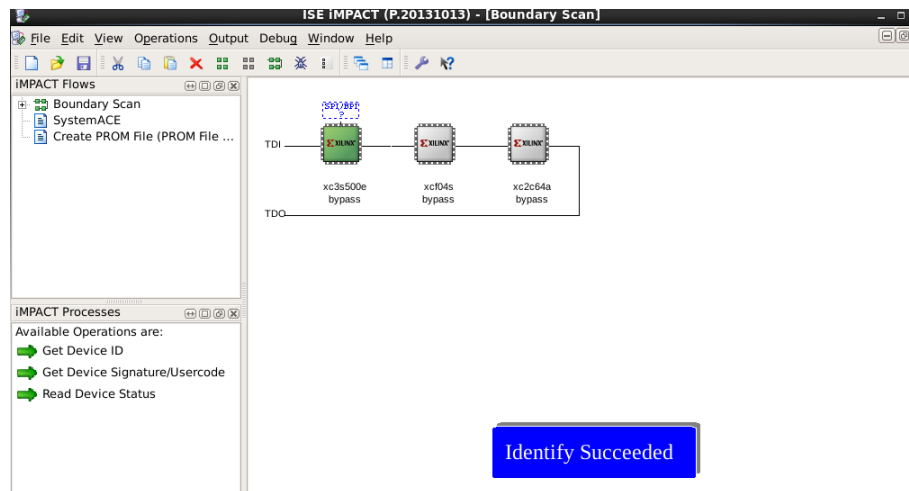
[https://web.archive.org/web/20090323004714/http://www.xilinx.com/products/silicon\\_solutions/cplds/coolrunner\\_series/coolrunner\\_ii\\_cplds/index.htm](https://web.archive.org/web/20090323004714/http://www.xilinx.com/products/silicon_solutions/cplds/coolrunner_series/coolrunner_ii_cplds/index.htm)

## Programming the Spartan 3e from the .bit file

The s3esk\_startup folder contains a .bit and .mcs file: **s3esk\_startup.bit** and **s3esk\_startup\_rev2.mcs**

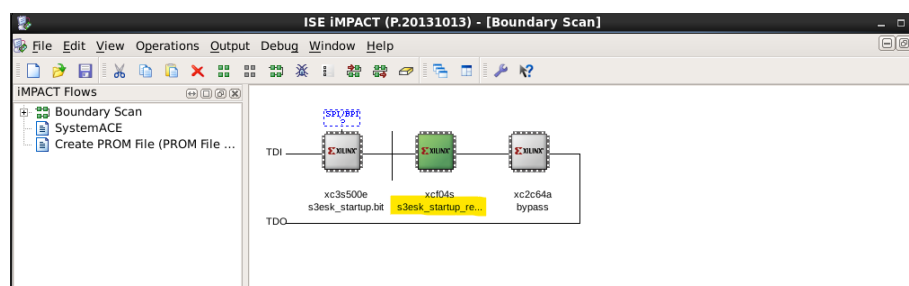
These can be programmed into the Spartan 3E board with the iMPACT programmer (within Virtualbox)





## Programming the xc04s Flash from the mcs file

The **xc04s** flash can also be programmed from the .mcs file in the same way



## Building from source

Create a new project

**Design Properties**

Name: e3esk

Location: /home/ise/FPGA/Spartan3e/s3esk\_startup\_working

Working directory: /home/ise/FPGA/Spartan3e/s3esk\_startup\_working

Description:

**Project Settings**

Property Name	Value
Top-Level Source Type	HDL
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S500E
Package	FG320
Speed	-4

Help Cancel OK

Get the KCPSM3 files from here <https://www.amd.com/en/products/adaptive-socs-and-fpgas/intellectual-property/picoblaze.html>

Load e3esk\_startup.vhd

Load control.vhd

Load kcpsm3.vhd

Edit control.vhs and replace BSCAN\_VIRTEX2 with BSCAN\_SPARTAN3, then save the file.

Load s3esk\_startup.ucf

Build

## Demo

### blink.v

```
module blink(
    clk, // clock signal
    led, // LED pin
    sw
);
```

```

// inputs and outputs
input clk;
input [3:0] sw;

output reg [7:0] led;
reg my_led = 0;

// internal variable
reg [25:0] counter = 50_000_000; // 26 bit variable

always @(posedge clk) begin
    if (counter == 0) begin                // at 1 second
        counter <= 50_000_000;           // reset counter
        my_led <= !my_led;               // invert
    end
    else begin
        counter <= counter - 1;          // decrease
    end

    led[7:7] <= my_led;
    led[6:4] <= 0;
    led[3:0] <= sw[3:0];
end
endmodule

```

## blink.ucf

```

NET "clk" LOC = "C9" | IOSTANDARD = LVCMOS33;
NET "clk" PERIOD = 20.0ns HIGH 40%;

NET "led<0>" LOC = "F12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8;
NET "led<1>" LOC = "E12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8;
NET "led<2>" LOC = "E11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8;
NET "led<3>" LOC = "F11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8;
NET "led<4>" LOC = "C11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8;
NET "led<5>" LOC = "D11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8;
NET "led<6>" LOC = "E9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8;
NET "led<7>" LOC = "F9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8;

NET "sw<0>" LOC = "L13" | IOSTANDARD = LVTTTL | PULLUP;
NET "sw<1>" LOC = "L14" | IOSTANDARD = LVTTTL | PULLUP;
NET "sw<2>" LOC = "H18" | IOSTANDARD = LVTTTL | PULLUP;
NET "sw<3>" LOC = "N17" | IOSTANDARD = LVTTTL | PULLUP;

```

# Nexys (Digital System Builder) (ISE 14.7)

## Introduction

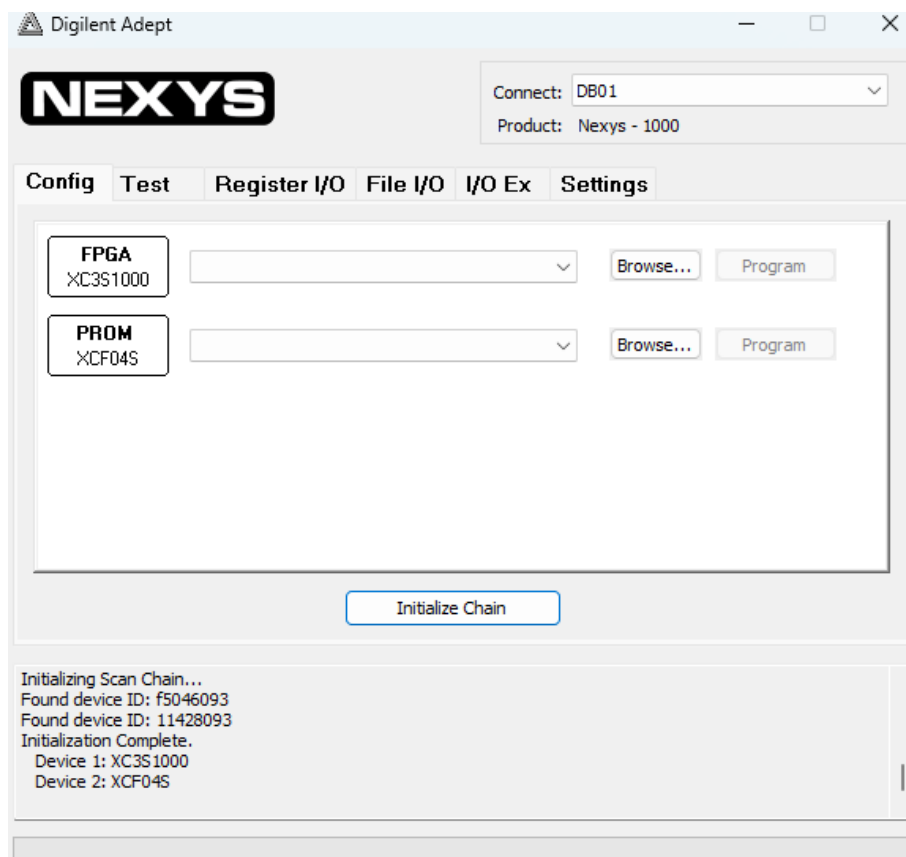
[https://digilent.com/reference/\\_media/nexys/nexys\\_rm.pdf](https://digilent.com/reference/_media/nexys/nexys_rm.pdf)

<https://digilent.com/reference/nexys/nexys>

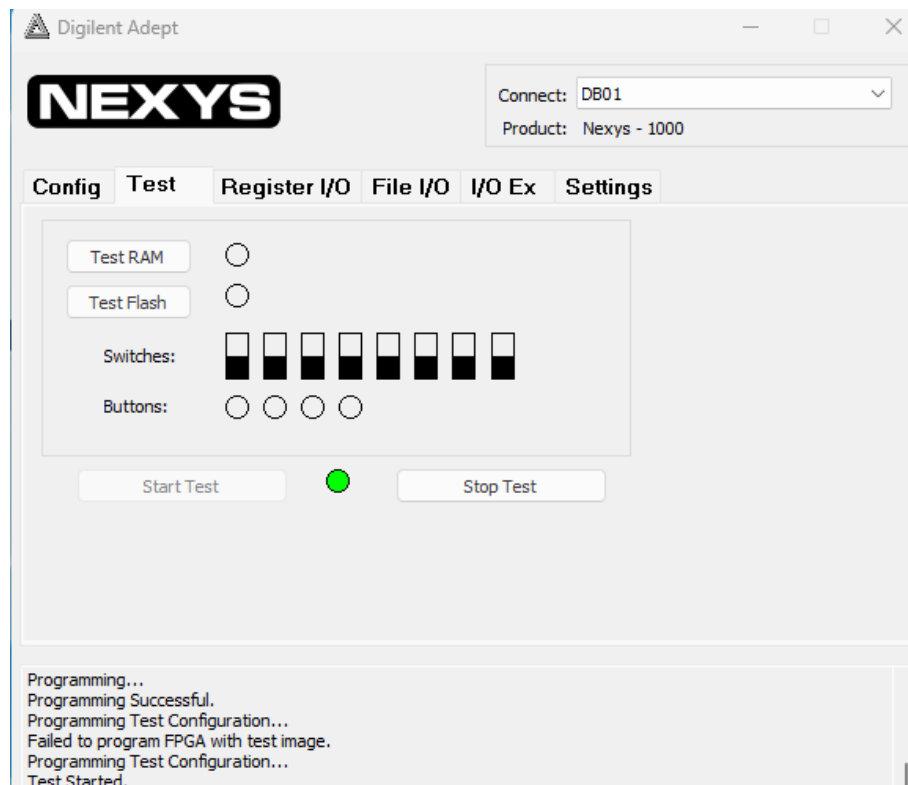
## Testing via Adept

Use **Adept** to find the board and do a test..

Open Adept and press “Initialise Chain”



In Test tab press “Start Test”

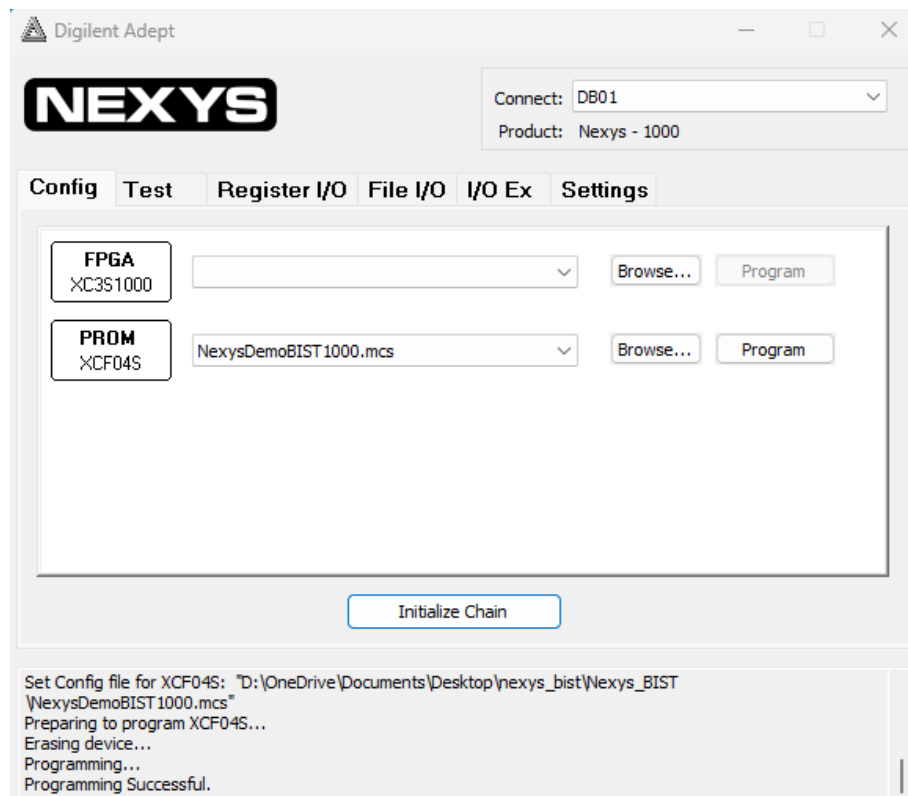


## NexysBIST binary

Download NexysBIST from [https://digilent.com/reference/\\_media/nexys/nexys\\_bist.zip](https://digilent.com/reference/_media/nexys/nexys_bist.zip)

Or linked from here: <https://digilent.com/reference/nexys/nexys>

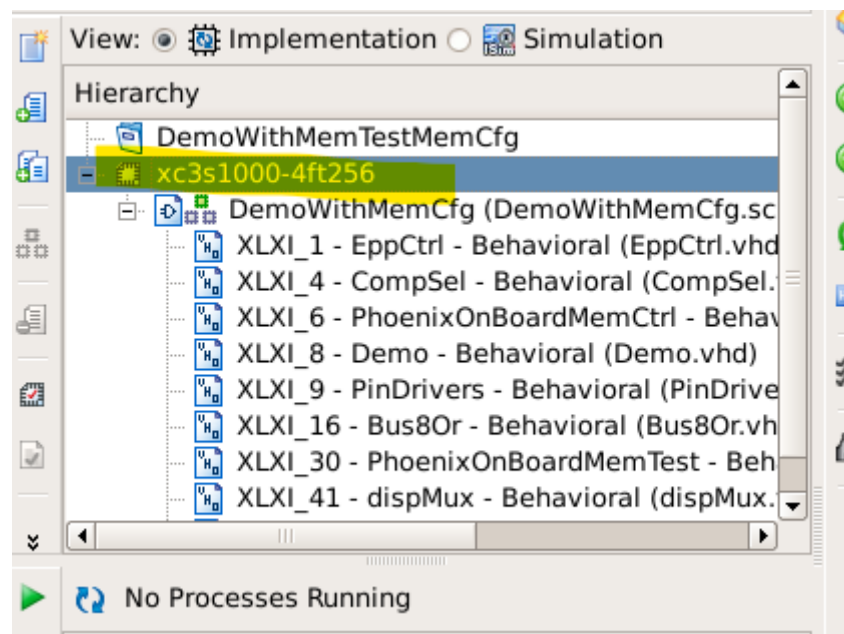
Program PROM with **NexysDemoBIST1000.mcs**



## Building NexysBIST from source

In Project Navigator open **DemoWithMemTestMemCfg.xise**

Change the Design Properties to select a **xc3s1000** device





**Design Properties** [X]

Name: DemoWithMemTestMemCfg

Location: /home/ise/FPGA/Nexys/Nexys\_BIST

Working directory: /home/ise/FPGA/Nexys/Nexys\_BIST

Description:

Project Settings

Property Name	Value
Top-Level Source Type	HDL
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3
Device	XC3S1000
Package	FT256
Speed	-4

Then build the project

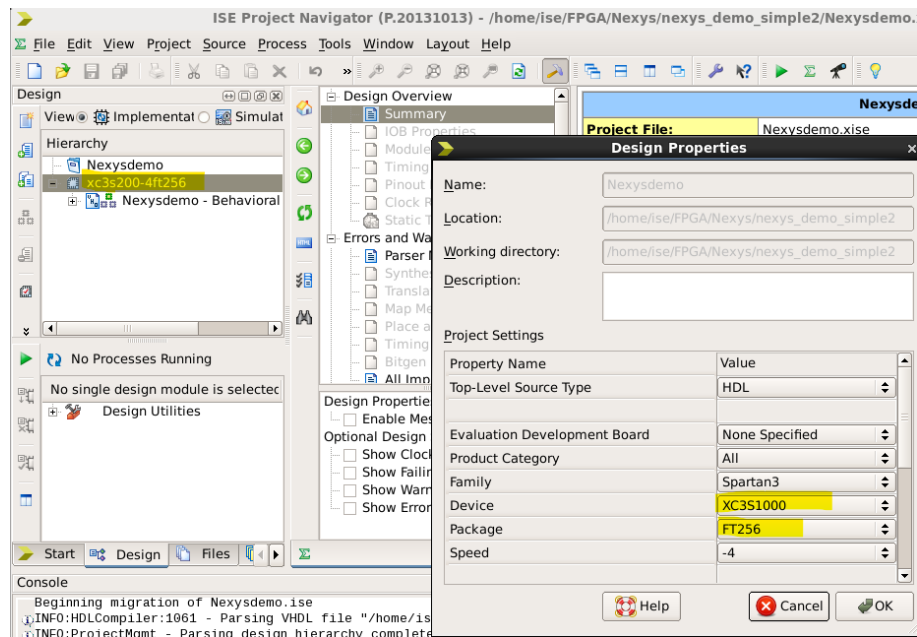
## Supplier example: Nexysdemo

Select device: **xc3s1000-4ft256**

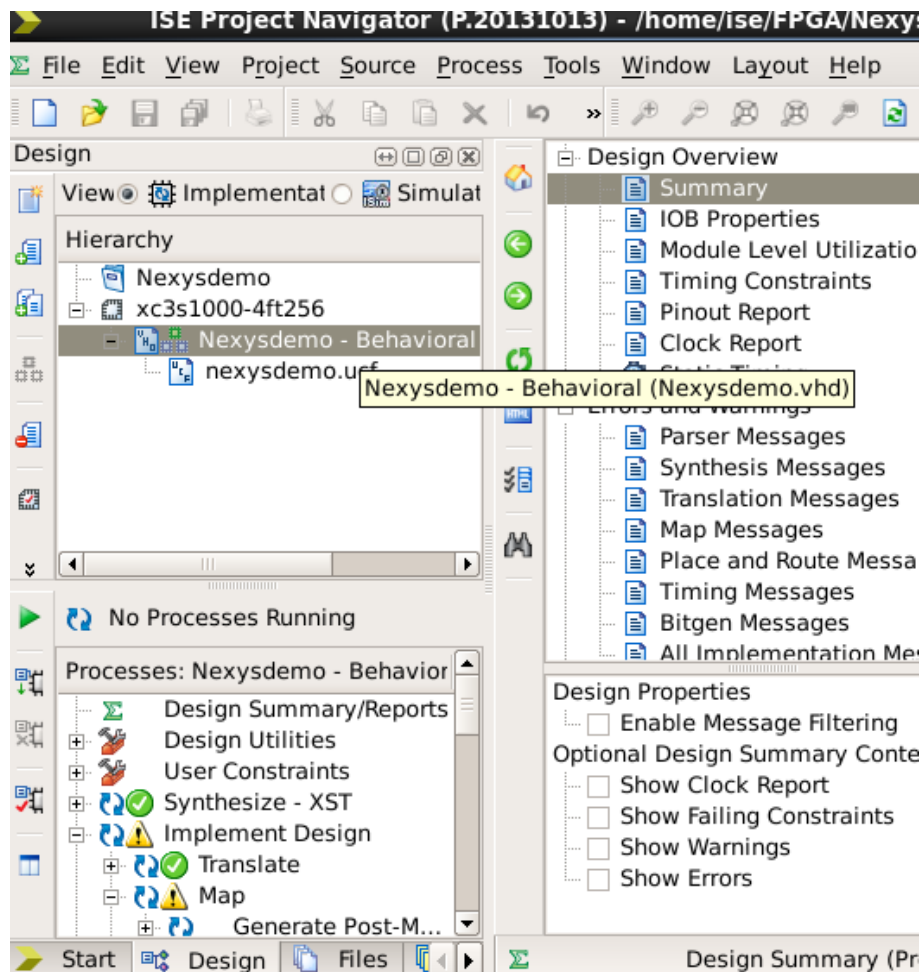
Project open: Nexysdemo.ise (change default from .xise)

Convert to ISE 14.7 (Migrate Only)

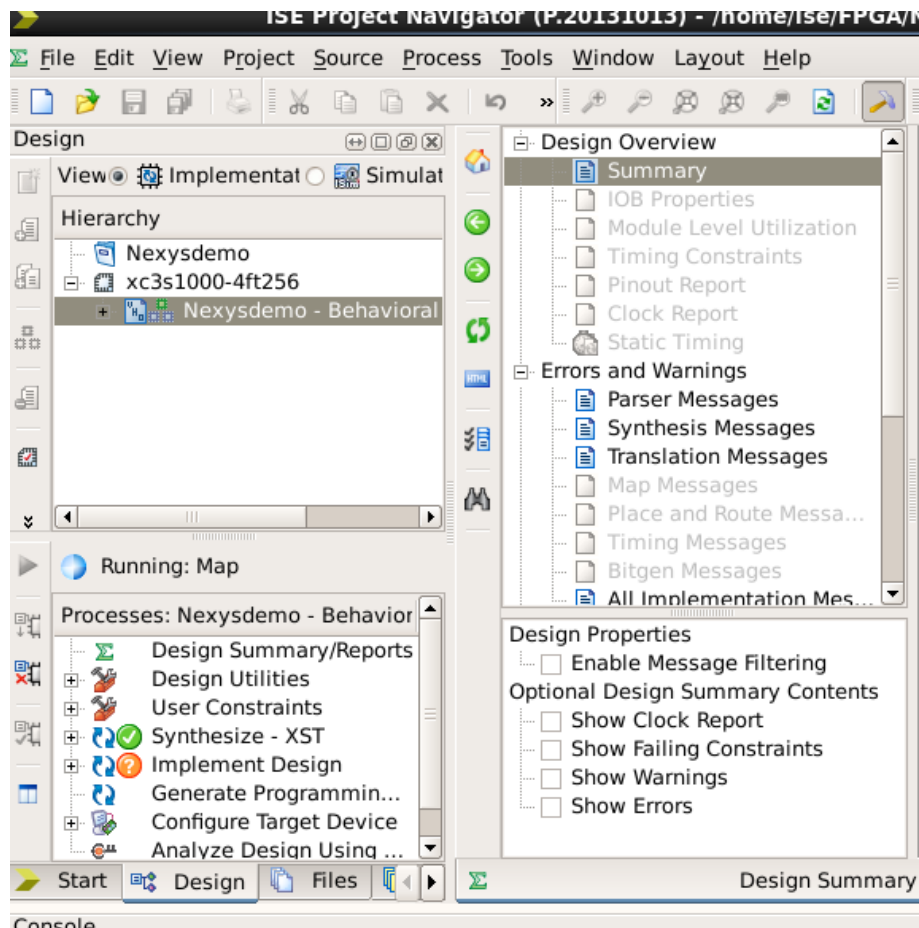
Right click on device and select Design Properties – change to xc3s1000



Click on Nexysdemo – Behavioral in the top left pane



Select Process / Implement Top Module



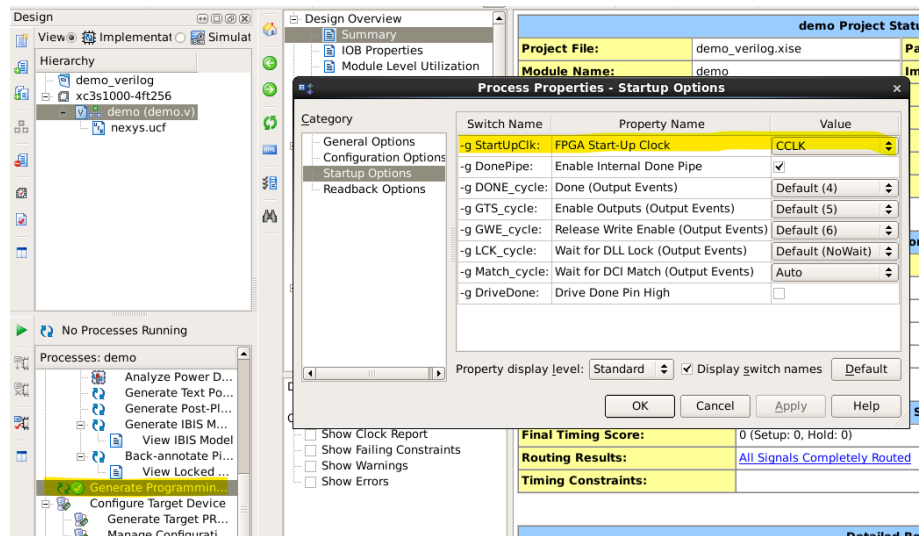
The process may stop with warnings on Implement Design.

In the Generate Programming File process, right click.

Select Process Properties

Then select the Startup Options tab and select JTAG CLK.

This will stop any error like 'Startup clock for this file is CCLK instead of JTAG CLK'



Run 'Generate Programming File'

Programming JTAG

Back in Windows, use Adept to load the Nexysdemo.bit file.



Programming Flash

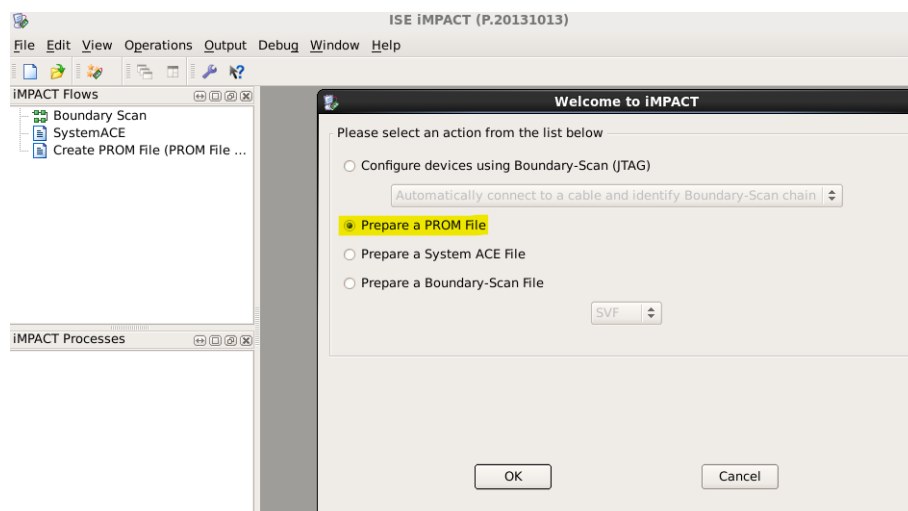
To program the Platform ROM (xcf04s) as shown in Adept, you need to use Impact in the VirtualBox

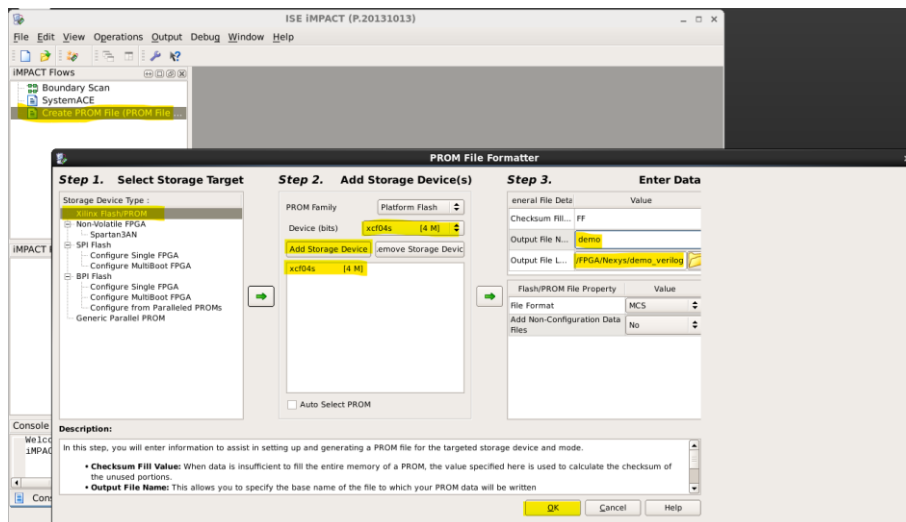
Open Impact

If asked about

Select Yes to 'Automatically create and save a project'

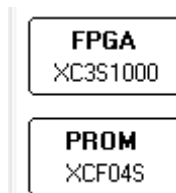
Select 'Prepare a PROM file'





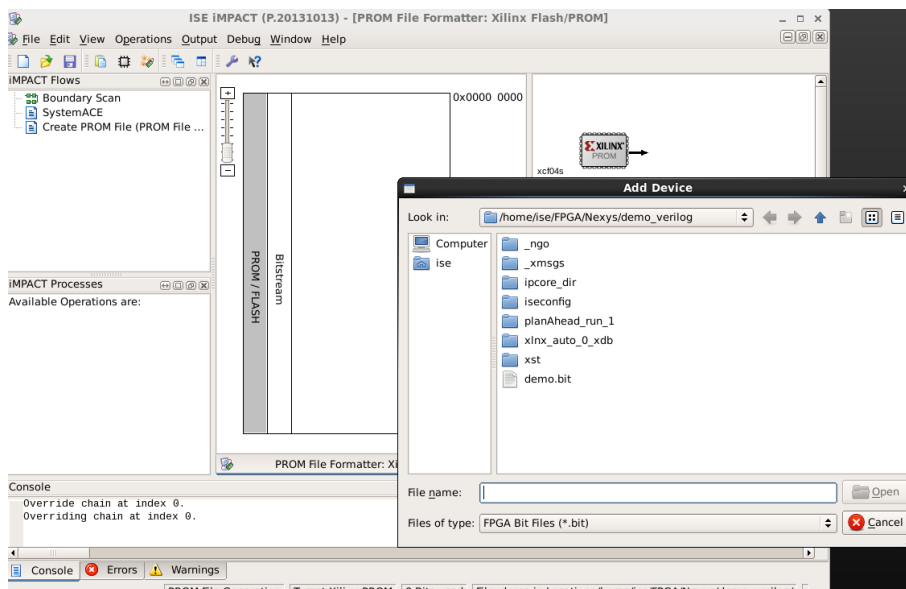
Select 'Xilinx Flash/PROM', right arrow

Then select Platform Flash, xcf04s and 'Add Storage Device'



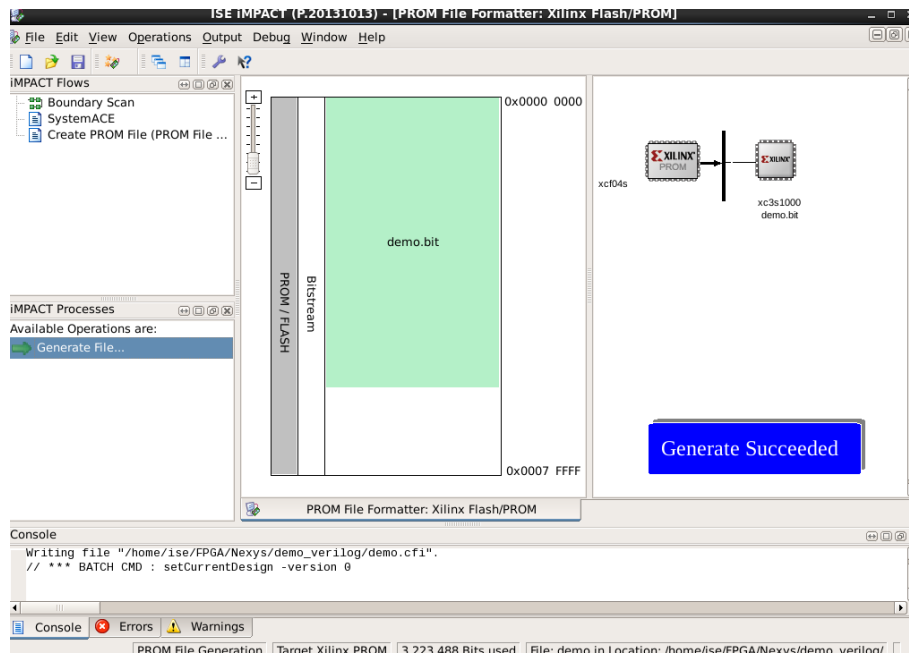
Then select the directory and file name you want, and click Ok

Then in the following screen select the .bit file you want to convert

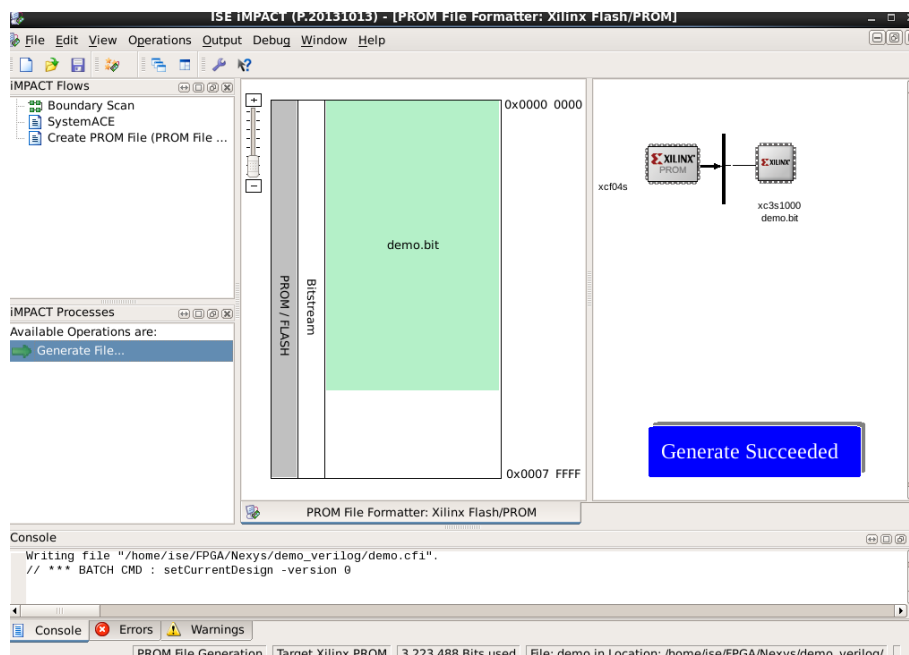


Say 'No' to the Add Device pop-up box.

Then Ok on the next pop-up box.



Then double-click on Generate File



Now, in Adept you can use this file (.mcs) to program the PROM

## Demo

### demo.v

```
module demo(  
    input mclk,  
    input [3:0] btn,  
    input [7:0] swt,  
    output reg [7:0] led,  
    output reg [3:0] anode,  
    output reg [7:0] ssg  
);  
  
    always @(posedge mclk) begin  
        led <= swt;  
        anode <= btn;  
        ssg <= 8'b00000000;  
    end  
endmodule
```

### nexys.ucf

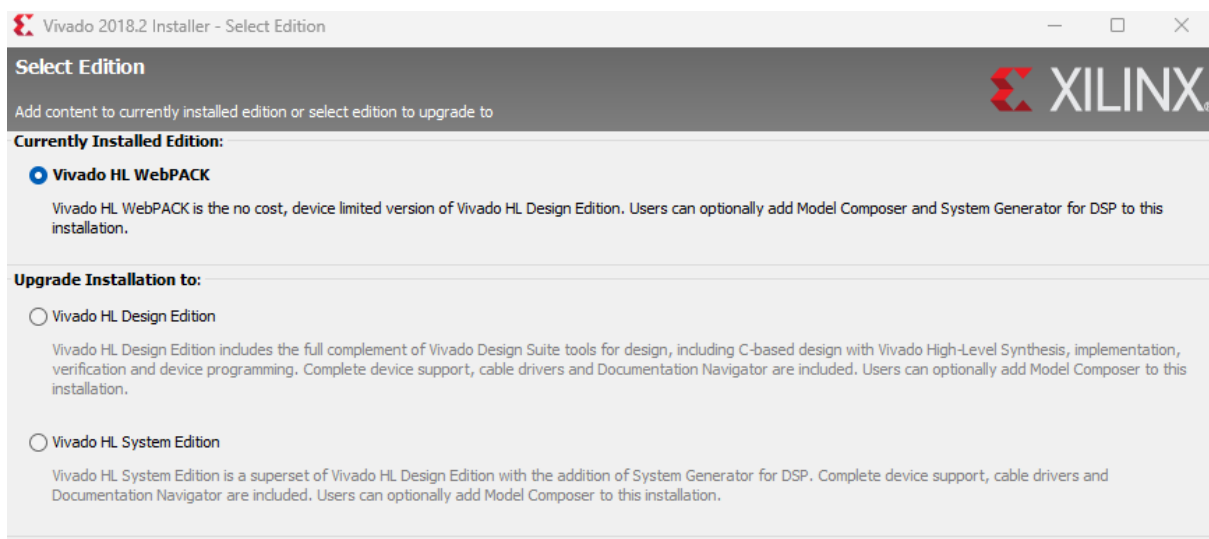
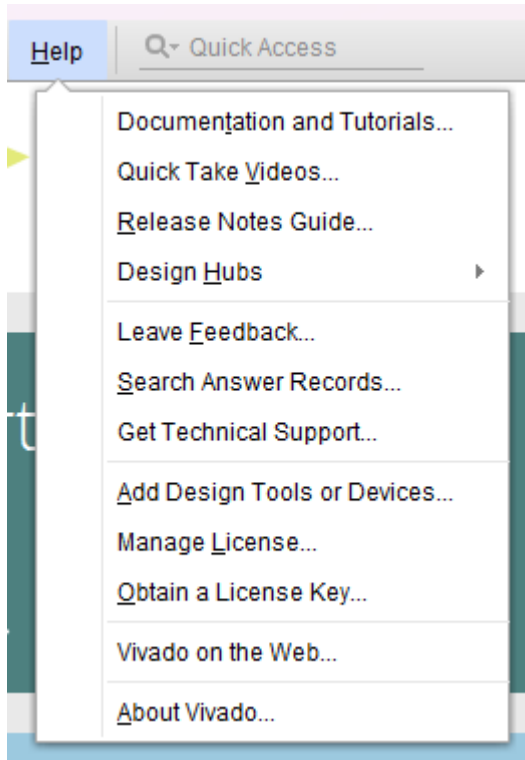
```
NET "mclk"      LOC = "A8";  
  
NET "btn<0>" LOC = "J13";  
NET "btn<1>" LOC = "K14";  
NET "btn<2>" LOC = "K13";  
NET "btn<3>" LOC = "K12";  
  
NET "swt<0>" LOC = "N15";  
NET "swt<1>" LOC = "J16";  
NET "swt<2>" LOC = "K16";  
NET "swt<3>" LOC = "K15";  
NET "swt<4>" LOC = "L15";  
NET "swt<5>" LOC = "M16";  
NET "swt<6>" LOC = "M15";  
NET "swt<7>" LOC = "N16";  
  
NET "anode<0>" LOC = "G14";  
NET "anode<1>" LOC = "G12";  
NET "anode<2>" LOC = "G13";  
NET "anode<3>" LOC = "F12";  
  
NET "ssg<0>" LOC = "F13";  
NET "ssg<1>" LOC = "E13";  
NET "ssg<2>" LOC = "G15";  
NET "ssg<3>" LOC = "H13";  
NET "ssg<4>" LOC = "J14";  
NET "ssg<5>" LOC = "E14";  
NET "ssg<6>" LOC = "G16";  
NET "ssg<7>" LOC = "H14";  
  
NET "led<0>" LOC = "L14";  
NET "led<1>" LOC = "L13";  
NET "led<2>" LOC = "M14";  
NET "led<3>" LOC = "L12";  
NET "led<4>" LOC = "N14";  
NET "led<5>" LOC = "M13";  
NET "led<6>" LOC = "P14";  
NET "led<7>" LOC = "R16";
```

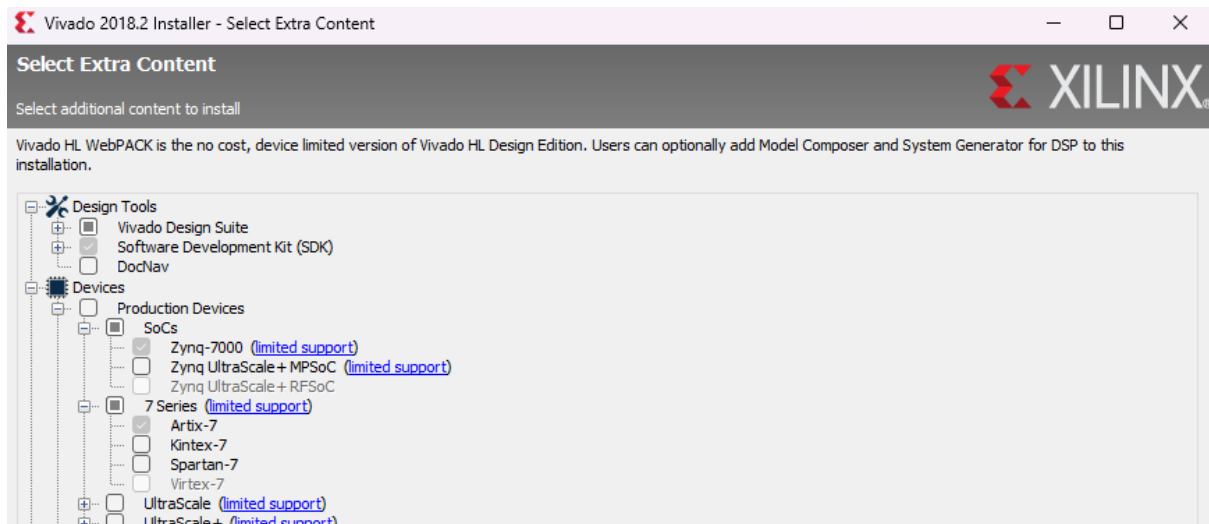


# Zybo (Vivado 2018.2)

## Introduction

Add the support for the Zybo chipset to Vivado: Help – Add Design Tools or Devices. It will start the installer to add the chipset.





Select Zynq-7000

Use xc7z010clg400-1 as the device.

## Demo

### sw\_led.v

```
`timescale 1ns / 1ps

module sw_led(
    input          clk,
    input  [3:0]   sw,    // Slide switch inputs
    output reg [3:0] led  // LED outputs
);

    always @(posedge clk)
    begin
        led <= sw;
    end
endmodule
```

### ZYBO\_constraints.xdc

```
##Clock signal
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { clk }];
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];

##Switches
set_property -dict { PACKAGE_PIN G15   IOSTANDARD LVCMOS33 } [get_ports { sw[0] }];
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { sw[1] }];
set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }];
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { sw[3] }];

##LEDs
set_property -dict { PACKAGE_PIN M14   IOSTANDARD LVCMOS33 } [get_ports { led[0] }];
set_property -dict { PACKAGE_PIN M15   IOSTANDARD LVCMOS33 } [get_ports { led[1] }];
set_property -dict { PACKAGE_PIN G14   IOSTANDARD LVCMOS33 } [get_ports { led[2] }];
set_property -dict { PACKAGE_PIN D18   IOSTANDARD LVCMOS33 } [get_ports { led[3] }];
```

# Gowin DK-START (Gowin)

## Introduction

Target device: GW2A-LV18PG256C8/I7

## Demo

Verilog: leds.v

```
module led (
    input sys_clk,           // clk input
    input sys_rst_n,        // reset input
    output reg [3:0] led,    // 4 LEDS pin
    input [3:0] btn,
    input swi0
);

reg [23:0] counter;
reg dir;

initial begin
    counter <= 24'd0;
    led <= 4'b0001;
    dir <= 1'b0;
end

always @(posedge sys_clk) begin
    if (counter < 24'd1349_9999)           // 0.5s delay
        counter <= counter + 1'd1;
    else
        counter <= 24'd0;
end

always @(posedge sys_clk) begin
    if (counter == 24'd1349_9999) begin    // 0.5s delay
        if (dir)
            led[3:0] <= {btn[0], led[3:1]};
        else
            led[3:0] <= {led[2:0], btn[0]};
        end

        if (btn[3] == 1'b0)
            led <= 4'b0000;

        dir <= swi0;
    end
endmodule
```

Physical constraints: leds.cst

```
IO_LOC "led[0]" F14;
IO_PORT "led[0]" PULL_MODE=UP DRIVE=8;
IO_LOC "led[1]" F13;
IO_PORT "led[1]" PULL_MODE=UP DRIVE=8;
IO_LOC "led[2]" G12;
IO_PORT "led[2]" PULL_MODE=UP DRIVE=8;
IO_LOC "led[3]" F16;
IO_PORT "led[3]" PULL_MODE=UP DRIVE=8;

IO_LOC "btn[0]" T5;
IO_PORT "btn[0]" PULL_MODE=UP;
IO_LOC "btn[1]" T4;
IO_PORT "btn[1]" PULL_MODE=UP;
IO_LOC "btn[2]" T3;
IO_PORT "btn[2]" PULL_MODE=UP;
IO_LOC "btn[3]" T2;
IO_PORT "btn[3]" PULL_MODE=UP;

IO_LOC "swi0" E9;

IO_LOC "sys_clk" H11;
IO_PORT "sys_clk" PULL_MODE=UP
```

# Tango Nano 9K (Gowin)

Target device: GW1NR-LV9QN88PC6/I5

## Demo

Verilog: leds.v

```
module led (
    input sys_clk,           // clk input
    input sys_rst_n,        // reset input
    output reg [5:0] led     // 6 LEDS pin
);

reg [23:0] counter;

always @(posedge sys_clk or negedge sys_rst_n) begin
    if (!sys_rst_n)
        counter <= 24'd0;
    else if (counter < 24'd0349_9999) // 0.5s delay
        counter <= counter + 1'd1;
    else
        counter <= 24'd0;
end

always @(posedge sys_clk or negedge sys_rst_n) begin
    if (!sys_rst_n)
        led <= 6'b111110;
    else if (counter == 24'd0349_9999) // 0.5s delay
        led[5:0] <= {led[4:0],led[5]};
    else
        led <= led;
end

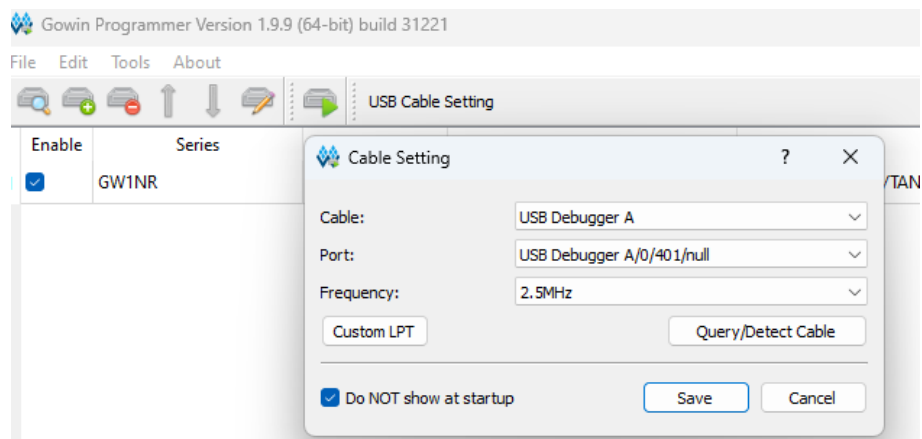
endmodule
```

Physical constraints: leds.cst

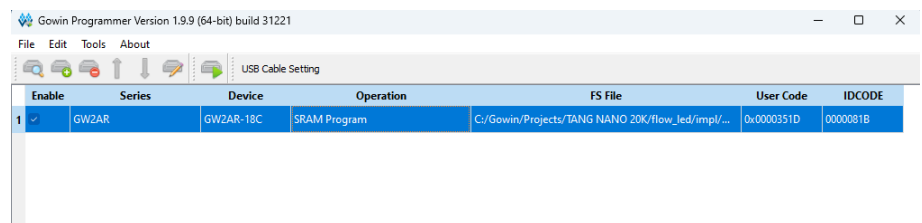
```
IO_LOC "led[5]" 16;
IO_PORT "led[5]" PULL_MODE=UP DRIVE=8;
IO_LOC "led[4]" 15;
IO_PORT "led[4]" PULL_MODE=UP DRIVE=8;
IO_LOC "led[3]" 14;
IO_PORT "led[3]" PULL_MODE=UP DRIVE=8;
IO_LOC "led[2]" 13;
IO_PORT "led[2]" PULL_MODE=UP DRIVE=8;
IO_LOC "led[1]" 11;
IO_PORT "led[1]" PULL_MODE=UP DRIVE=8;
IO_LOC "led[0]" 10;
IO_PORT "led[0]" PULL_MODE=UP DRIVE=8;
IO_LOC "sys_rst_n" 4;
IO_PORT "sys_rst_n" PULL_MODE=UP;
IO_LOC "sys_clk" 52;
IO_PORT "sys_clk" IO_TYPE=LVCNMOS33 PULL_MODE=UP;
```

Open Programmer

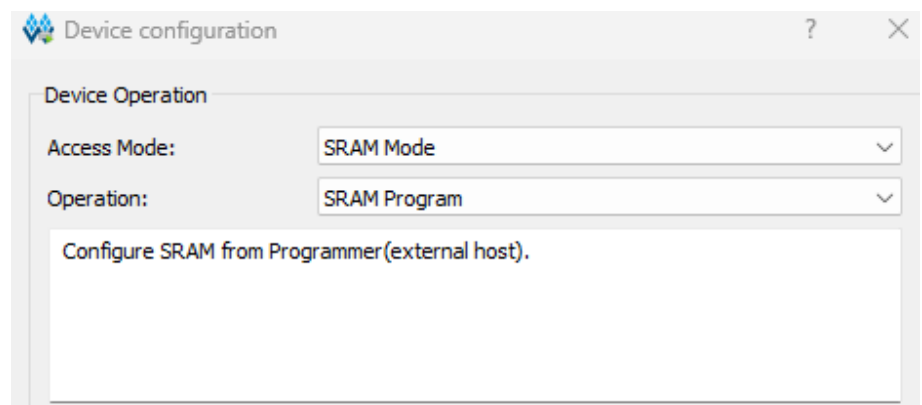
Press USB Cable Setting and select USB Debugger A



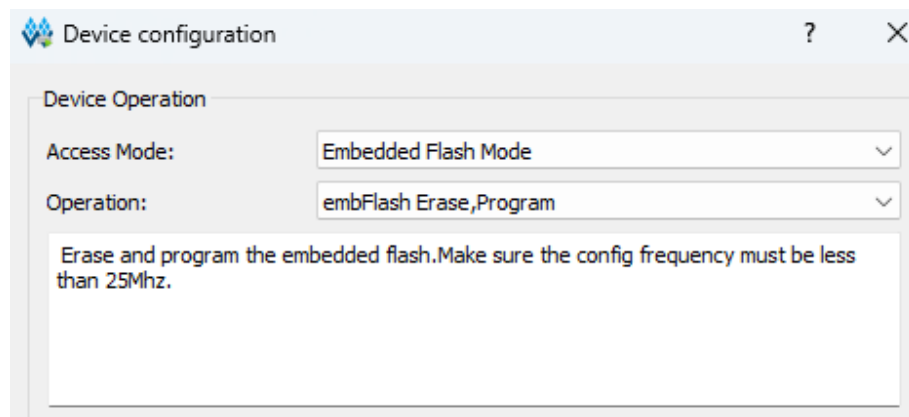
Choose Operation - either program to SRAM or Flash



Programming SRAM



Programming Flash



# Tang Nano 20K (Gowin)

## Introduction

Target device: **GW2AR-LV18QN88C8/I7**

## Demo

Verilog files: **leds.v**

```
module top#(
    parameter led_number = 6
)
(
    input      clk,
    output [ led_number-1 :0] leds
);

reg count_1s_flag;
reg [23:0] count_1s = 'd0;

always @(posedge clk ) begin
    if( count_1s < 27000000/2 ) begin
        count_1s <= count_1s + 'd1;
        count_1s_flag <= 'd0;
    end
    else begin
        count_1s <= 'd0;
        count_1s_flag <= 'd1;
    end
end

reg [5:0] leds_value = 'd1;

always @(posedge clk ) begin
    if( count_1s_flag ) begin
        leds_value[ led_number-1 :0] <= {leds_value[ led_number-2 :0] , leds_value[
led_number-1 ] };
    end
end

assign leds = ~leds_value;

endmodule
```

Physical constraints: leds.cst

```
IO_LOC  "leds[5]" 20;
IO_PORT "leds[5]" PULL_MODE=UP DRIVE=8 BANK_VCCIO=1.8;
IO_LOC  "leds[4]" 19;
IO_PORT "leds[4]" PULL_MODE=UP DRIVE=8 BANK_VCCIO=1.8;
IO_LOC  "leds[3]" 18;
IO_PORT "leds[3]" PULL_MODE=UP DRIVE=8 BANK_VCCIO=1.8;
IO_LOC  "leds[2]" 17;
IO_PORT "leds[2]" PULL_MODE=UP DRIVE=8 BANK_VCCIO=1.8;
IO_LOC  "leds[1]" 16;
IO_PORT "leds[1]" PULL_MODE=UP DRIVE=8 BANK_VCCIO=1.8;
IO_LOC  "leds[0]" 15;
IO_PORT "leds[0]" PULL_MODE=UP DRIVE=8 BANK_VCCIO=1.8;
IO_LOC  "clk" 4;
IO_PORT "clk" PULL_MODE=UP BANK_VCCIO=1.8;
```



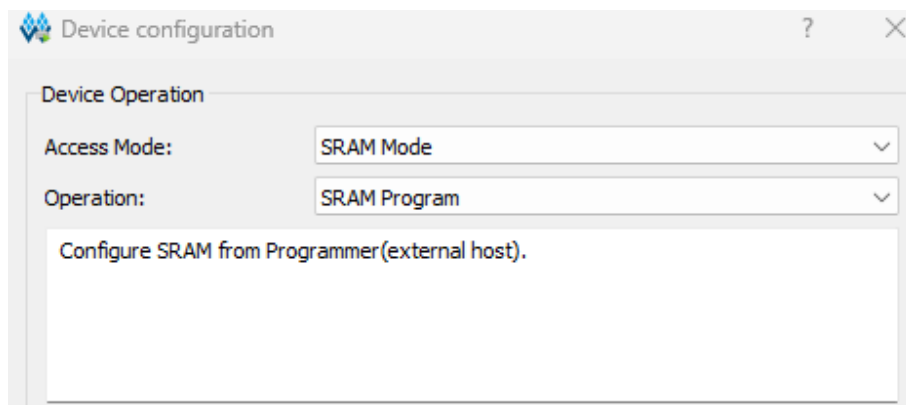
Timing constraints: leds.sdc

```
create_clock -name clk_27M -period 37.037 -waveform {0 18.518} [get_ports {clk}]
```

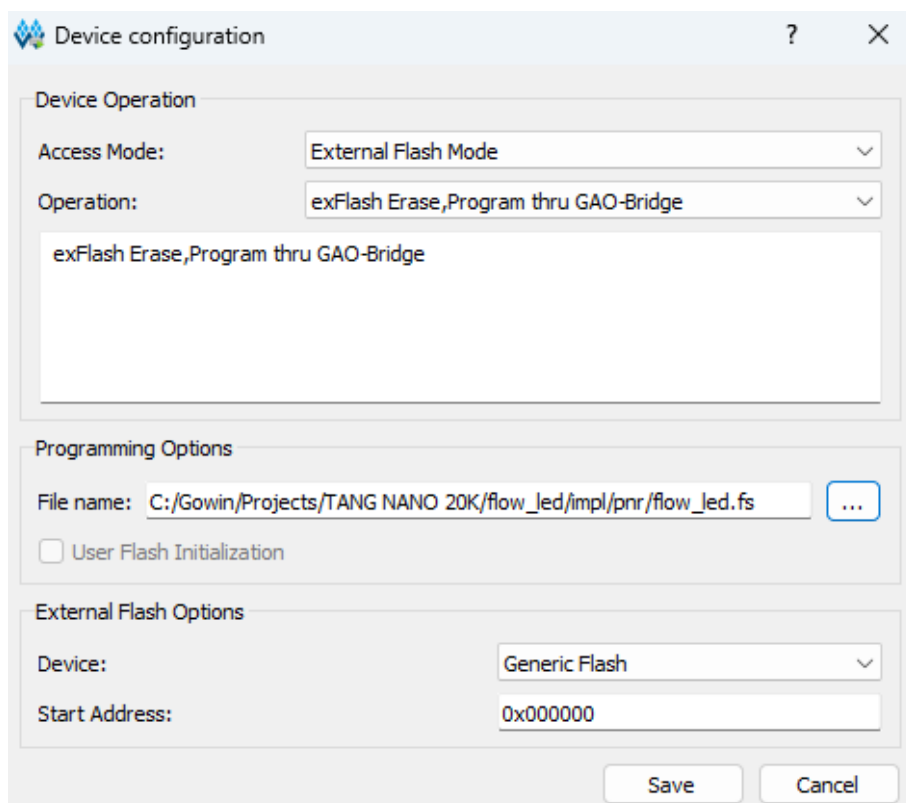
In Programmer / Operation

Select either **SRAM** mode or **External Flash** mode

Programming SRAM



Programming flash



# Tang Primer 25k (Gowin)

## Introduction

Uses LEDS in top PMOD, near the USB host socket.

Target device: **GW5A-LV25MG121NES**

The 'Select Device' dialog box has a filter section with dropdowns for Series (GW5A), Device (GW5A-25), Device Version (A), Package (Any), Speed (Any), and Voltage (Any). Below the filter is a table of devices with columns: Part Number, Device, Device Version, Package, Speed, Voltage, IO, and LUT. The row for GW5A-LV25MG121NES is highlighted.

Part Number	Device	Device Version	Package	Speed	Voltage	IO	LUT
GW5A-EV25UG3245C1/I0	GW5A-25	A	UBGA324S	C1/I0	EV	239	2304C
GW5A-LV25MG121NC2/I1	GW5A-25	A	MBGA121N	C2/I1	LV	82	2304C
GW5A-EV25UG3245ES	GW5A-25	A	UBGA324S	ES	EV	239	2304C
<b>GW5A-LV25MG121NES</b>	<b>GW5A-25</b>	<b>A</b>	<b>MBGA121N</b>	<b>ES</b>	<b>LV</b>	<b>82</b>	<b>2304C</b>
GW5A-EV25LQ100C1/I0	GW5A-25	A	LQFP100	C1/I0	EV	80	2304C
GW5A-EV25PG256CC2/I1	GW5A-25	A	PBGA256C	C2/I1	EV	191	2304C
GW5A-EV25UG256CES	GW5A-25	A	UBGA256C	ES	EV	191	2304C
GW5A-EV25UG256CC1/I0	GW5A-25	A	UBGA256C	C1/I0	EV	191	2304C
GW5A-EV25LQ100ES	GW5A-25	A	LQFP100	ES	EV	80	2304C
GW5A-EV25PG256CC1/I0	GW5A-25	A	PBGA256C	C1/I0	EV	191	2304C

## Demo

Verilog files: **pmod\_led.v**

```
module top(  
    input clk,  
    output pmod_io[7:0]  
);  
  
    reg [29:0] counter;  
  
    assign pmod_io[0] = ~counter[29];  
    assign pmod_io[1] = ~counter[28];  
    assign pmod_io[2] = ~counter[27];  
    assign pmod_io[3] = ~counter[26];  
    assign pmod_io[4] = ~counter[25];  
    assign pmod_io[5] = ~counter[24];  
    assign pmod_io[6] = ~counter[23];  
    assign pmod_io[7] = ~counter[22];  
  
    initial begin  
        counter = 0;  
    end  
  
    always @(posedge clk)  
    begin  
        counter <= counter + 1;  
    end  
endmodule // top
```

Physical constraints: **pmod\_led.cst**

```
IO_LOC "pmod_io[0]" G5;  
IO_LOC "pmod_io[1]" F5;
```

```

IO_LOC "pmod_io[2]" G7;
IO_LOC "pmod_io[3]" G8;
IO_LOC "pmod_io[4]" H7;
IO_LOC "pmod_io[5]" H8;
IO_LOC "pmod_io[6]" H5;
IO_LOC "pmod_io[7]" J5;

IO_LOC "clk" E2;

IO_PORT "pmod_io[0]" PULL_MODE=NONE DRIVE=8 BANK_VCCIO=3.3;
IO_PORT "pmod_io[1]" PULL_MODE=NONE DRIVE=8 BANK_VCCIO=3.3;
IO_PORT "pmod_io[2]" PULL_MODE=NONE DRIVE=8 BANK_VCCIO=3.3;
IO_PORT "pmod_io[3]" PULL_MODE=NONE DRIVE=8 BANK_VCCIO=3.3;
IO_PORT "pmod_io[4]" PULL_MODE=NONE DRIVE=8 BANK_VCCIO=3.3;
IO_PORT "pmod_io[5]" PULL_MODE=NONE DRIVE=8 BANK_VCCIO=3.3;
IO_PORT "pmod_io[6]" PULL_MODE=NONE DRIVE=8 BANK_VCCIO=3.3;
IO_PORT "pmod_io[7]" PULL_MODE=NONE DRIVE=8 BANK_VCCIO=3.3;

IO_PORT "clk" PULL_MODE=NONE DRIVE=OFF BANK_VCCIO=3.3;

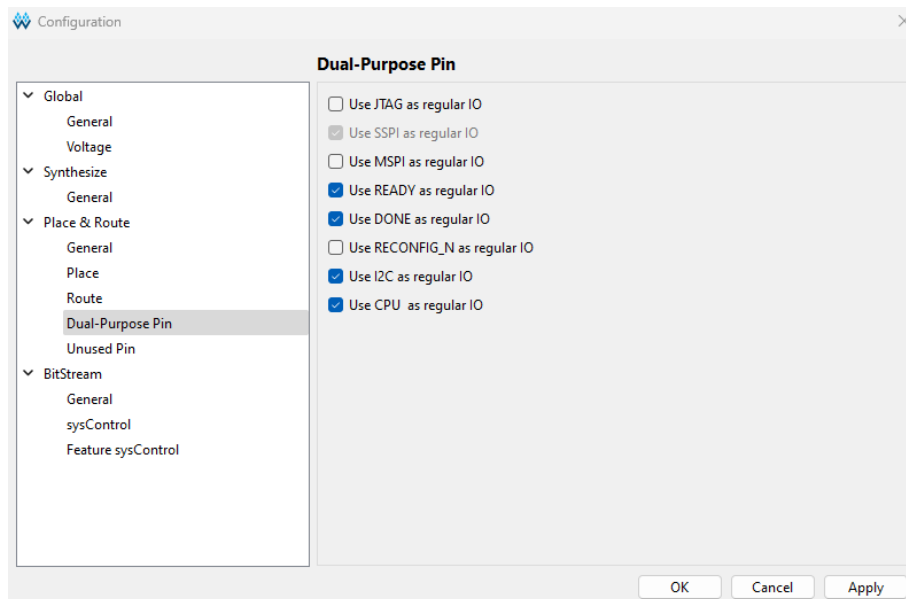
```

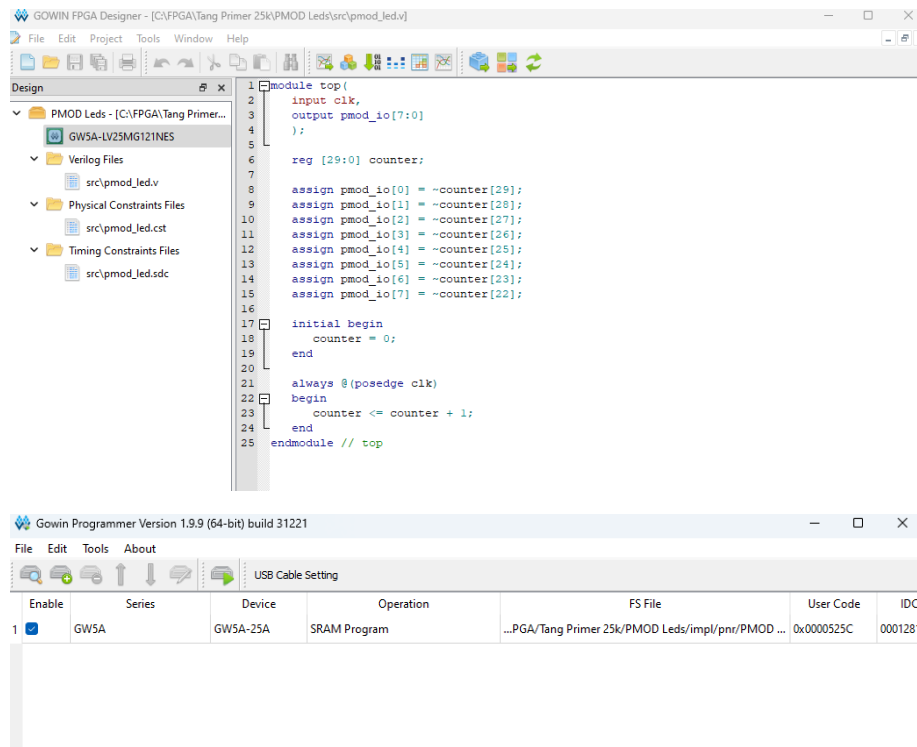
Timing constraints: **pmod\_led.sdc**

```
create_clock -name clk_50m -period 20 -waveform {0 10} [get_ports {clk}]
```

In Project / Configuration select “Dual-Purpose Pin”

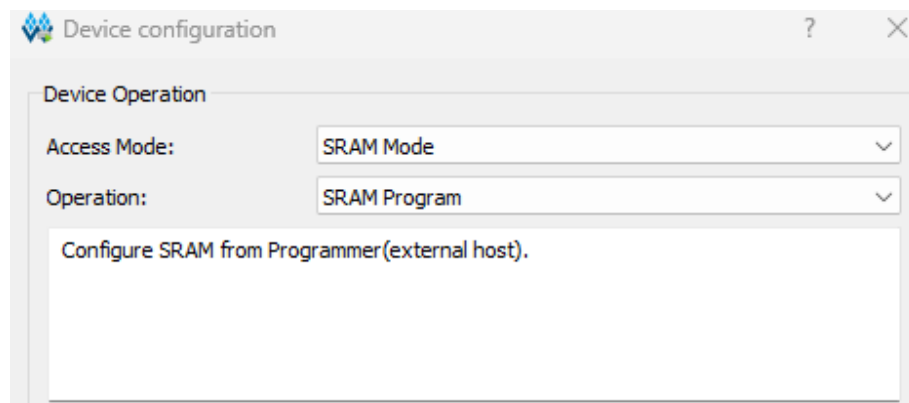
Select this for “Use CPU as regular IO” and click Apply





Select SRAM or External Flash mode

Programming SRAM



Programming flash

Device configuration

Device Operation

Access Mode: External Flash Mode SAT

Operation: exFlash Erase,Program SAT

exFlash Erase,Program SAT

Programming Options

File name: C:/FPGA/Tang Primer 25k/PMOD Leds/impl/pnr/PMOD Leds.fs

☐ User Flash Initialization

External Flash Options

Device: Generic Flash

Start Address: 0x000000

Save Cancel

# Lattice ICEBreaker 1.0e (OSS CAD Suite)

## Introduction

Open Command prompt

In oss-cad-suite run: **environment.bat**

Or from your directory: **c:\oss-cad-suite\environment.bat**

## Demo

Create a directory for your project.

### blink.v

```
/* Small test design actuating all IO on the iCEBreaker dev board. */
module top (
    input  CLK,

    output LED1,
    output LED2,
    output LED3,
    output LED4,
    output LED5,

    input  BTN_N,
    input  BTN1,
    input  BTN2,
    input  BTN3,

    output LEDR_N,
    output LEDG_N,

    output P1A1, P1A2, P1A3, P1A4, P1A7, P1A8, P1A9, P1A10,
    output P1B1, P1B2, P1B3, P1B4, P1B7, P1B8, P1B9, P1B10
);

    localparam BITS = 5;
    localparam LOG2DELAY = 22;

    reg [BITS+LOG2DELAY-1:0] counter = 0;
    reg [BITS-1:0] outcnt;

    always @(posedge CLK) begin
        counter <= counter + 1;
        outcnt <= counter >> LOG2DELAY;
    end

    assign {LED1, LED2, LED3, LED4, LED5} = outcnt ^ (outcnt >> 1);

    assign {LEDR_N, LEDG_N} = ~(!BTN_N + BTN1 + BTN2 + BTN3);

    assign {P1A1, P1A2, P1A3, P1A4, P1A7, P1A8, P1A9, P1A10,
            P1B1, P1B2, P1B3, P1B4, P1B7, P1B8, P1B9, P1B10} = 1 << (outcnt & 15);
endmodule
```

### blink.pcf

```
# 12 MHz clock
set_io  CLK      35
```

```

set_frequency CLK 12

# RS232
#set_io RX 6
#set_io TX 9

# LEDs and Button
set_io BTN_N 10
set_io LEDR_N 11
set_io LEDG_N 37

# RGB LED Driver
#set_io LED_RED_N 39
#set_io LED_GRN_N 40
#set_io LED_BLU_N 41
#set_io LED_RGB[0] 39
#set_io LED_RGB[1] 40
#set_io LED_RGB[2] 41

# SPI Flash
#set_io FLASH_SCK 15
#set_io FLASH_SSB 16
#set_io FLASH_IO0 14
#set_io FLASH_IO1 17
#set_io FLASH_IO2 12
#set_io FLASH_IO3 13

# PMOD 1A
set_io P1A1 4
set_io P1A2 2
set_io P1A3 47
set_io P1A4 45
set_io P1A7 3
set_io P1A8 48
set_io P1A9 46
set_io P1A10 44

# PMOD 1B
set_io P1B1 43
set_io P1B2 38
set_io P1B3 34
set_io P1B4 31
set_io P1B7 42
set_io P1B8 36
set_io P1B9 32
set_io P1B10 28

# PMOD 2
#set_io P2_1 27
#set_io P2_2 25
#set_io P2_3 21
#set_io P2_4 19
#set_io P2_7 26
#set_io P2_8 23
#set_io P2_9 20
#set_io P2_10 18

# LEDs and Buttons (PMOD 2)
set_io LED1 26
set_io LED2 27
set_io LED3 25
set_io LED4 23
set_io LED5 21
set_io BTN1 20
set_io BTN2 19
set_io BTN3 18

```

Run commands

```
yosys -p "read_verilog blink.v ; synth_ice40; write_json blink.json"
```

or

```
yosys -p "synth_ice40 -json blink.json" blink.v
```

then

```
nextpnr-ice40 --up5k --freq 25 --pcf blink.pcf --json blink.json --package sg48  
--asc bitstream.txt
```

```
icepack bitstream.txt bitstream.bin
```

Then to program the icebreaker

```
icaprogram bitstream.bin
```



# IceSugar (OSS CAD Suite)

## Introduction

Sample code here

<https://github.com/wuxx/icesugar/tree/master/src/basic/verilog/leds>

<https://github.com/wuxx/icesugar/blob/master/src/common/io.pcf>

## Demo

### leds.v

```
//light up the leds according to a counter to cycle through every one
/*
  b[25] b[24] b[23]
  0     0     0   black   (all off)
  0     0     1   red
  0     1     0   green
  0     1     1   yellow  (red + green)
  1     0     0   blue
  1     0     1   magenta (red + blue)
  1     1     0   cyan    (green + blue)
  1     1     1   white
*/

module top(input [3:0] SW, input clk, output LED_R, output LED_G, output LED_B);
  reg [25:0] counter;

  assign LED_R = ~counter[23];
  assign LED_G = ~counter[24];
  assign LED_B = ~counter[25];

  initial begin
    counter = 0;
  end

  always @(posedge clk)
  begin
    counter <= counter + 1;
  end
endmodule // top
```

### io.pcf

```
# For the iCESugar Board (iCE40UP5K-QFN48)

set_io LED_G 41
set_io LED_R 40
set_io LED_B 39

set_io SW[0] 18
set_io SW[1] 19
set_io SW[2] 20
set_io SW[3] 21
```

```

set_io clk      35

set_io RX       4
set_io TX       6

set_io USB_DP   10
set_io USB_DN   9
set_io USB_PULLUP 11

# PMOD 1
set_io P1_1     10
set_io P1_2     6
set_io P1_3     3
set_io P1_4     48
set_io P1_9     47
set_io P1_10    2
set_io P1_11    4
set_io P1_12    9

# PMOD 2
set_io P2_1     46
set_io P2_2     44
set_io P2_3     42
set_io P2_4     37
set_io P2_9     36
set_io P2_10    38
set_io P2_11    43
set_io P2_12    45

# PMOD 3
set_io P3_1     34
set_io P3_2     31
set_io P3_3     27
set_io P3_4     25
set_io P3_9     23
set_io P3_10    26
set_io P3_11    28
set_io P3_12    32

# PMOD 4
set_io P4_1     21
set_io P4_2     20
set_io P4_3     19
set_io P4_4     18

#spi
set_io SPI_SS   16
set_io SPI_SCK  15
set_io SPI_MOSI 17
set_io SPI_MISO 14

```

```

yosys -p "synth_ice40 -json leds.json" leds.v
nextpnr-ice40 --up5k --pcf io.pcf --json leds.json --package sg48 --asc
bitstream.txt
icepack bitstream.txt bitstream.bin

```

Copy to IceSugar drive

# Xilinx PicoBlaze

## Introduction

The main web page for PicoBlaze is here: <https://www.amd.com/en/products/adaptive-socs-and-fpgas/intellectual-property/picoblaze.html>

We need KPCSM3 for Spartan 3e, and KPCSM6 for Artix 7

<b>PicoBlaze releases for earlier device families are available below.</b>
PicoBlaze for UltraScale, 7 series, 6 series FPGAs >> <a href="#">KCPSM6 Rev 9 – September 30, 2014 (ZIP)</a>
PicoBlaze for Virtex 5 FPGAs >> <a href="#">KCPSM3 (ZIP)</a>
PicoBlaze for Spartan 3, Virtex 4, Virtex II and Virtex II Pro FPGAs >> <a href="#">KCPSM3 (ZIP)</a>
PicoBlaze for Virtex, Virtex E, Spartan II and Spartan IIE FPGAs >> <a href="#">XAPP213 (ZIP)</a>
PicoBlaze for Virtex II, Virtex II Pro FPGAs >> <a href="#">XAPP627 (ZIP)</a>
PicoBlaze for CoolRunner™ II CPLDs >> <a href="#">XAPP387 (ZIP)</a>

It has an assembler that requires 32 bit Windows, so for a 64 bit version you will need a variant of DOSBox – either DOSBox Staging <https://www.dosbox-staging.org> or DOSBox-X <https://dosbox-x.com/>

Or you can use the Python based replacement, opbasm

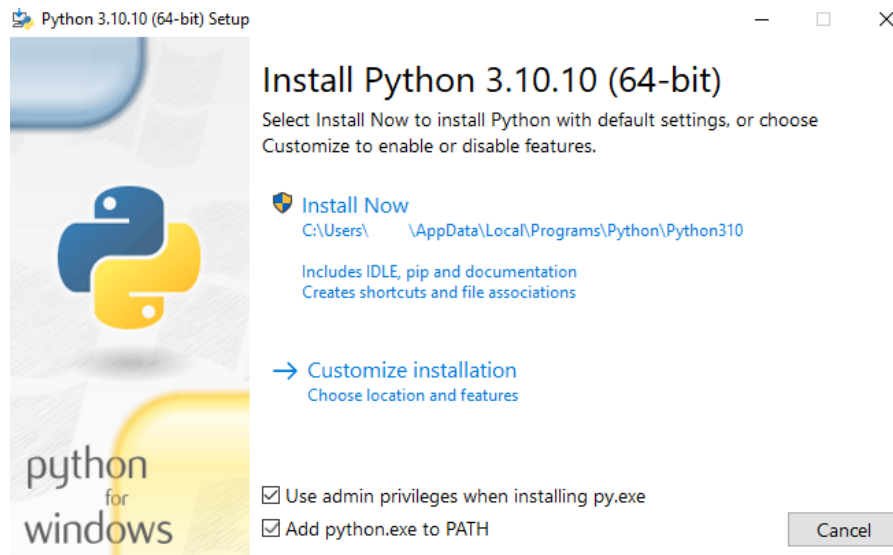
## KCPSM3 / 6 replacement opbasm

Opbasm is a replacement for the PicoBlaze KCPSM3 assembler, based on Python.  
Important links are:

<https://kevinpt.github.io/opbasm/>

<https://kevinpt.github.io/opbasm/#installation>

Install Python and select to install using administrator priveledges



Install opbasm using this command

```
pip install --upgrade https://github.com/kevinpt/opbasm/tarball/master
```

And use like this

```
opbasm -3 example.psm
```

In the directory being used you will need to **rom\_form** files (rom\_form.vhd and rom\_form.v). There are in the KCPSM3 download in the Assembler folder

cleanup	11/05/2025 15:48	Windows Batch File	1 KB
int_test.psm	11/05/2025 15:48	PSM File	2 KB
KCPSM3	11/05/2025 15:48	Application	89 KB
ROM_form.coe	11/05/2025 15:48	COE File	1 KB
ROM_form.v	11/05/2025 15:48	V File	15 KB
ROM_form	11/05/2025 15:48	Hard Disk Image F...	13 KB
unlock.psm	11/05/2025 15:48	PSM File	58 KB

## Build e3esk\_startup

We can test with the e3esk\_startup program from Spartan 3e

Using the Spartan 3e code as an example, we can covert control.psm into a VHDL file.

We need **control.psm** and **rom\_form.vhd**.

Add this line into **rom\_form.vhd** for proc\_reset otherwise you get an error in ISE.

```
entity {name} is
  Port (
    address : in  std_logic_vector(9 downto 0);
    instruction : out std_logic_vector(17 downto 0);
    proc_reset : out std_logic;
    clk : in  std_logic);
end {name};
```

Run opbasm

```
opbasm -3 control.psm
```

Then replace **control.vhd** in the e3esk\_startup program and rebuild it.

## Demo program (KCPSM3 – Spartan 3e)

Create a Picblaze assembly file **prog.psm**

```
start:
  INPUT  s0, 00 ; read switches into register s0
  OUTPUT s0, 01 ; write contents of s0 to output port 00 - leds.
  JUMP   start ; loop back to start
```

Ensure you have **ROM\_form.v** in your working directory

Run opbasm

```
opbasm -3 prog.psm
```

You will get output like this

```
OPBASM - Open PicoBlaze Assembler 1.3.10
Target architecture: PicoBlaze-3
Device configuration:
  Memory size: 1024, Scratchpad size: 64

Reading source: prog.psm

Assembling code...
SUCCESS

  3 instructions out of 1024 (0%)
  Highest occupied address: 002 hex

Found template:
  ROM_form.v

Writing output
  mem map: prog.mem
  log file: prog.log
  Verilog file: prog.v

Formatted source:
  prog.fmt
```

It will create **prog.v**

Then create **picoblaze\_led.v**

```
`timescale 1 ps / 1ps
```

```

module picoblaze_led (led, switch, clk);

    output reg [3:0] led;
    input      [3:0] switch;
    input      clk;

    // Signals used to connect KCPSM3 to program ROM and I/O logic
    wire [9:0] address;
    wire [17:0] instruction;
    wire [7:0] port_id;
    wire [7:0] out_port;
    reg [7:0] in_port;
    wire write_strobe;
    wire read_strobe;

    // Inserting KCPSM3 and the program memory
    kcpsm3 processor
    ( .address(address),
      .instruction(instruction),
      .port_id(port_id),
      .write_strobe(write_strobe),
      .out_port(out_port),
      .read_strobe(read_strobe),
      .in_port(in_port),
      .clk(clk));

    prog program
    ( .address(address),
      .instruction(instruction),
      .clk(clk));

    // Adding the output registers to the processor
    always @(posedge clk) begin
        if (port_id[0] == 0 && read_strobe == 1'b 1)
            begin
                in_port[3:0] <= switch;
            end

        if (port_id[0] == 1 && write_strobe == 1'b 1)
            begin
                led <= out_port[3:0];
            end
    end

endmodule

```

And create a **spartan3e.ucf** file

```

NET "clk" PERIOD = 20.0ns HIGH 50%;
NET "clk" LOC = "C9" | IOSTANDARD = LVTTTL;
#
NET "led<0>" LOC = "F12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
NET "led<1>" LOC = "E12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
NET "led<2>" LOC = "E11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
NET "led<3>" LOC = "F11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 4;
#
NET "switch<0>" LOC = "L13" | IOSTANDARD = LVTTTL | PULLUP;
NET "switch<1>" LOC = "L14" | IOSTANDARD = LVTTTL | PULLUP;
NET "switch<2>" LOC = "H18" | IOSTANDARD = LVTTTL | PULLUP;
NET "switch<3>" LOC = "N17" | IOSTANDARD = LVTTTL | PULLUP;

```

Then create a project to include **picoblaze\_led.v**, **spartan3e.ucf**, **kpcsm3.v** and **prog.v**

Synthesize, implement and then generate program file.

## Demo program (KCPSM6 – Nexys 4 DDR)

Create a Picblaze assembly file **prog.psm**

This is like the prior example but has a 1 second delay in processing, implemented with multiple loops in the Picoblaze assembly code.

```
CONSTANT inner_count, 0'd
CONSTANT middle_count, 206'd
CONSTANT outer_count, 236'd

CONSTANT switch_port, 0'd
CONSTANT led_port, 1'd

CONSTANT count_base, 10000'b

LOAD    s1, count_base

main_loop:
    INPUT  s0, switch_port    ; read switches into register s0
    OR     s0, s1
    OUTPUT s0, led_port        ; write contents of s0 to output port 80 - leds
    CALL   delay

    ADD    s1, count_base
    JUMP   NZ, skip_reset
    LOAD   s1, count_base

skip_reset:
    JUMP   main_loop            ; loop back to start

delay:
    LOAD   sE, inner_count
    LOAD   sD, middle_count
    LOAD   sC, outer_count

delay_loop:
    SUB    sE, 1                ; inner loop
    JUMP   NZ, delay_loop

    SUB    sD, 1                ; middle loop
    JUMP   NZ, delay_loop

    LOAD   sD, middle_count      ; outer loop
    SUB    sC, 1
    JUMP   NZ, delay_loop
    RETURN
```

Ensure you have **ROM\_form.v** in your working directory, copied from the KCPSM6 downloaded folder

Run opbasm

```
opbasm -6 prog.psm
```

You will get output like this

```

OPBASM - Open PicoBlaze Assembler 1.3.10
Target architecture: PicoBlaze-6
Device configuration:
    Memory size: 4096, Scratchpad size: 256

Reading source: prog.psm

Assembling code...
SUCCESS

    20 instructions out of 4096 (0%)
    Highest occupied address: 013 hex

Found template:
    ROM_form.v

Writing output
    mem map: prog.mem
    log file: prog.log
    Verilog file: prog.v

Formatted source:
    prog.fmt

```

It will create **prog.v**

Then create **picoblaze\_led.v**

```

`timescale 1 ps / 1ps

module picoblaze_led (LED, SW, CLK100MHZ);

    output reg [7:0] LED;
    input [7:0] SW;
    input CLK100MHZ;

    // Signals used to connect KCPSM6 to program ROM and I/O logic
    wire [11:0] address;
    wire [17:0] instruction;
    wire bram_enable;
    wire [7:0] port_id;
    wire [7:0] out_port;
    reg [7:0] in_port;
    wire write_strobe;
    wire k_write_strobe;
    wire read_strobe;
    wire interrupt; // changed to wire
    wire interrupt_ack;
    wire kcpsm6_sleep;
    wire kcpsm6_reset; // changed to wire

    assign kcpsm6_sleep = 1'b0;
    assign interrupt = 1'b0;

    kcpsm6 #(
        .interrupt_vector (12'h3FF),
        .scratch_pad_memory_size(64),
        .hwbuild (8'h00))
    processor (

```



```

        .address            (address),
        .instruction         (instruction),
        .bram_enable         (bram_enable),
        .port_id             (port_id),
        .write_strobe        (write_strobe),
        .k_write_strobe      (k_write_strobe),
        .out_port             (out_port),
        .read_strobe         (read_strobe),
        .in_port             (in_port),
        .interrupt            (interrupt),
        .interrupt_ack        (interrupt_ack),
        .reset                (kcpsm6_reset),
        .sleep                (kcpsm6_sleep),
        .clk                  (CLK100MHZ));

    // '7S', '2' and '1' for a Artix-7, Kintex-7 or Virtex-7 design.
    prog #(
        .C_FAMILY             ("7S"),    //Family 'S6' or 'V6' - use '7S' for Artix 7
        .C_RAM_SIZE_KWORDS   (2),        //Program size '1', '2' or '4'
        .C_JTAG_LOADER_ENABLE (1))        //Include JTAG Loader when set to '1'
    program_rom (
        .rdl                  (kcpsm6_reset),
        .enable                (bram_enable),
        .address               (address),
        .instruction            (instruction),
        .clk                    (CLK100MHZ));

    // Adding the output registers to the processor
    always @(posedge CLK100MHZ) begin
        if (port_id[0] == 0 && read_strobe == 1'b 1)
            begin
                in_port[7:0] <= SW;
            end

        if (port_id[0] == 1 && write_strobe == 1'b 1)
            begin
                LED <= out_port[7:0];
            end
    end
endmodule

```

And create a **Nexys4DDR.xdc** file

```

## Clock signal
set_property -dict { PACKAGE_PIN E3      IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ }];
#IO_L12P_T1_MRCC_35 Sch=clk100mhz
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports
{CLK100MHZ}];

##Switches
set_property -dict { PACKAGE_PIN J15      IOSTANDARD LVCMOS33 } [get_ports { SW[0] }];
set_property -dict { PACKAGE_PIN L16      IOSTANDARD LVCMOS33 } [get_ports { SW[1] }];
set_property -dict { PACKAGE_PIN M13      IOSTANDARD LVCMOS33 } [get_ports { SW[2] }];
set_property -dict { PACKAGE_PIN R15      IOSTANDARD LVCMOS33 } [get_ports { SW[3] }];
set_property -dict { PACKAGE_PIN R17      IOSTANDARD LVCMOS33 } [get_ports { SW[4] }];
set_property -dict { PACKAGE_PIN T18      IOSTANDARD LVCMOS33 } [get_ports { SW[5] }];
set_property -dict { PACKAGE_PIN U18      IOSTANDARD LVCMOS33 } [get_ports { SW[6] }];
set_property -dict { PACKAGE_PIN R13      IOSTANDARD LVCMOS33 } [get_ports { SW[7] }];

## LEDs
set_property -dict { PACKAGE_PIN H17      IOSTANDARD LVCMOS33 } [get_ports { LED[0] }];
set_property -dict { PACKAGE_PIN K15      IOSTANDARD LVCMOS33 } [get_ports { LED[1] }];
set_property -dict { PACKAGE_PIN J13      IOSTANDARD LVCMOS33 } [get_ports { LED[2] }];
set_property -dict { PACKAGE_PIN N14      IOSTANDARD LVCMOS33 } [get_ports { LED[3] }];

```

```
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { LED[4] }];
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { LED[5] }];
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { LED[6] }];
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { LED[7] }];
```

Then create a project to include **picoblaze\_led.v**, **Nexys4DDR.xdc**, **kpcsm6.v** and **prog.v**

Synthesize, implement and then generate program file.

## Demo program (pauloBlaze – Nexys 4 DDR)

There is an alternative implementatino of PicoBlaze by Paul Genssler, available here:

<https://github.com/krabo0om/pauloBlaze>

Download the zip file and find the **sources** folder – these are the files you will need.

This is a drop-in (mostly) replacement for KCPSM6.

If you have a KCPSM6 project already (as above) then the only changes are to include all of the pauloBlaze files and change the reference to KCPSM6 in your top level file to paulBlaze

Create an assembly file **prog.psm**

```
CONSTANT inner_count, 0'd
CONSTANT middle_count, 206'd
CONSTANT outer_count, 236'd

CONSTANT switch_port, 0'd
CONSTANT led_port, 1'd

CONSTANT count_base, 10000'b

LOAD    s1, count_base

main_loop:
    INPUT    s0, switch_port    ; read switches into register s0
    OR       s0, s1
    OUTPUT   s0, led_port        ; write contents of s0 to output port 80 - leds
    CALL     delay
    ADD      s1, count_base
    JUMP     NZ, skip_reset
    LOAD     s1, count_base

skip_reset:
    JUMP     main_loop            ; loop back to start

delay:
    LOAD     sE, inner_count
    LOAD     sD, middle_count
    LOAD     sC, outer_count

delay_loop:
    SUB      sE, 1                ; inner loop
    JUMP     NZ, delay_loop

    SUB      sD, 1                ; middle loop
    JUMP     NZ, delay_loop

    LOAD     sD, middle_count     ; outer loop
    SUB      sC, 1
    JUMP     NZ, delay_loop
```

```
RETURN
```

Ensure you have **ROM\_form.v** in your working directory, copied from the KCPSM6 downloaded folder

Run opbasm

```
opbasm -6 prog.psm
```

You will get output like this

```
OPBASM - Open PicoBlaze Assembler 1.3.10
Target architecture: PicoBlaze-6
Device configuration:
  Memory size: 4096, Scratchpad size: 256

Reading source: prog.psm

Assembling code...
SUCCESS

  20 instructions out of 4096 (0%)
  Highest occupied address: 013 hex

Found template:
  ROM_form.v

Writing output
  mem map: prog.mem
  log file: prog.log
  Verilog file: prog.v

Formatted source:
  prog.fmt
```

It will create **prog.v**

Then create **pauloblaze\_led.v** – the only change from **picoblaze\_led.v** above is to change **KPCSM6** to **pauloBlaze**

```
`timescale 1 ps / 1ps

module picoblaze_led (LED, SW, CLK100MHZ);

  output reg [7:0] LED;
  input [7:0] SW;
  input CLK100MHZ;

  // Signals used to connect KCPSM6 to program ROM and I/O logic
  wire [11:0] address;
  wire [17:0] instruction;
  wire bram_enable;
  wire [7:0] port_id;
  wire [7:0] out_port;
  reg [7:0] in_port;
```

```

wire          write_strobe;
wire          k_write_strobe;
wire          read_strobe;
wire          interrupt;           // changed to wire
wire          interrupt_ack;
wire          kcpasm6_sleep;
wire          kcpasm6_reset;      // changed to wire

assign kcpasm6_sleep = 1'b0;
assign interrupt = 1'b0;

pauloBlaze #(
    .interrupt_vector      (12'h3FF),
    .scratch_pad_memory_size(64),
    .hwbuild               (8'h00))
processor (
    .address               (address),
    .instruction            (instruction),
    .bram_enable           (bram_enable),
    .port_id               (port_id),
    .write_strobe          (write_strobe),
    .k_write_strobe        (k_write_strobe),
    .out_port              (out_port),
    .read_strobe           (read_strobe),
    .in_port               (in_port),
    .interrupt             (interrupt),
    .interrupt_ack         (interrupt_ack),
    .reset                 (kcpasm6_reset),
    .sleep                 (kcpasm6_sleep),
    .clk                   (CLK100MHZ));

    // '7S', '2' and '1' for a Artix-7, Kintex-7 or Virtex-7 design.
prog #(
    .C_FAMILY              ("7S"),    //Family 'S6' or 'V6' - use '7S' for Artix 7
    .C_RAM_SIZE_KWORDS     (2),      //Program size '1', '2' or '4'
    .C_JTAG_LOADER_ENABLE   (1))      //Include JTAG Loader when set to '1'
program_rom (
    .rdl                   (kcpasm6_reset),
    .enable                (bram_enable),
    .address               (address),
    .instruction            (instruction),
    .clk                   (CLK100MHZ));

// Adding the output registers to the processor
always @(posedge CLK100MHZ) begin
    if (port_id[0] == 0 && read_strobe == 1'b 1)
        begin
            in_port[7:0] <= SW;
        end

    if (port_id[0] == 1 && write_strobe == 1'b 1)
        begin
            LED <= out_port[7:0];
        end
    end
endmodule

```

And create a **Nexys4DDR.xdc** file

```

## Clock signal
set_property -dict { PACKAGE_PIN E3      IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ }];
#IO_L12P_T1_MRCC_35 Sch=clk100mhz
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports
{CLK100MHZ}];

```

```

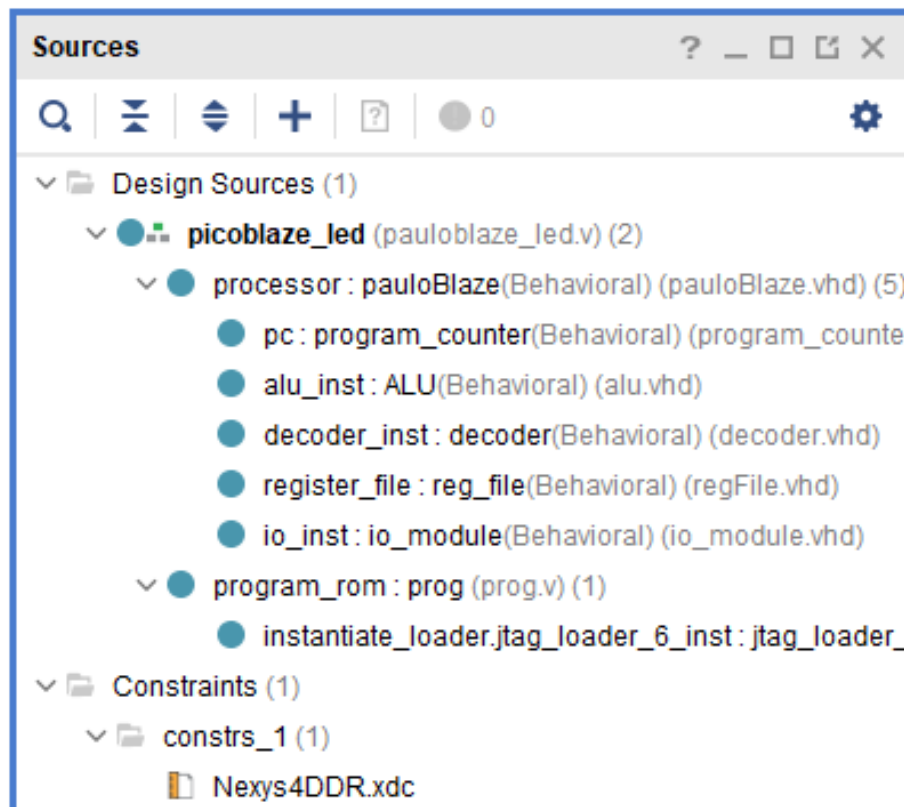
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { SW[0] }];
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { SW[1] }];
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { SW[2] }];
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { SW[3] }];
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { SW[4] }];
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { SW[5] }];
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { SW[6] }];
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { SW[7] }];

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { LED[0] }];
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { LED[1] }];
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { LED[2] }];
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { LED[3] }];
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { LED[4] }];
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { LED[5] }];
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { LED[6] }];
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { LED[7] }];

```

From the paulBlaze folder, copy these items (all of them in the sources folder): **alu.vhd**, **decoder.vhd**, **io\_module.vhd**, **op\_code.vhd**, **pauloBlaze.vhd**, **program\_counter.vhd**, **refFile.vhd**

Then create a project to include **pauloblaze\_led.v**, **Nexys4DDR.xdc**, **prog.v** and all the files above.



Synthesize, implement and then generate program file.

# Appendix A: Demo program links

## Example programs

DK\_START\_GW2A-LV18PG256C8I7:

Tang Nano 9k: <https://github.com/sipeed/TangNano-9K-example>

Tang Nano 20l: <https://github.com/sipeed/TangNano-20K-example>

Tang Primer 25k: <https://github.com/sipeed/TangPrimer-25K-example>

Basys 3: <http://digilent.com/reference/programmable-logic/basys-3/demos/gpio>

Nexys 4 DDR: <https://github.com/Digilent/Nexys-4-DDR-OOB>

Arty S7: <https://github.com/Digilent/Arty-S7-50-GPIO>

Spartan 3e:

[https://web.archive.org/web/20090323004714/http://www.xilinx.com/products/boards/s3estarter/files/s3esk\\_startup.zip](https://web.archive.org/web/20090323004714/http://www.xilinx.com/products/boards/s3estarter/files/s3esk_startup.zip)

Nexys: [https://digilent.com/reference/\\_media/nexys/nexys\\_bist.zip](https://digilent.com/reference/_media/nexys/nexys_bist.zip)

Icebreaker 1.0e:

IceSugar:

## Appendix B: Useful links

### OSS CAS Suite

<https://github.com/YosysHQ/oss-cad-suite-build>

<https://github.com/YosysHQ/oss-cad-suite-build/releases/tag/2025-03-18>

<https://www.zenembed.com/ice40>

### Trial project for an IceBreaker 1.0e

<https://clifford.at/icestorm>

[https://blog.waynejohnson.net/doku.php/open\\_source\\_on\\_goboard](https://blog.waynejohnson.net/doku.php/open_source_on_goboard)

<https://github.com/YosysHQ/nextpnr>

<https://gojimmypi.blogspot.com/2020/08/icebreaker-bitsy-fpga-early-adopter.html>

### IceSugar

<https://github.com/wuxx/icesugar/>

<https://github.com/wuxx/icesugar/blob/master/src/common/io.pcf>

### Sipeed Nano getting started guides

Nano 20k: <https://wiki.sipeed.com/hardware/en/tang/tang-nano-20k/example/led.html>

Nano 9k: <https://wiki.sipeed.com/hardware/en/tang/Tang-Nano-9K/examples/led.html>

### Tutorials

ISE: [https://faculty.kfupm.edu.sa/COE/aimane/coe405/Tutorial\\_Nexys3\\_ISE14\\_7.pdf](https://faculty.kfupm.edu.sa/COE/aimane/coe405/Tutorial_Nexys3_ISE14_7.pdf)

Spartan 3e Getting Started: <https://therobotfix.wordpress.com/2011/06/27/getting-started-with-spartan-3e-fpga-and-verilog/>

Spartan 3e LCD: <https://therobotfix.wordpress.com/2011/06/28/controlling-the-spartan-3e-lcd/>

Spartan 3e DAC: <https://therobotfix.wordpress.com/2011/06/29/controlling-the-sparatan-3e-dac/>

Nexys4 SPI Flash: <http://blog.penkovsky.com/2014/10/how-to-program-spi-flash-memory-on-xilinx-nexys4.html>

## Digilent getting started guides

Vivado: [https://digilent.com/reference/vivado/getting\\_started/2018.2](https://digilent.com/reference/vivado/getting_started/2018.2)

Vivado: [https://digilent.com/reference/vivado/getting\\_started/start](https://digilent.com/reference/vivado/getting_started/start)

Basys 3: <https://digilent.com/reference/learn/programmable-logic/tutorials/basys-3-getting-started/>

Basys 3: <https://digilent.com/reference/learn/programmable-logic/tutorials/basys-3-programming-guide/start>

Arty S7: <https://digilent.com/reference/learn/programmable-logic/tutorials/arty-programming-guide/start>

General links: <https://digilent.com/reference/learn/programmable-logic/tutorials/start>

Spartan HDL modules: [https://my.eng.utah.edu/~cs3710/xilinx-docs/spartan3e\\_hdl.pdf](https://my.eng.utah.edu/~cs3710/xilinx-docs/spartan3e_hdl.pdf)

## Digilent board wikis

Board definition files: <https://github.com/Digilent/vivado-boards/>

XDC files: <https://github.com/Digilent/digilent-xdc>

Nexys 4 DDR: <https://digilent.com/reference/programmable-logic/nexys-4-ddr/start>

Nexys 4 DDR with Flash: <https://digilent.com/reference/learn/programmable-logic/tutorials/nexys-4-ddr-programming-guide/start>

Basys 3: <https://digilent.com/reference/programmable-logic/basys-3/start>

Arty S7: <https://digilent.com/shop/arty-s7-spartan-7-fpga-development-board>

Spartan-3E Libraries Guide for HDL Designs: <https://digilent.com/reference/programmable-logic/spartan-3e/start>

Spartan 3e reference: [https://my.eng.utah.edu/~cs3710/xilinx-docs/spartan3e\\_hdl.pdf](https://my.eng.utah.edu/~cs3710/xilinx-docs/spartan3e_hdl.pdf)

Nexys Digital System Builder: <https://digilent.com/reference/nexys/nexys>

## Cable driver locations

C:\Xilinx\Vivado\2018.2\data\xicom\cable\_drivers\nt64\digilent

## Picoblaze

<https://www-users.york.ac.uk/~mjf5/bomb/index.html>



# Appendix C: Zadig and USB Serial drivers

## Reverting from the Zadig drivers

The Zadig driver will remove the FTDI drivers, and this may stop other boards from working properly. To resolve this you need to switch between drivers.

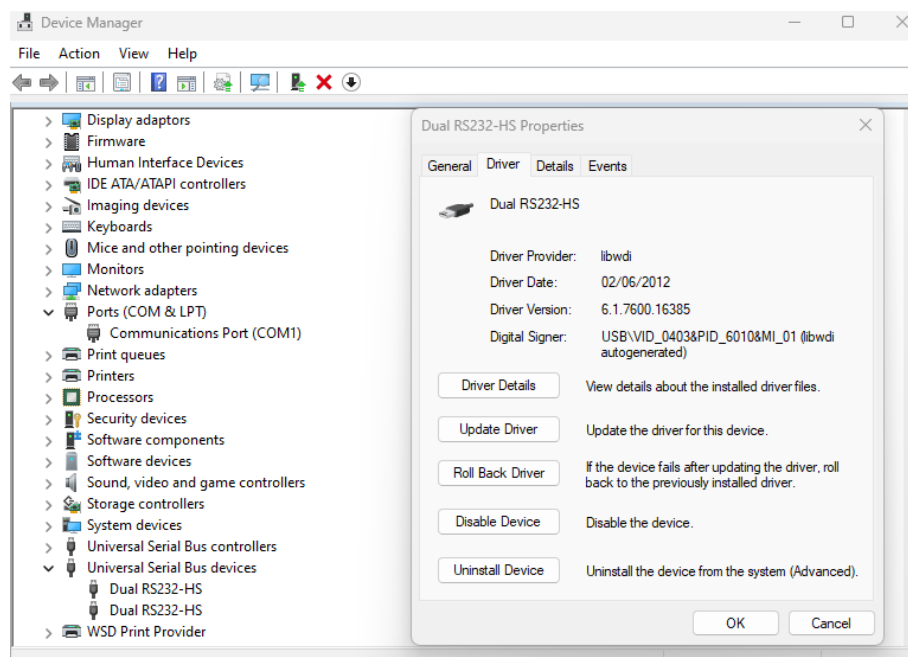
There are two approaches to this – either select an older driver within Device Manager, or delete the driver installed by Zadig (preferably using pnputil)

The FTDI drivers use these ids:

VID 0403

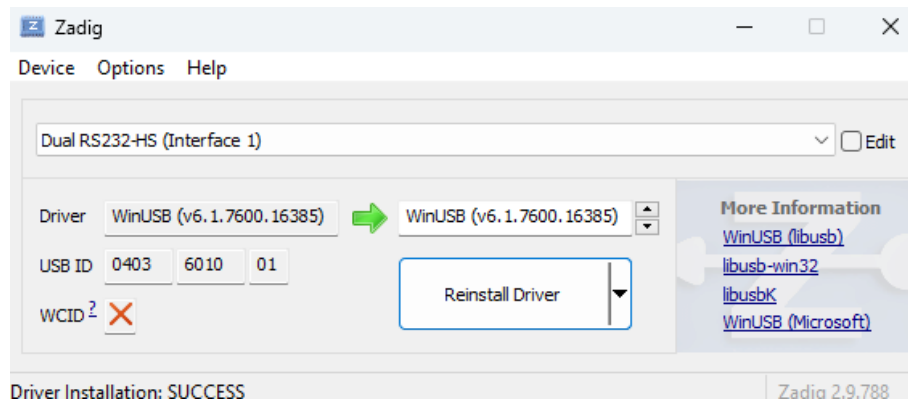
PID 6010

## Change to another driver



Select the device (“Dual RS232-HS”)

If unsure which device, Zadig shows the name of the device as in this image



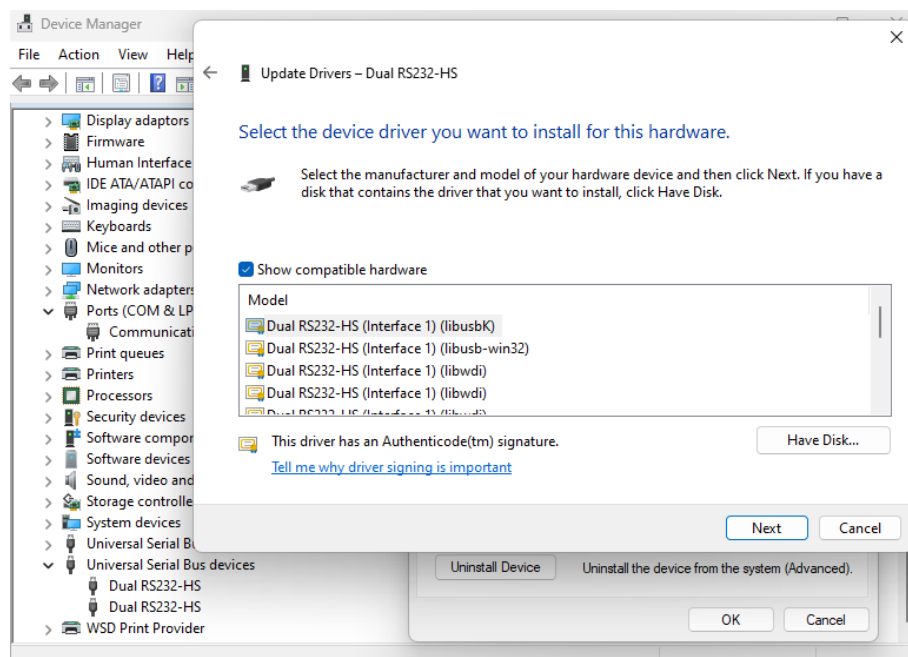
Right click for Properties

Select Driver tab, then Update Driver option

Select “Browse my computer for drivers”

Select “Let me pick from a list of available drivers on my computer”

Then select USB Serial Converter B Version: 2.12.36.20 or similar



## Delete the Zadig driver

Find the oemXXX.inf files for each device to be deleted: there will probably be two USB Serial Converter A, USB Serial Converter B and one COM port.

You can detect which entries ones by plugging in your device with Device Manager open

Open Device Manager

Select each device

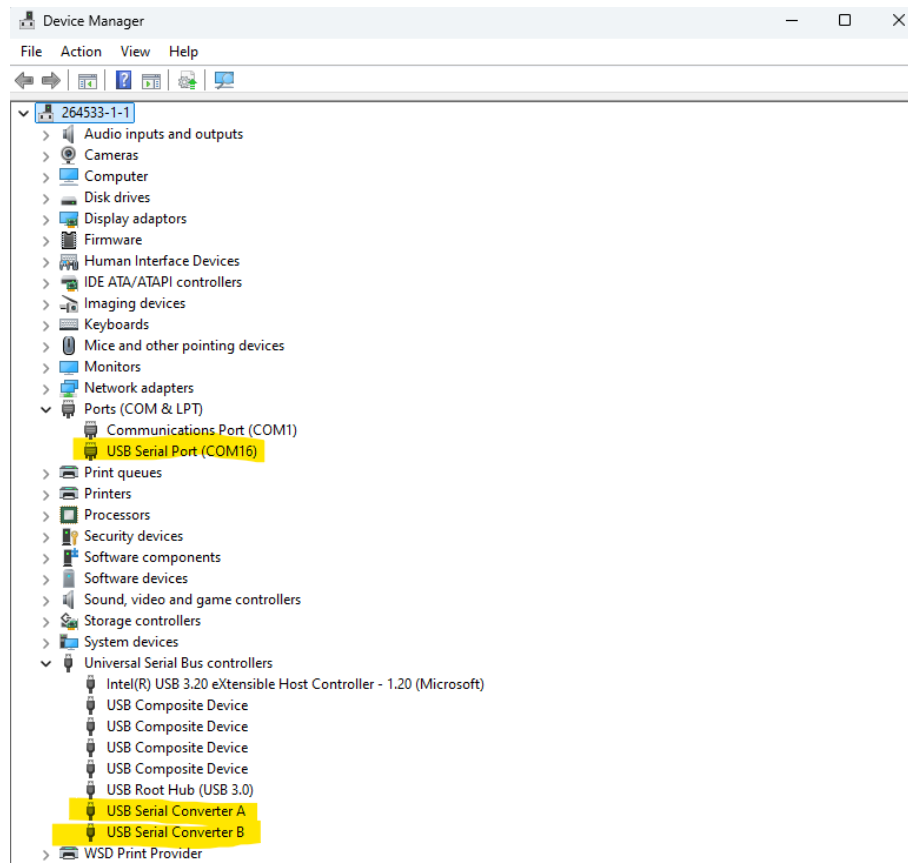
Right click

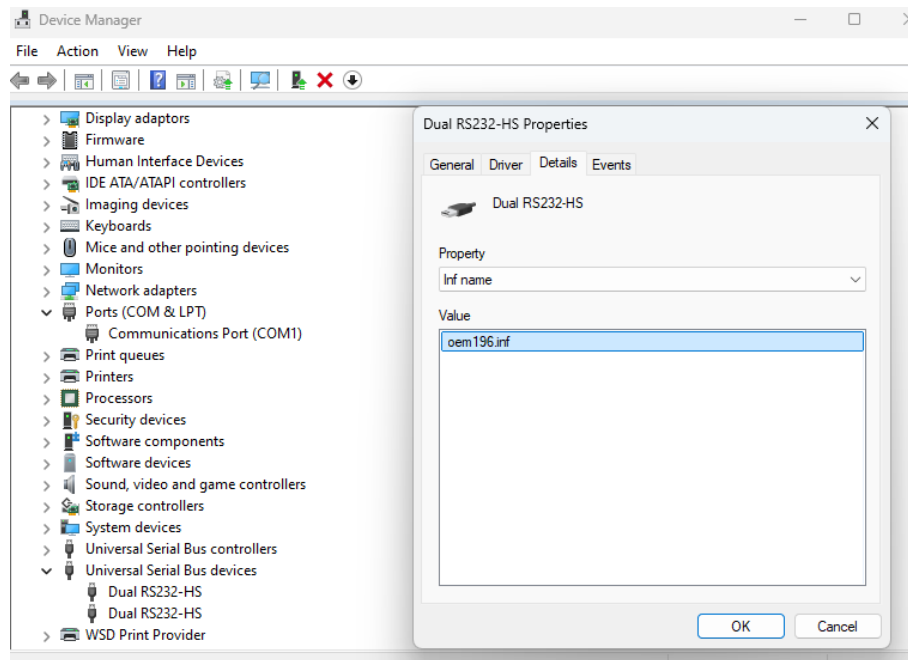
Select Properties

Select Details tab

Select “Inf name” in the Property drop-down

Note down the value (in the form oemXXX.inf)





Delete these oemXXX.inf drivers

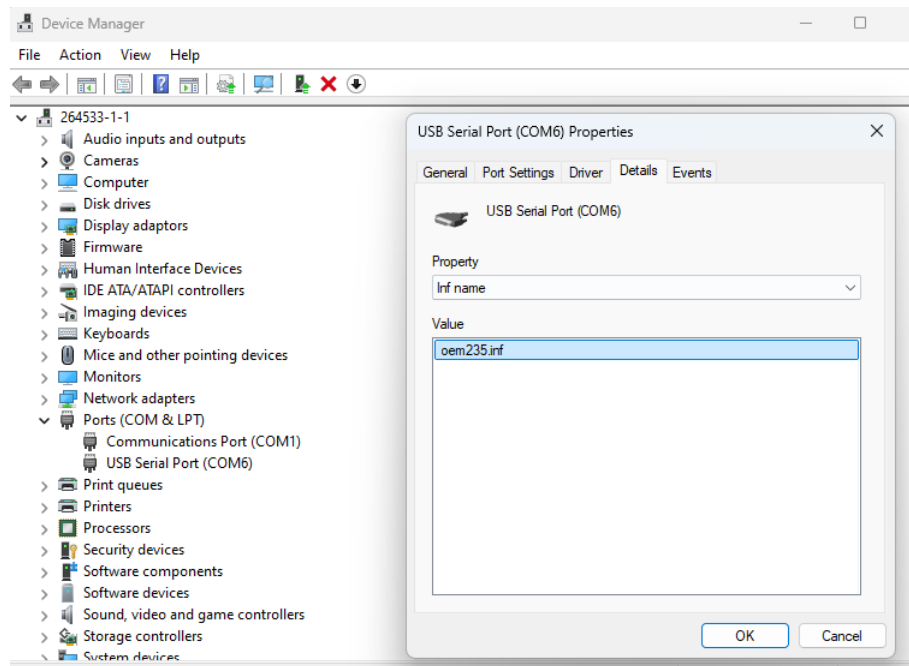
```
> pnputil /delete-driver oemXXX.inf
```

Example:

```
pnputil /delete-driver oem235.inf  
pnputil /delete-driver oem234.inf
```

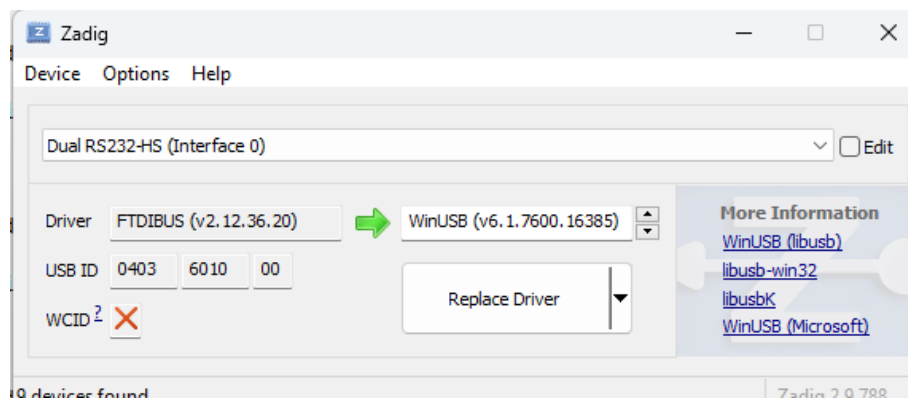
## Seeing how the driver changes when installing Zadig

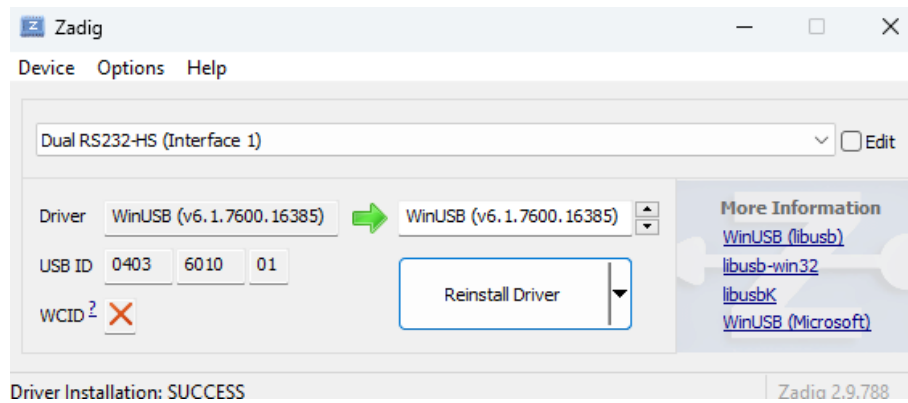
Before installation



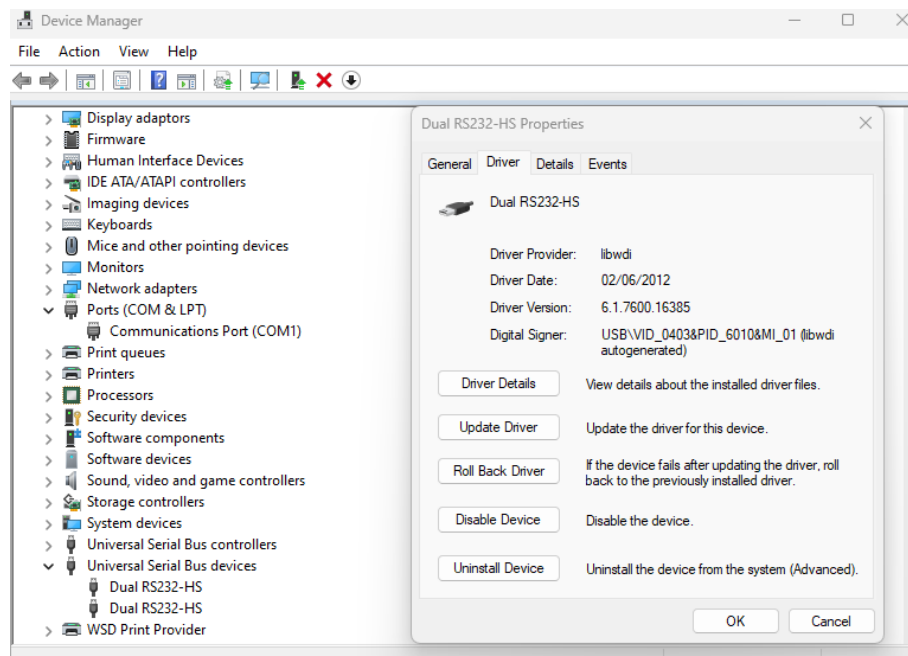
Instance ID:	FTDIBUS\VID_0403+PID_6010+210352B9E6CEB\0000
Device Description:	USB Serial Port (COM16)
Class Name:	Ports
Class GUID:	{4d36e978-e325-11ce-bfc1-08002be10318}
Manufacturer Name:	FTDI
Status:	Disconnected
Driver Name:	oem235.inf

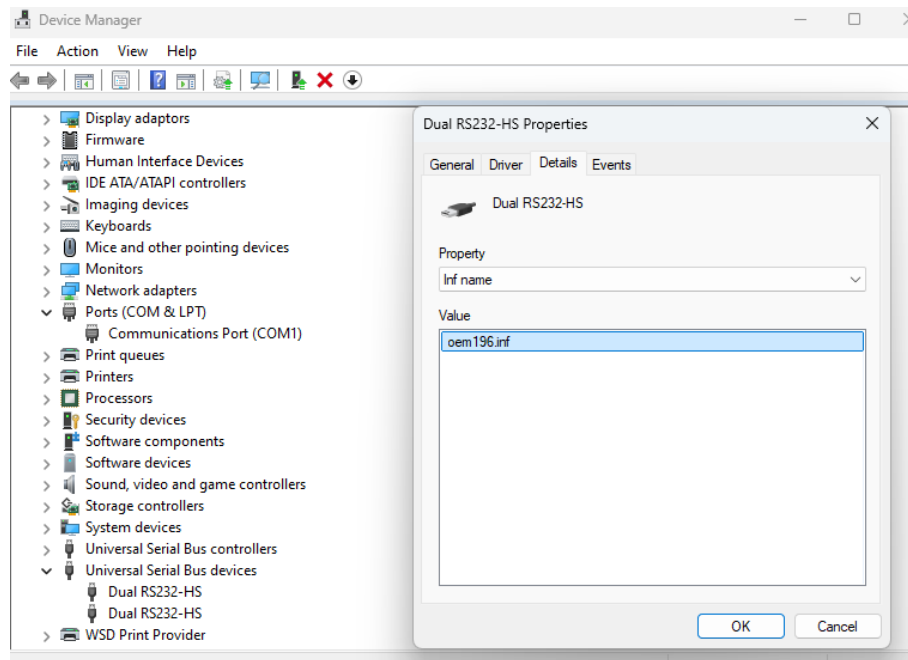
Run Zadig





Then check in Device Manager





Instance ID:  
 USB\VID\_0403&PID\_6010&MI\_01\6&2448050f&0&0001  
 Device Description: Dual RS232-HS  
 Class Name: USBDevice  
 Class GUID: {88bae032-5a81-49f0-bc3d-a4ff138216d6}  
 Manufacturer Name: Future Technology Devices International, Ltd  
 Status: Started  
 Driver Name: oem196.inf

## Ways to dump all PNP drivers

```

pnputil /enum-drivers >drivers.txt
notepad drivers.txt
  
```

## USB chips in use

Board	USB chip
DK_START_GW2A-LV18PG256C8I7	FT2232HL
Tang Nano 9k	BL702 (emulating FT2232D)
Tang Nano 20k	BL616 (emulating FT2232)
Tang Primer 25k	BL616 (emulating FT2232)
Basys 3	FT2232HQ
Nexsys 4 DDR	FT2232HQ
Arty S7	FT2232HQ
Spartan 3e Starter Board	VID 03FD PID 000D Programming Cables / Xilinx Embedded Platform USB Firmware Loader [winusb.sys]
Spartan 3 Digital System Builder	VID 1443 PID 0005 Digilent Onboard USB

	[dmodusb.sys]
Icebreaker 1.0e	FT2232H

Most are either FTDI chips or emulate the FT2232 chip.

## Useful web sites

<https://github.com/pbatard/libwidi/wiki/Zadig>

<https://github.com/pbatard/libwidi/wiki/faq#user-content-Zadig>

<https://www.anoopcnair.com/pnputil-driver-manager-tool-install-drivers/>

<https://mirobetm.blogspot.com/2014/08/removing-libwidi-drivers.html>

<https://learn.microsoft.com/en-us/windows-hardware/drivers/devtest/pnputil-command-syntax>

<https://answers.microsoft.com/en-us/windows/forum/all/bluetooth-keyboard-worked-fine-for-a-year-now-dead/1e8bbcf0-e701-4d9e-833c-c7ef0ad227c3>

<https://superuser.com/questions/1719698/prevent-windows-10-from-installing-onenote-printer>

[https://ftdichip.com/wp-content/uploads/2022/05/AN\\_396-FTDI-Drivers-Installation-Guide-for-Windows-10\\_11.pdf](https://ftdichip.com/wp-content/uploads/2022/05/AN_396-FTDI-Drivers-Installation-Guide-for-Windows-10_11.pdf)