Department of Design Engineering and Mathematics

Middlesex University

Coursework 3: MicroBlaze and embedding Custom IP with Nexys Board

Dr Purav Shah

Tek Bahadur Pun

M00834324

PDE3111 – SoC Design & Implementation

BEng Electronic Engineering

January 2024

Tabel of Contents

1. Introduction

The purpose of this report is to set up a customized Digital Stopwatch IP on vivado and implement it with the MicroBlaze IP core. This report will also outline the technical process of implementing the custom Digital Stopwatch IP AXI block into the MicroBlaze processor in vivado and writing the software needed using Vitis to regulate the hardware's operation.

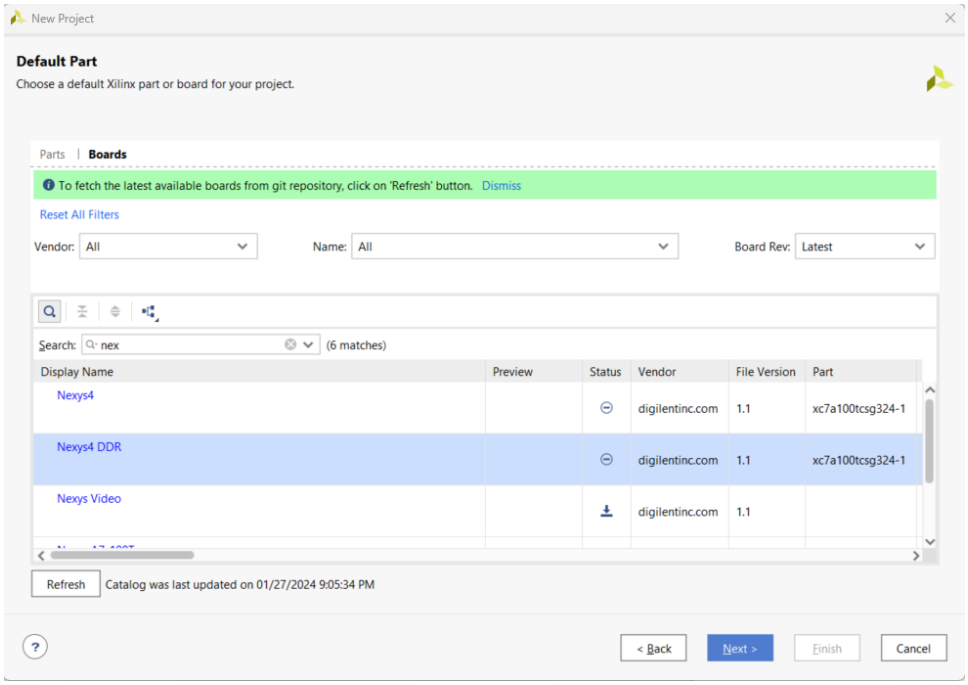
2. Materials and Experimental Setup

- Development board
  - Nexys 4 DDR
- FPGA Development
  - Vivado Design
- Software Development
  - Vitis Platform

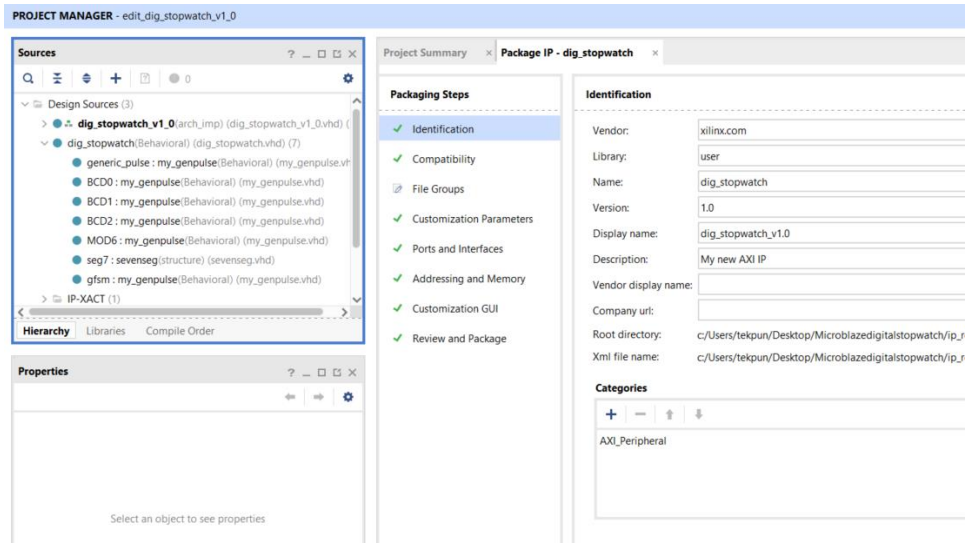
Vivado will be used to create the custom Digital Stopwatch IP and later implemented with the Microblaze IP Core with necessary components to generate a bitstream to program the board to check correct working inputs and outputs on the board. After bitstream generation and necessary checks, XSA file to be imported on the Vitis software for software development.

3. Experimental Procedure & Analysis

3.1. FPGA Development



It is important to pick the correct board that is being used in the project, for this case, the board used was Nexys4 DDR.



New dig_stopwatch package ip created added with the provided files, dig_stopwatch.vhd, my_genpluse.vhd and sevenseg.vhd as design source files in the project.

```
entity dig_stopwatch_v1_0_S00_AXI is
    generic (
        -- Users to add parameters here

        -- User parameters ends
        -- Do not modify the parameters beyond this line

        -- Width of S_AXI data bus
        C_S_AXI_DATA_WIDTH  : integer   := 32;
        -- Width of S_AXI address bus
        C_S_AXI_ADDR_WIDTH  : integer   := 4
    );
    port (
        -- Users to add ports here

          pause: in std_logic;
          segs: out std_logic_vector (6 downto 0);
          EN: out std_logic_vector (7 downto 0);

        -- User ports ends
        -- Do not modify the ports beyond this line

        -- Global Clock Signal
        S_AXI_ACLK  : in std_logic;
        -- Global Reset Signal. This Signal is Active LOW
        S_AXI_ARESETN   : in std_logic;
        -- Write address (issued by master, acceped by Slave)
        S_AXI_AWADDR    : in std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
```

Added ports from dig_stopwatch to digi_stopwatch_v1_0_S00_AXI_inst file

```
architecture arch_imp of dig_stopwatch_v1_0_S00_AXI is

        component dig_stopwatch is
            port (resetn, clock, pause: in std_logic;
                  segs: out std_logic_vector (6 downto 0);
                    EN: out std_logic_vector (7 downto 0));
        end component;

    -- AXI4LITE signals
    signal axi_awaddr   : std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
    signal axi_awready  : std_logic;
    signal axi_wready   : std_logic;
    signal axi_bresp    : std_logic_vector(1 downto 0);
    signal axi_bvalid   : std_logic;
```

Component declaration for dig_stopwatch in digi_stopwatch_v1_0_S00_AXI_inst file

```
        -- Add user logic here

UUI: dig_stopwatch
port map (
    resetn =>S_AXI_ARESETN,
    clock => S_AXI_ACLK,
    pause => pause,
    segs=>segs,
    EN => EN
    );

    -- User logic ends

end arch_imp;
```

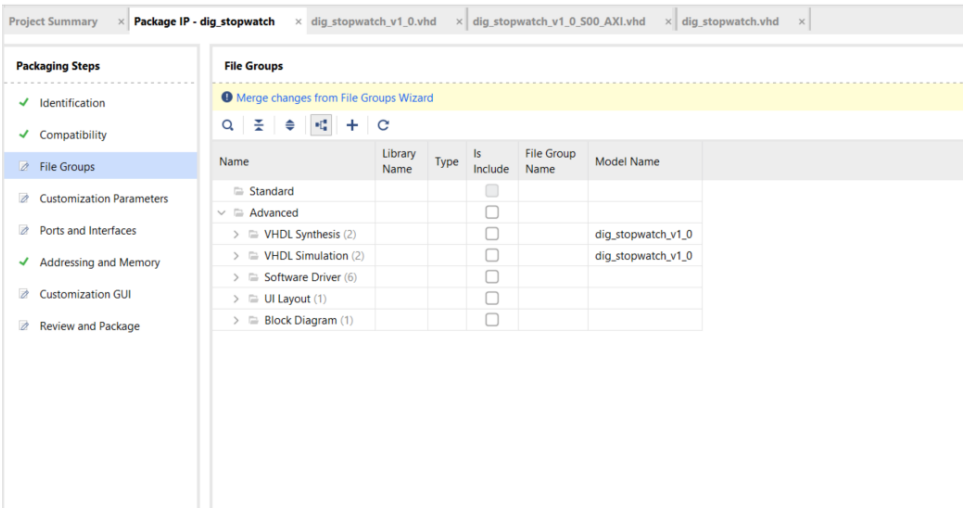
Port mapping dig_stopwatch components to signals in digi_stopwatch_v1_0_S00_AXI_inst file

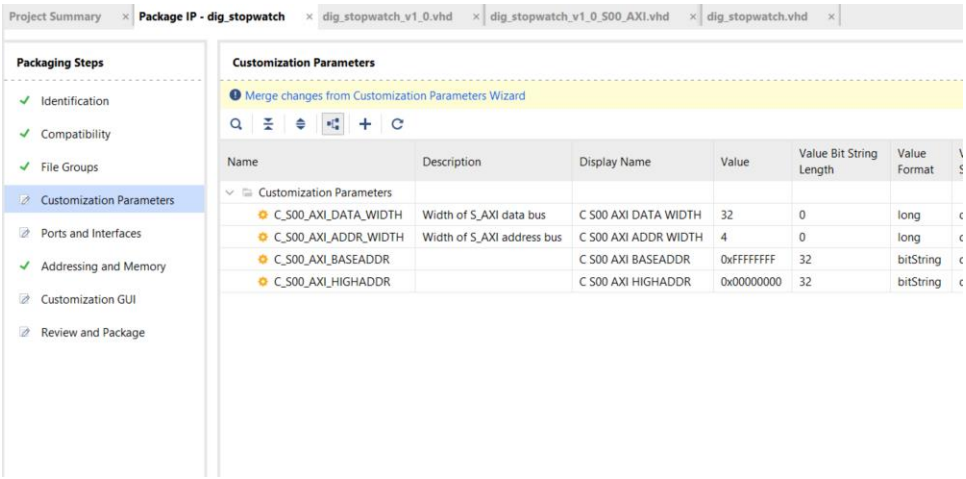
```
13        -- Parameters of Axi Slave Bus Interface S00_AXI
14        C_S00_AXI_DATA_WIDTH    : integer   := 32;
15        C_S00_AXI_ADDR_WIDTH    : integer   := 4
16    );
17    port (
18        -- Users to add ports here
19
20          pause: in std_logic;
21          segs: out std_logic_vector (6 downto 0);
22          EN: out std_logic_vector (7 downto 0);
23
24        -- User ports ends
25        -- Do not modify the ports beyond this line
26
27
28        -- Ports of Axi Slave Bus Interface S00_AXI
29        s00_axi_aclk    : in std_logic;
30        s00_axi_aresetn : in std_logic;
31        s00_axi_awaddr  : in std_logic_vector(C_S00_AXI_ADDR_WIDTH-1 downto 0);
```

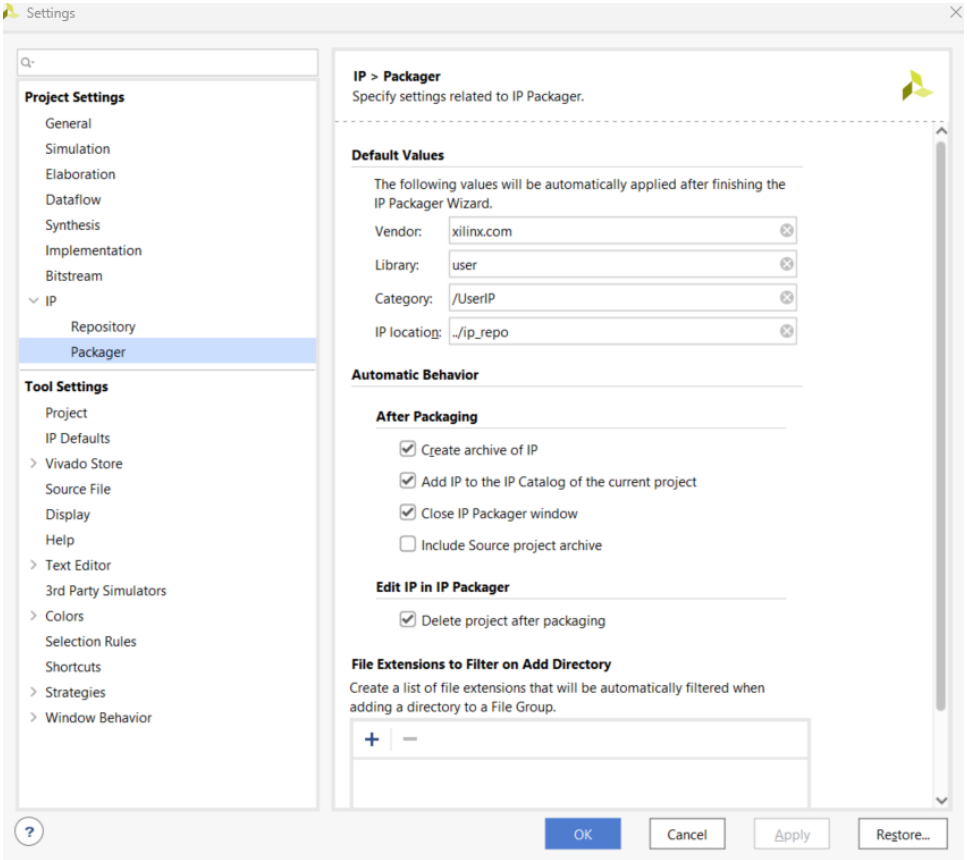
Added ports to digi_stopwatch_v1_0_arch_imp file

```
architecture arch_imp of dig_stopwatch_v1_0 is

    -- component declaration
    component dig_stopwatch_v1_0_S00_AXI is
        generic (
        C_S_AXI_DATA_WIDTH  : integer   := 32;
        C_S_AXI_ADDR_WIDTH  : integer   := 4
        );
        port (
          segs: out std_logic_vector (6 downto 0);
          EN: out std_logic_vector (7 downto 0);
          pause: in std_logic;
        S_AXI_ACLK  : in std_logic;
        S_AXI_ARESETN   : in std_logic;
        S_AXI_AWADDR    : in std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
        S_AXI_AWPROT    : in std_logic_vector(2 downto 0);
        S_AXI_AWVALID   : in std_logic;
```

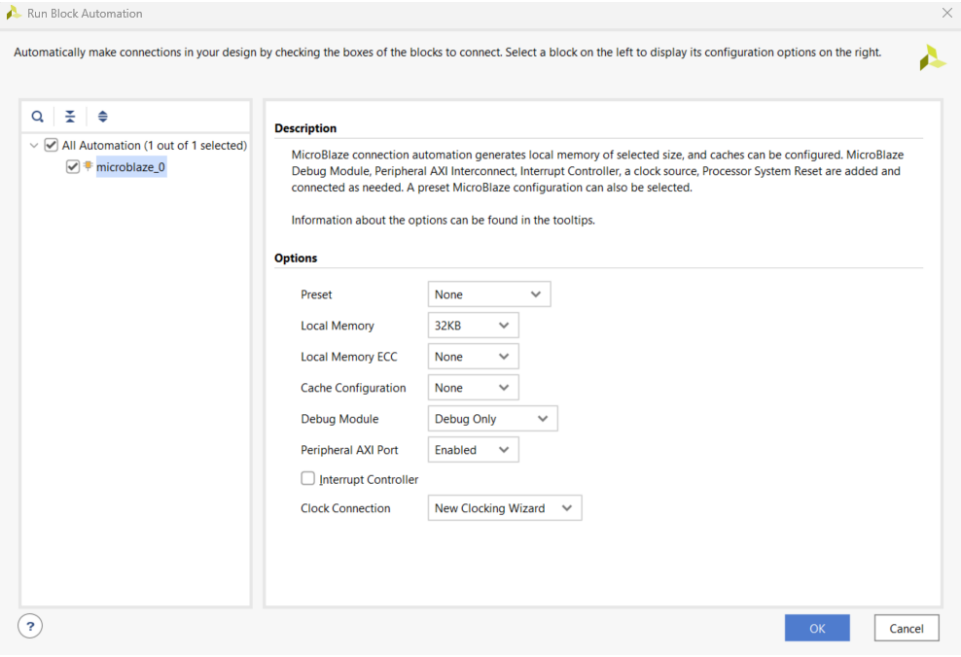
Added port component to digi_stopwatch_v1_0_arch_imp file

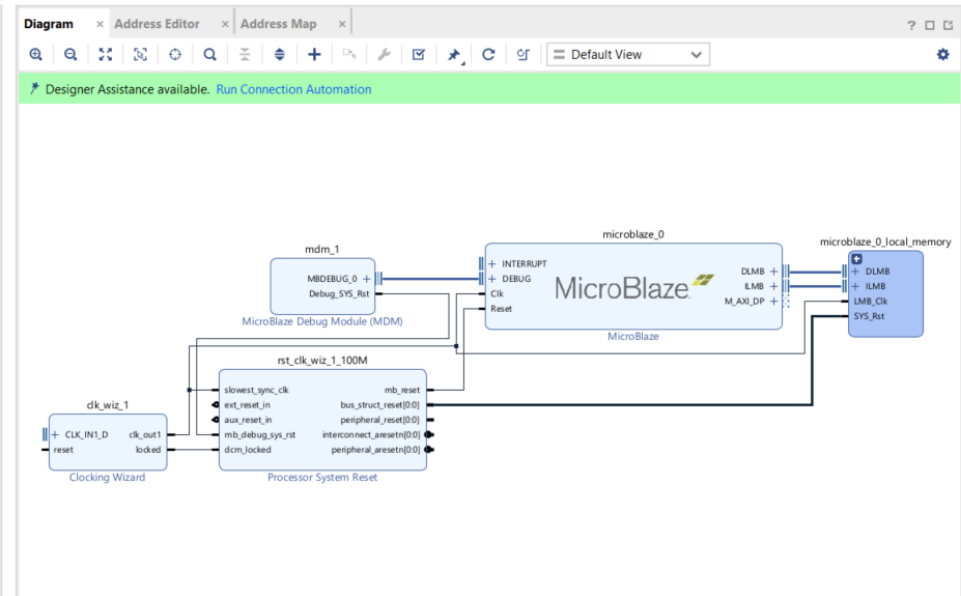




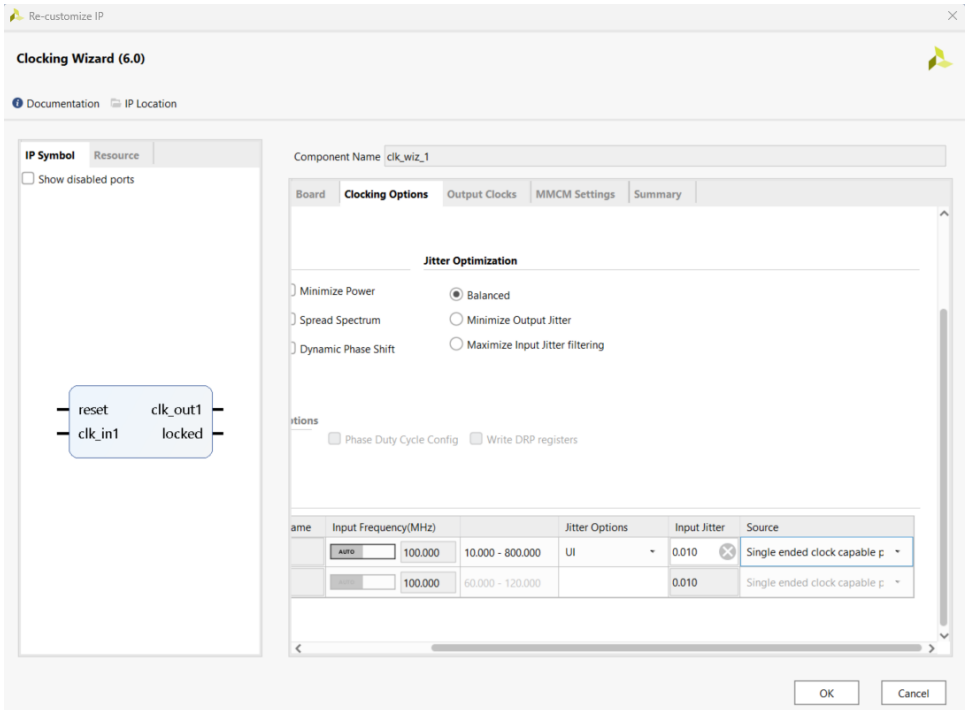


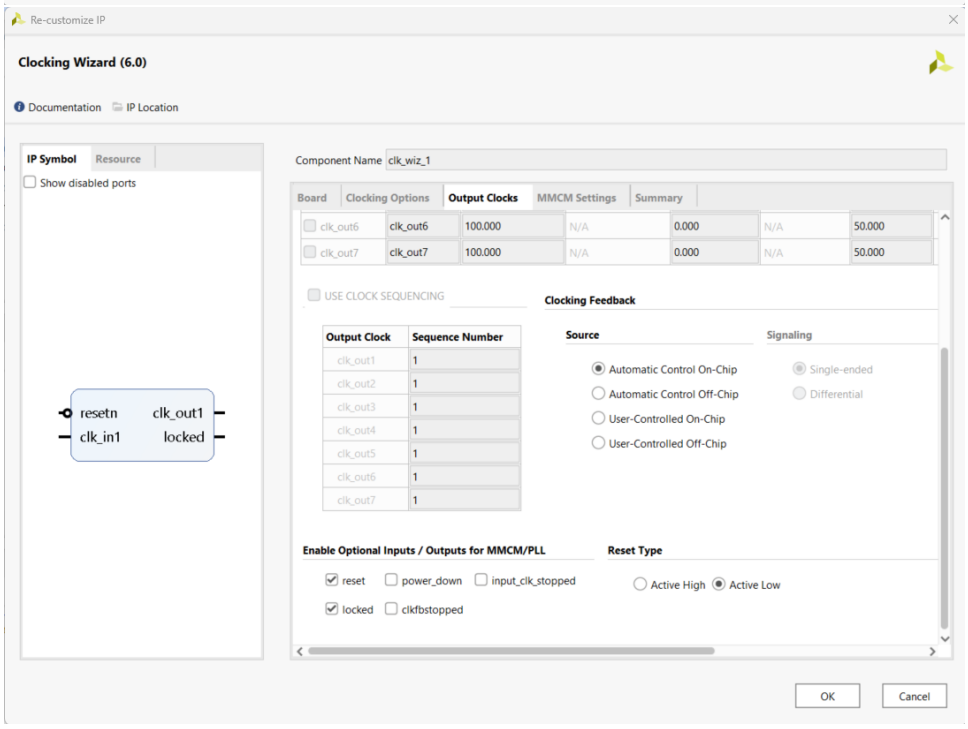
Merging changes and packing to create custom digi_stopwatch IP.



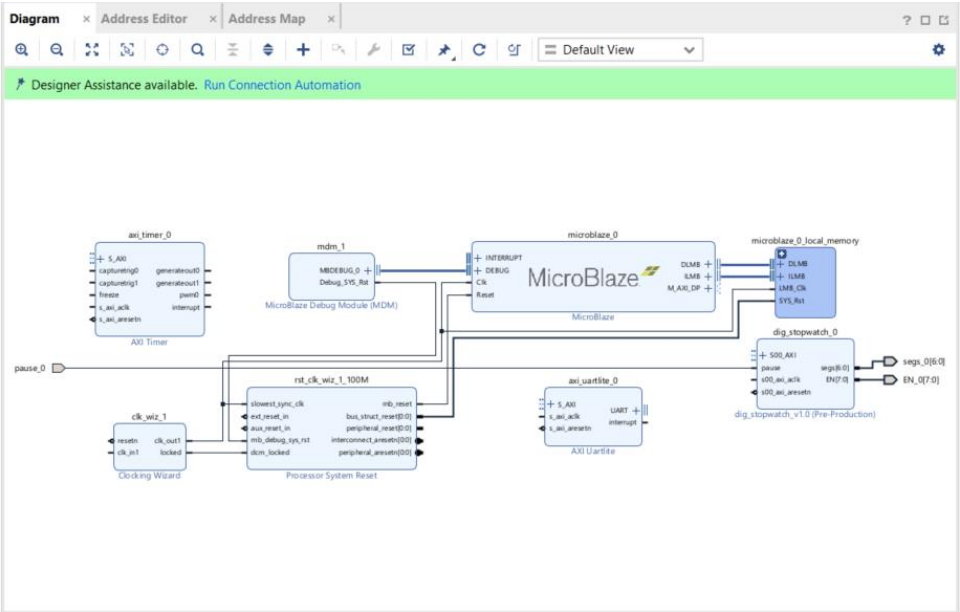


Adding Microblaze IP on block design and running block automation to automatically add peripherals according to the options.



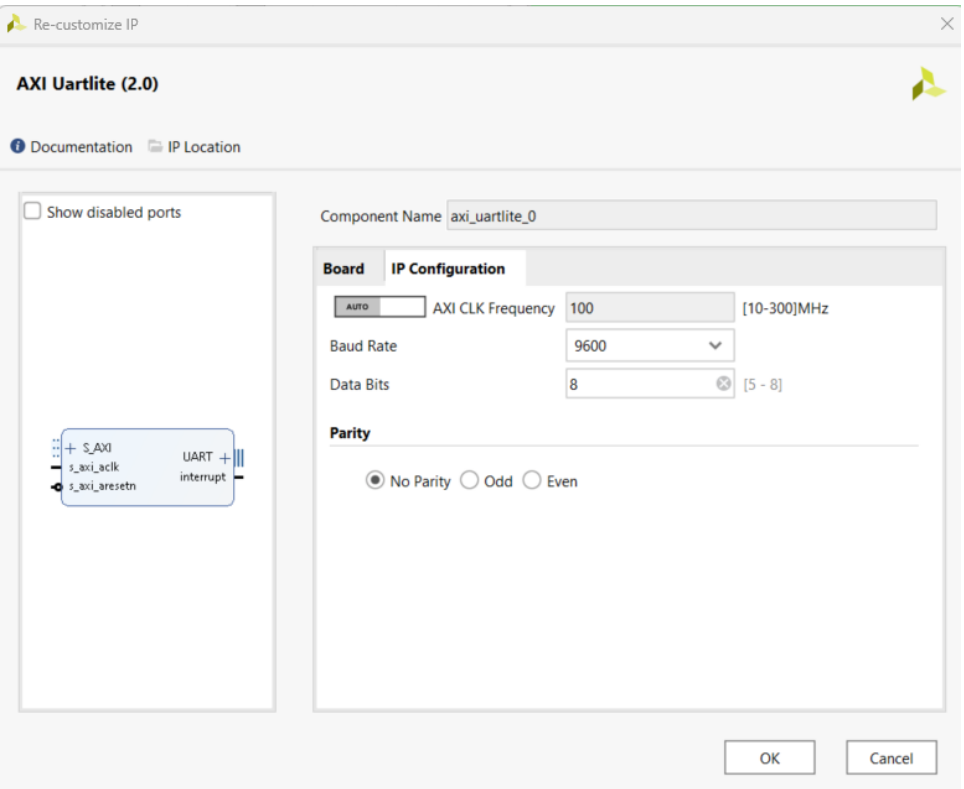


Changing to single ended clock from differential and reset type to active low.

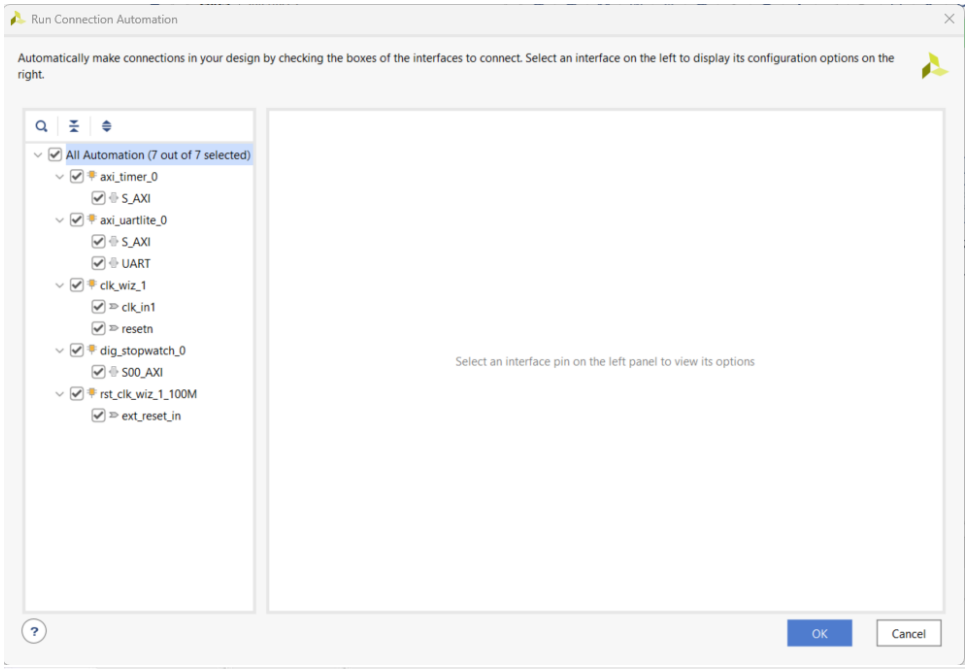


Added three IP components

- AXI UART (Serial Communication)
- AXI Timer (Accurate Timer)
- Custom Digital Stopwatch



Setting baud rate in AXI UART value as 9600 for later use to match it with local machine. Setting segs, EN, pause as external.


Run Connection Automation
Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.
All Automation (7 out of 7 selected)
axi_timer_0
S_AXI
axi_uartlite_0
S_AXI
UART
clk_wiz_1
clk_in1
resetn
dig_stopwatch_0
S00_AXI
rst_clk_wiz_1_100M
ext_reset_in
Select an interface pin on the left panel to view its options
OK
Cancel

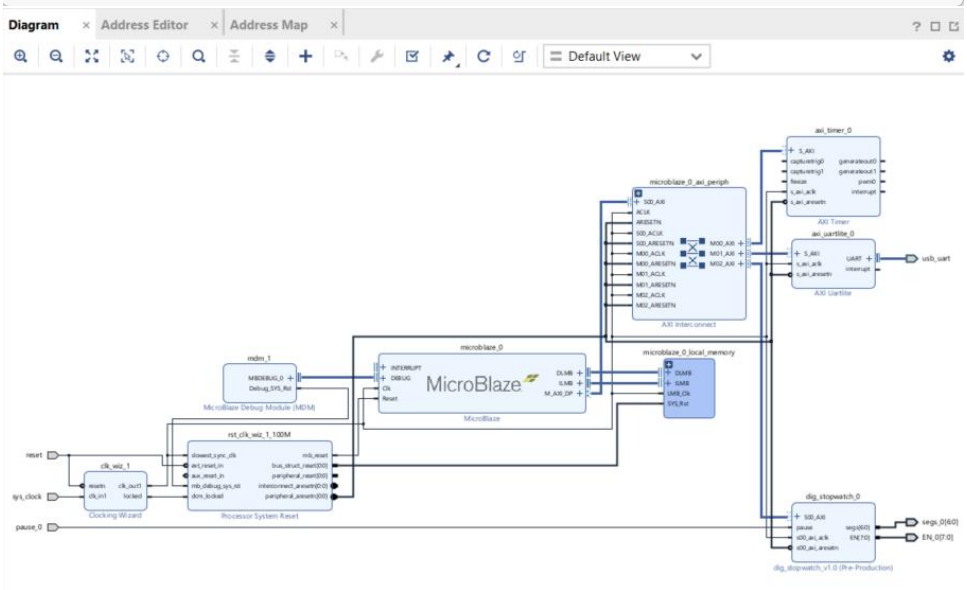
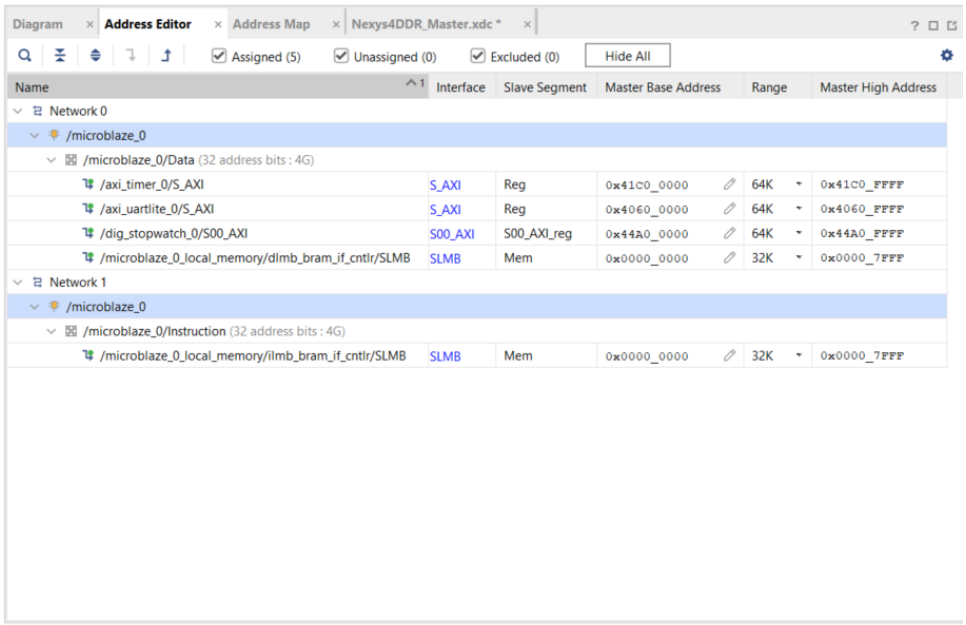
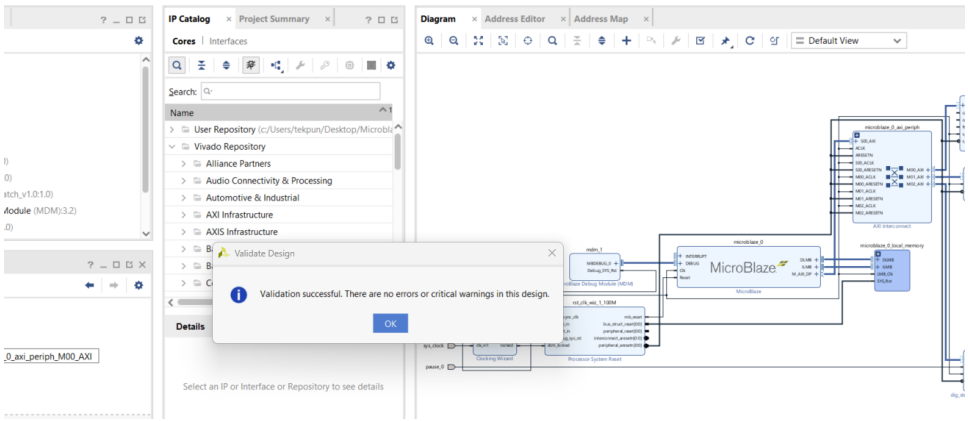
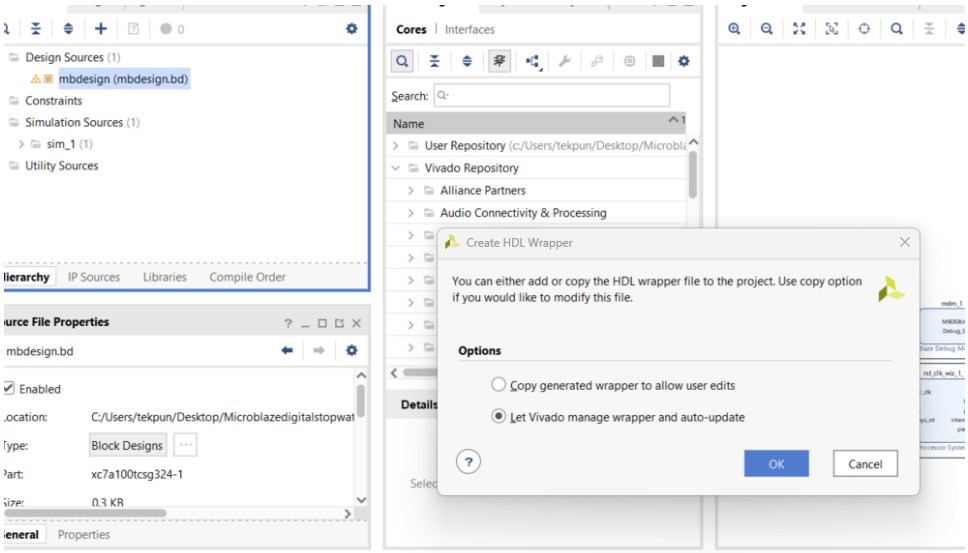


Diagram
Address Editor
Address Map
Default View
axi_timer_0
microblaze_0_axi_periph
axi_uartlite_0
usb_uart
mdm_1
microblaze_0
MicroBlaze
microblaze_0_local_memory
rst_clk_wiz_1_100M
clk_wiz_1
reset
sys_clock
pause_0
dig_stopwatch_0
segs_0[6:0]
EN_0[7:0]

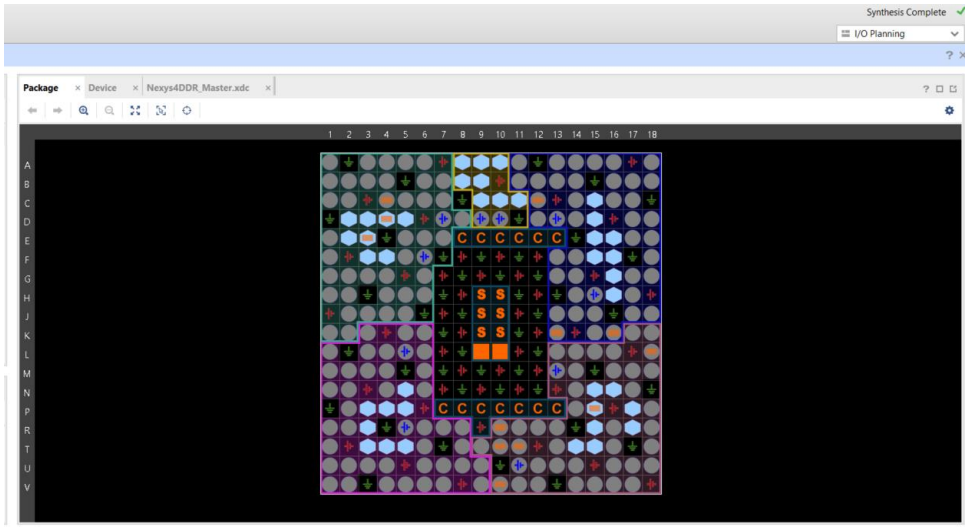




Running connection automation for automatic connection between all IP and validating design afterwards.

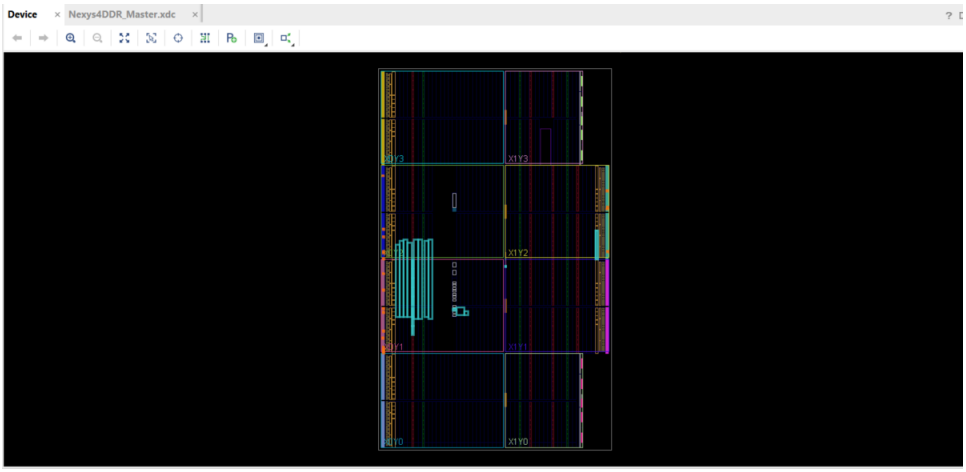


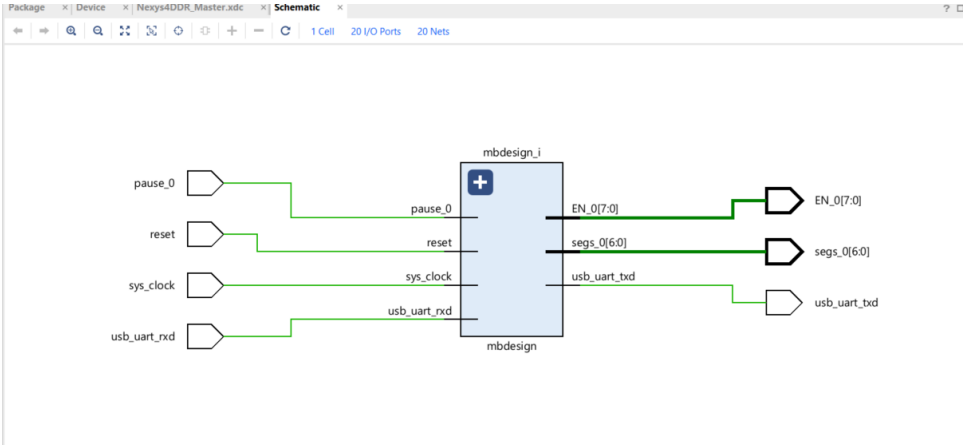
Creating HDL Wrapper to translate the block design into a source file.



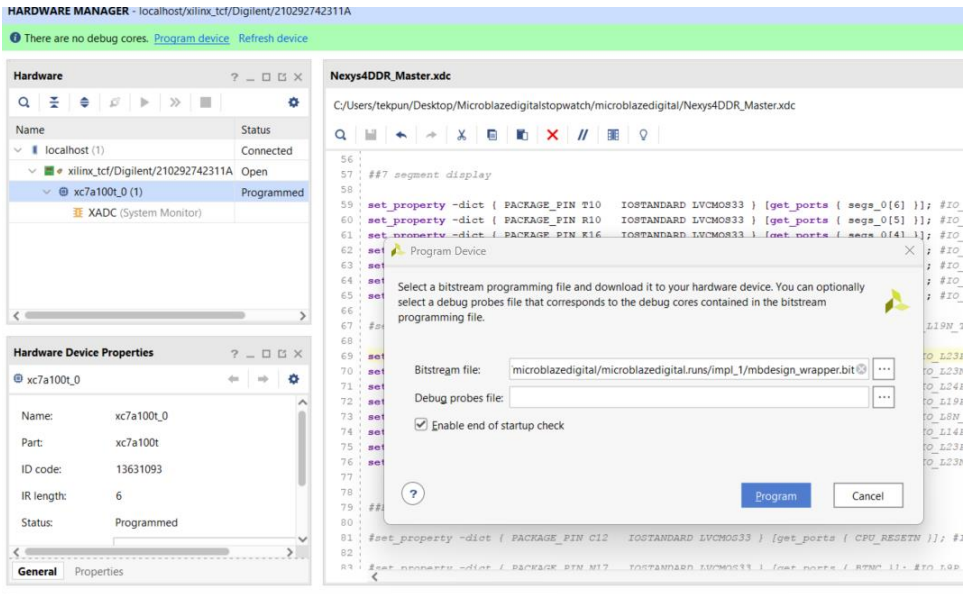
All ports (20)												
CLK.SYS_CLOCK_34497 (1)	IN						✓	35	LVCMOS33*		3.300	
Scalar ports (1)												
sys_clock	IN	clk				E3	✓	35	LVCMOS33*		3.300	
RST.RESET_34497 (1)	IN						✓	15	LVCMOS33*		3.300	
usb_uart_34497 (2)	(Multiple)						✓	35	LVCMOS33*		3.300	
EN_0 (8)	OUT						✓	(Multiple)	LVCMOS33*		3.300	12
EN_0[7]	OUT					U13	✓	14	LVCMOS33*		3.300	12
EN_0[6]	OUT					K2	✓	35	LVCMOS33*		3.300	12
EN_0[5]	OUT					T14	✓	14	LVCMOS33*		3.300	12
EN_0[4]	OUT					P14	✓	14	LVCMOS33*		3.300	12
EN_0[3]	OUT					J14	✓	15	LVCMOS33*		3.300	12
EN_0[2]	OUT					T9	✓	14	LVCMOS33*		3.300	12
EN_0[1]	OUT					J18	✓	15	LVCMOS33*		3.300	12
EN_0[0]	OUT					J17	✓	15	LVCMOS33*		3.300	12
segs_0 (7)	OUT						✓	(Multiple)	LVCMOS33*		3.300	12
segs_0[6]	OUT					T10	✓	14	LVCMOS33*		3.300	12
segs_0[5]	OUT					R10	✓	14	LVCMOS33*		3.300	12
segs_0[4]	OUT					K16	✓	15	LVCMOS33*		3.300	12
segs_0[3]	OUT					K13	✓	15	LVCMOS33*		3.300	12
segs_0[2]	OUT					P15	✓	14	LVCMOS33*		3.300	12
segs_0[1]	OUT					T11	✓	14	LVCMOS33*		3.300	12
segs_0[0]	OUT					L18	✓	14	LVCMOS33*		3.300	12
Scalar ports (1)												
pause_0	IN					V10	✓	14	LVCMOS33*		3.300	

```
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

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


##Switches

#set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { sel[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10
#set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { sel[11] }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
#set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { resetn }]; #IO_L24P_T3_35 Sch=sw[12]
#set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { SW[13] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
#set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { SW[14] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[1
set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { pause_0 }]; #IO_L21P_T3_DQS_14 Sch=sw[15]

##7 segment display

set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { segs_0[6] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { segs_0[5] }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { segs_0[4] }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { segs_0[3] }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { segs_0[2] }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { segs_0[1] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { segs_0[0] }]; #IO_L4P_T0_D04_14 Sch=cg

#set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { DP }]; #IO_L19N_T3_A21_VREF_15 Sch=dp

set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { EN[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { EN[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { EN[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { EN[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { EN[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { EN[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { EN[6] }]; #IO_L23P_T3_35 Sch=an[6]
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { EN[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]
```

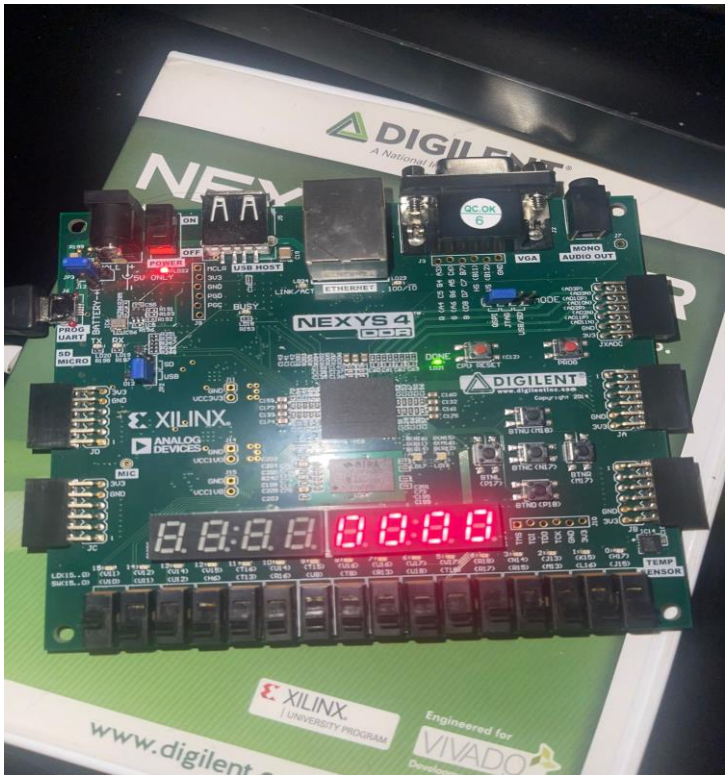




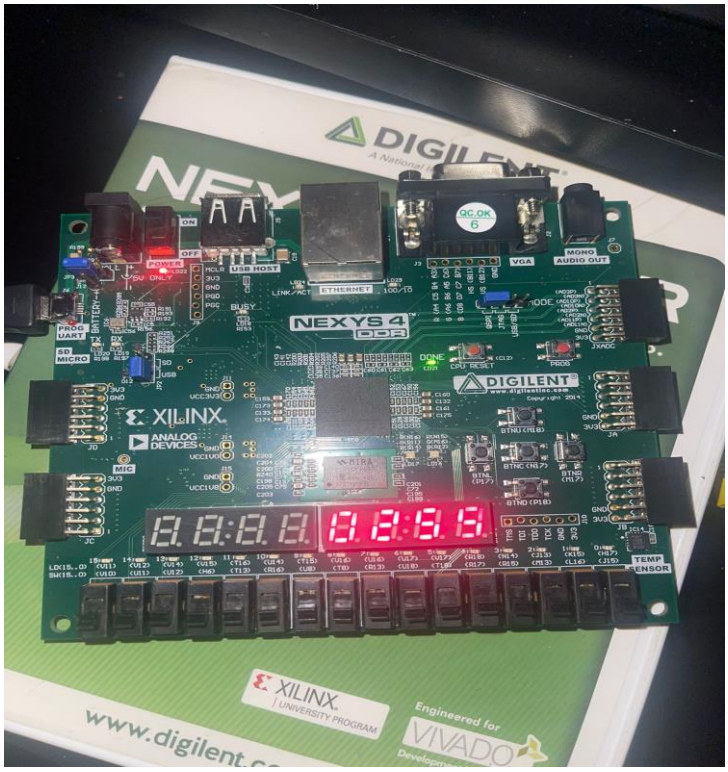
Runing synthesis and implementation with I/O floor plan constraints for 7 segment display, switch and clock.



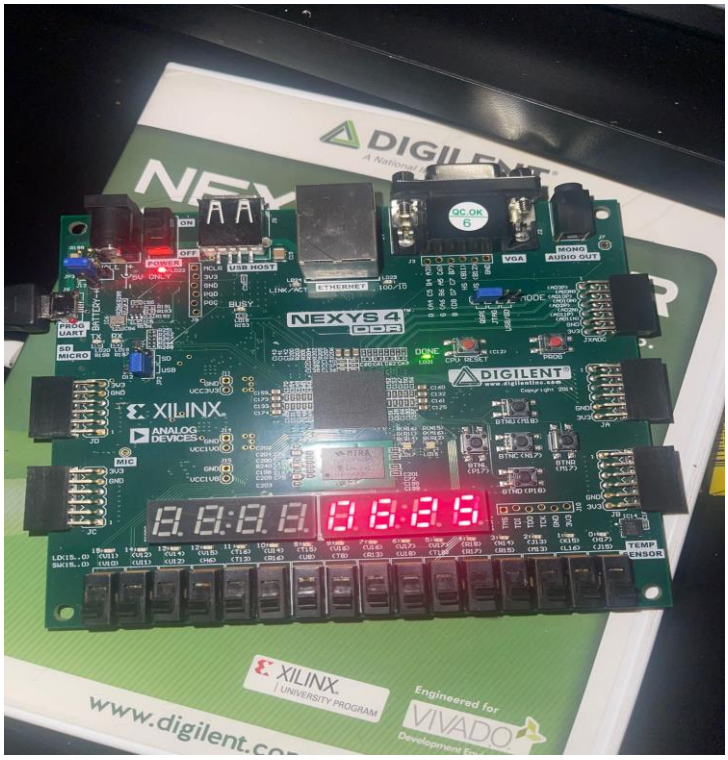
Generating bitstream and programming to Nexys board.



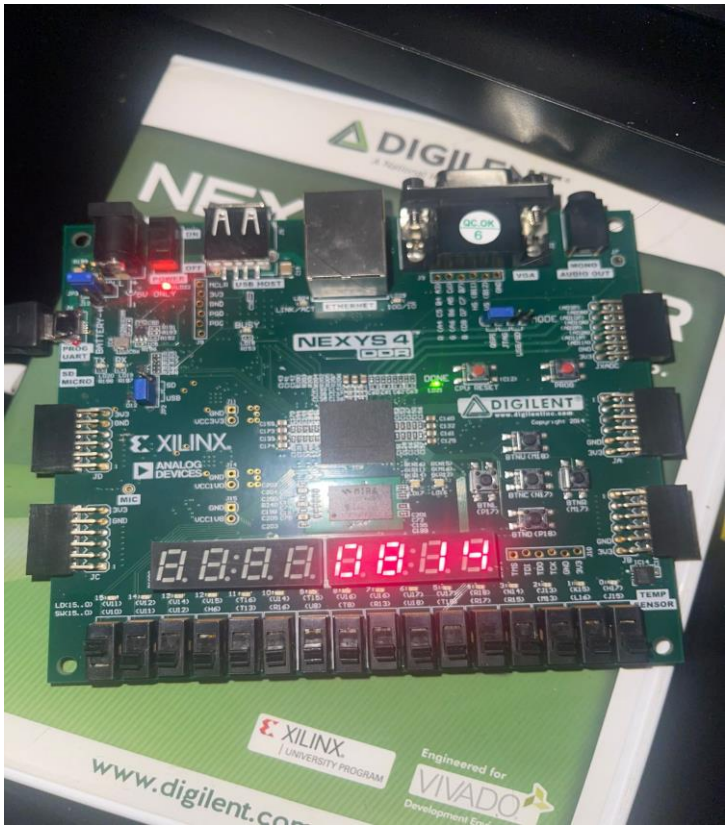
Initial state with pause. Switch V10(Most left switch) is the input pause.



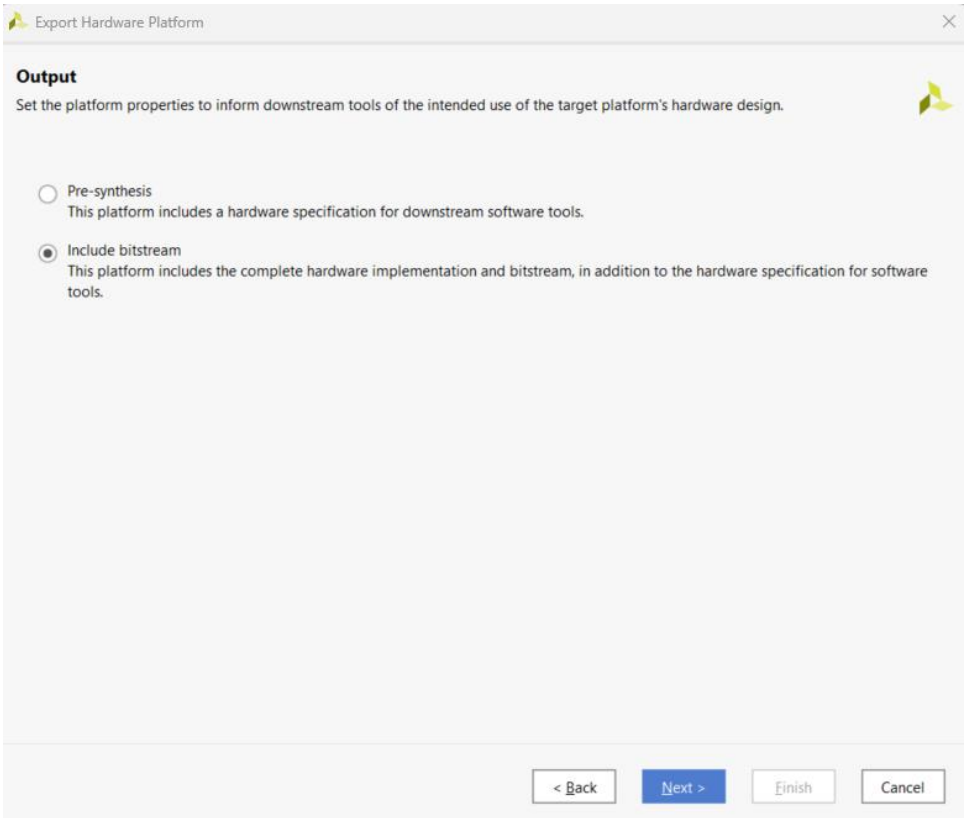
When the switch is triggered to high, the timer starts



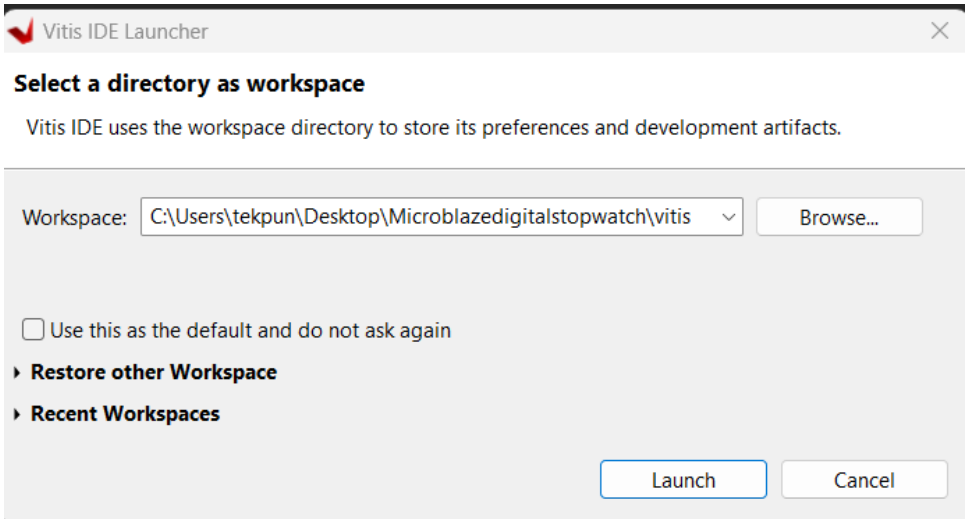
Switch set as low causing the timer to stop



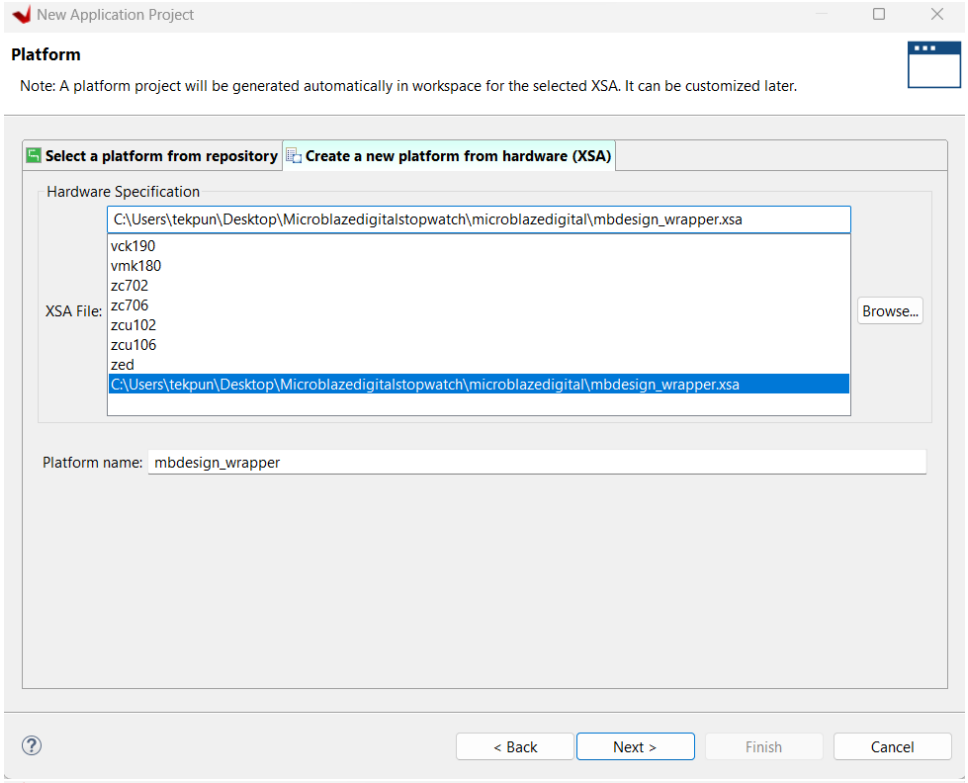
Switch set as high, timer running.

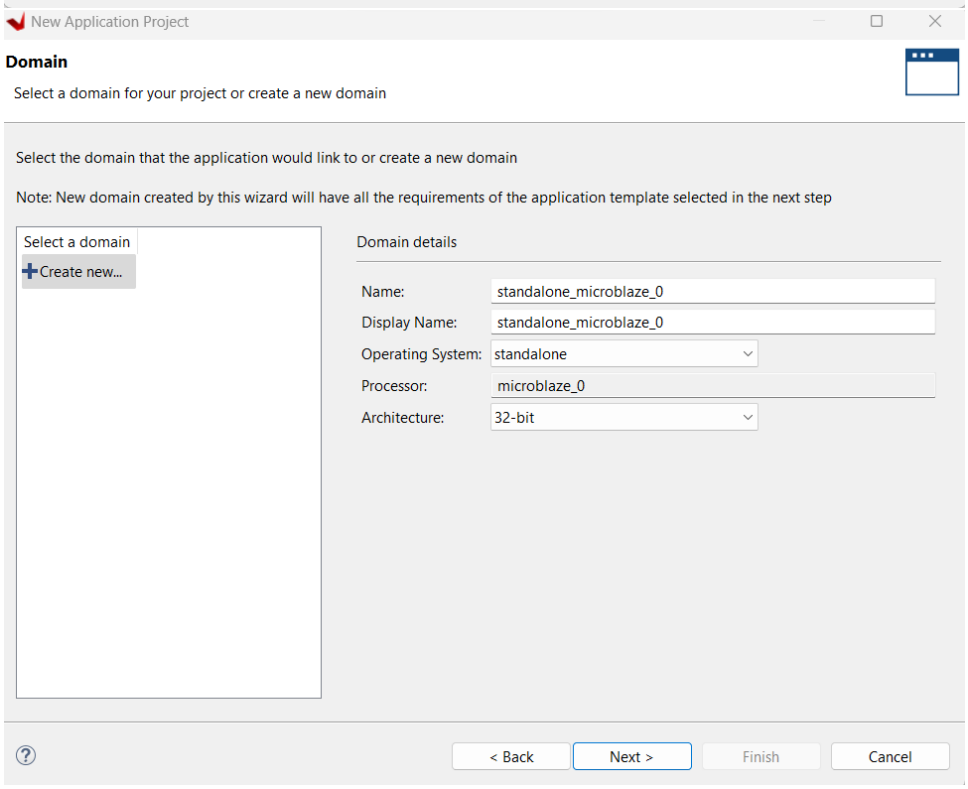


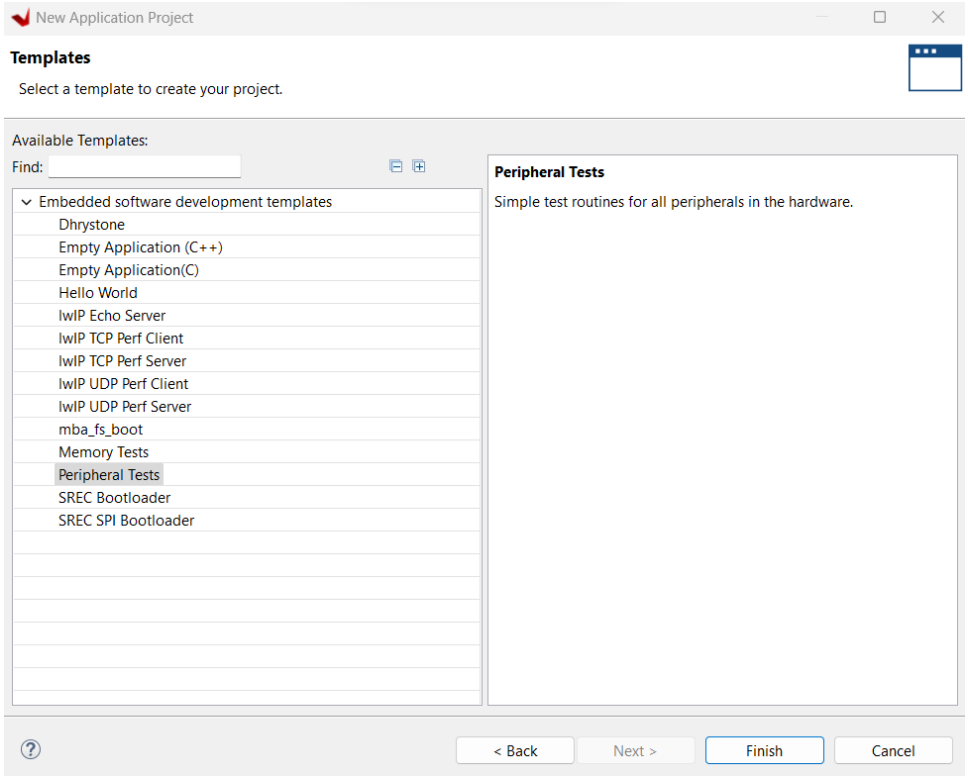
Exporting hardware.



Launching vitis.


New Application Project
Platform
Note: A platform project will be generated automatically in workspace for the selected XSA. It can be customized later.
Select a platform from repository
Create a new platform from hardware (XSA)
Hardware Specification
XSA File:
C:\Users\tekpun\Desktop\Microblazedigitalstopwatch\microblazedigital\mbdesign_wrapper.xsa
vck190
vmk180
zc702
zc706
zcu102
zcu106
zed
C:\Users\tekpun\Desktop\Microblazedigitalstopwatch\microblazedigital\mbdesign_wrapper.xsa
Browse...
Platform name:
mbdesign_wrapper
< Back
Next >
Finish
Cancel


New Application Project
Domain
Select a domain for your project or create a new domain
Select the domain that the application would link to or create a new domain
Note: New domain created by this wizard will have all the requirements of the application template selected in the next step
Select a domain
Create new...
Domain details
Name:
standalone_microblaze_0
Display Name:
standalone_microblaze_0
Operating System:
standalone
Processor:
microblaze_0
Architecture:
32-bit
< Back
Next >
Finish
Cancel



Setting platform with the XSA file imported from vivado and creating a peripheral test application.

```
#include <stdio.h>
#include "xparameters.h"
#include "xil_cache.h"
#include "xtmrctr.h"
#include "tmrctr_header.h"
int main ()
{
   Xil_ICacheEnable();
   Xil_DCacheEnable();
   print("---Entering main---\n\r");


   {
      int status;

      print("\r\n Running TmrCtrSelfTestExample() for axi_timer_0...\r\n");

      status = TmrCtrSelfTestExample(XPAR_AXI_TIMER_0_DEVICE_ID, 0x0);

      if (status == 0) {
         print("TmrCtrSelfTestExample PASSED\r\n");
      }
      else {
         print("TmrCtrSelfTestExample FAILED\r\n");
      }
   }
```

```
Console  Problems  Vitis Log  Guidance
Build Console [mbpheritest, Debug]
04:45:57 **** Incremental Build of configuration Debug for project mbpheritest ****
make all
'Building file: ../src/testperiph.c'
'Invoking: MicroBlaze gcc compiler'
mb-gcc -Wall -O0 -g3 -c -fmessage-length=0 -MT"src/testperiph.o" -IC:/Users/tekpun/Desktop/Microblazedigitalstopwatch/
../src/testperiph.c:31:10: fatal error: xparameters.h: No such file or directory
   31 | #include "xparameters.h"
      |          ^~~~~~~~~~~~~~~
compilation terminated.
make: *** [src/subdir.mk:26: src/testperiph.o] Error 1

04:45:58 Build Finished (took 247ms)
```

First build after creating a peripheral test resulted in error stating "#inlcude xparameters.h no such file or directory"

https://support.xilinx.com/s/question/0D52E00006jpcvVSAQ/fatal-error-xparametersh-no-such-file-or-directoryxilinx-platform-definition-filexpfm-is-removed-after-building-the-project?language=en_US Researching about this error helped to solve the build issue as there was problem with the makefile for the custom dig_stopwatch IP that we created. The directory was not the same as the Board support package for the custom dig_stopwatch IP as the rest of the IP's that was implemented.

```
pheri_test_system   pheri_test   Makefile
COMPILER=
ARCHIVER=
CP=cp
COMPILER_FLAGS=
EXTRA_COMPILER_FLAGS=
LIB=libxil.a

RELEASEDIR=../../../lib
INCLUDEDIR=../../../include
INCLUDES=-I./. -I${INCLUDEDIR}

INCLUDEFILES=*.h
LIBSOURCES=*.c
OUTS = *.o

libs:
    echo "Compiling dig_stopwatch..."
    $(COMPILER) $(COMPILER_FLAGS) $(EXTRA_COMPILER_FLAGS) $(INCLUDES) $(LIBSOURCES)
    $(ARCHIVER) -r ${RELEASEDIR}/${LIB} ${OUTS}
    make clean

include:
    ${CP} $(INCLUDEFILES) $(INCLUDEDIR)

clean:
    rm -rf ${OUTS}
```

Makefile for C:\Users\tekpun\Desktop\Microblazedigitalstopwatch\vitisfinal\mbdesign_wrapper\hw\drivers\dig_stopwatch_v1_0\src

And also

C:\Users\tekpun\Desktop\Microblazedigitalstopwatch\vitisfinal\mbdesign_wrap

per\microblaze_0\standalone_microblaze_0\bsp\microblaze_0\libsrc\dig_stopwatch_v1_0\src .

Both of the makefile for the custom ip on the BSP settings were similar whereas makefile for the other ip generated automatically from the ip catalogue was different.

pheri_test_system | pheri_test | Makefile | Makefile | Makefile

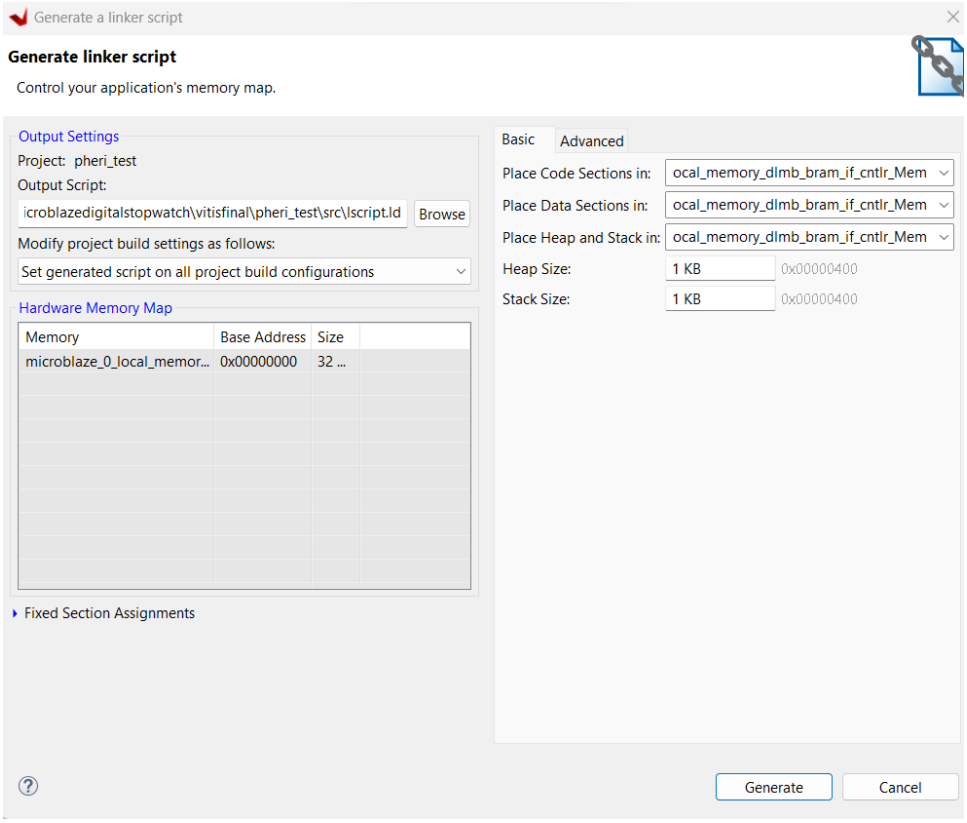
```
DRIVER_LIB_VERSION = 1.0
COMPILER=
ARCHIVER=
CP=cp
COMPILER_FLAGS=
EXTRA_COMPILER_FLAGS=
LIB=libxil.a

CC_FLAGS = $(COMPILER_FLAGS)
ECC_FLAGS = $(EXTRA_COMPILER_FLAGS)

RELEASEDIR=../../../lib/
INCLUDEDIR=../../../include/
INCLUDES=-I./. -I$(INCLUDEDIR)

SRCFILES:=$(wildcard *.c)

OBJECTS = $(addprefix $(RELEASEDIR), $(addsuffix .o, $(basename $(wildcard *.c))))

libs: $(OBJECTS)

DEPFILES := $(SRCFILES:%.c=$(RELEASEDIR)%.d)

include $(wildcard $(DEPFILES))

include $(wildcard ../../../../dep.mk)

$(RELEASEDIR)%.o: %.c
	${COMPILER} $(CC_FLAGS) $(ECC_FLAGS) $(INCLUDES) $(DEPENDENCY_FLAGS) $< -o $@

.PHONY: include
include: $(addprefix $(INCLUDEDIR),$(wildcard *.h))

$(INCLUDEDIR)%.h: %.h
	$(CP) $< $@

clean:
	rm -rf ${OBJECTS}
	rm -rf $(DEPFILES)
```

Makefile for uartlite, bramv4_8, cpu_v2_16,standalone_v8_0 and tmrctr_v4_9 on the BSP file. After changing makefile for the custom digital stopwach ip to the same as the IP from the Catalogue, the build was successful.

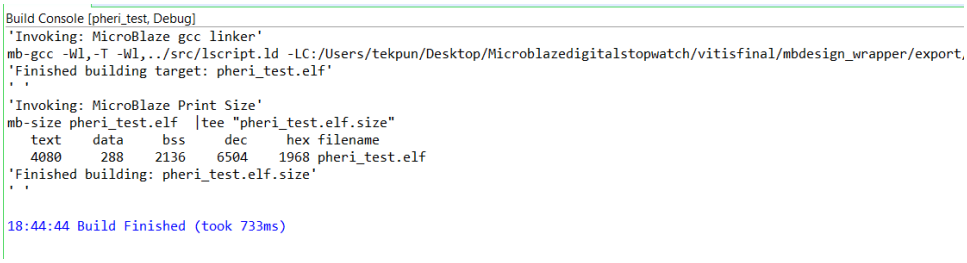
Build Console [pheri_test_system, Debug]

```
18:41:09 **** Build of configuration Debug for project pheri_test_system ****
make all
Skipping SD card image generation. Reason: "The system project only has applications for microblaze

18:41:09 Build Finished (took 114ms)
```

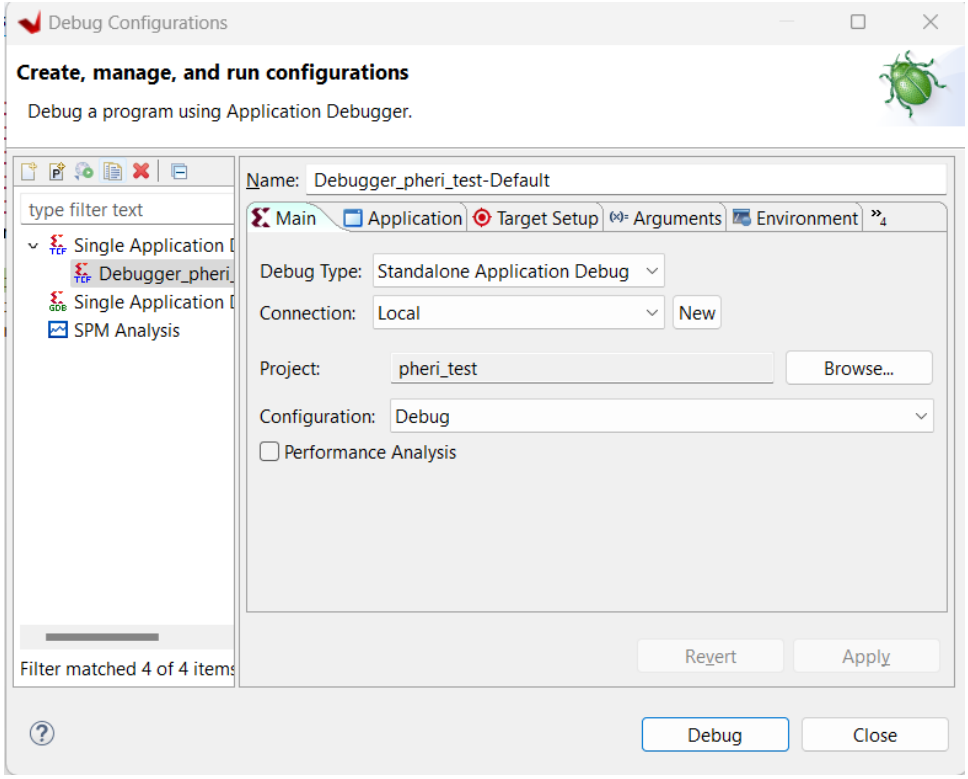
Build successful after changing makefile.



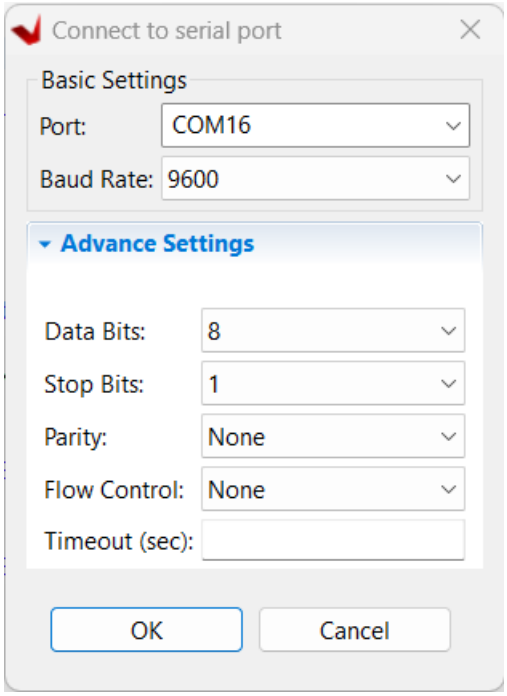
Generating linker script for pheri_test application



Rebuilding after generating linker script



Creating a single application debug for pher_test application



Connecting to serial port via AXI UART baud rate which we set earlier to 9600 and the port to which the board is connected.

```c
#include "tmrctr_header.h"
int main ()
{
   Xil_ICacheEnable();
   Xil_DCacheEnable();
   print("---Entering main---\n\r");


   {
      int status;

      print("\r\n Running TmrCtrSelfTestExample() for axi_timer_0...\r\n");

      status = TmrCtrSelfTestExample(XPAR_AXI_TIMER_0_DEVICE_ID, 0x0);

      if (status == 0) {
         print("TmrCtrSelfTestExample PASSED\r\n");
      }
      else {
         print("TmrCtrSelfTestExample FAILED\r\n");
      }
   }
```

Creating a breakpoint to check for output on serial monitor.

```
Console | Vitis Serial Terminal | Executables | Debug Shell | Vitis Log | Problems | Debugger Console
Connected to: Serial ( COM16, 9600, 0, 8 )

Connected to COM16 at 9600
---Entering main---


 Running TmrCtrSelfTestExample() for axi_timer_0...
TmrCtrSelfTestExample PASSED
---Exiting main---
```

Serial monitor output: axi_timer_0 test passed.

```
/* Definitions for driver DIG_STOPWATCH */
#define XPAR_DIG_STOPWATCH_NUM_INSTANCES 1

/* Definitions for peripheral DIG_STOPWATCH_0 */
#define XPAR_DIG_STOPWATCH_0_DEVICE_ID 0
#define XPAR_DIG_STOPWATCH_0_S00_AXI_BASEADDR 0x44A00000
#define XPAR_DIG_STOPWATCH_0_S00_AXI_HIGHADDR 0x44A0FFFF


/******************************************************************/
```

```
/* Definitions for driver UARTLITE */
#define XPAR_XUARTLITE_NUM_INSTANCES 1U

/* Definitions for peripheral AXI_UARTLITE_0 */
#define XPAR_AXI_UARTLITE_0_DEVICE_ID 0U
#define XPAR_AXI_UARTLITE_0_BASEADDR 0x40600000U
#define XPAR_AXI_UARTLITE_0_HIGHADDR 0x4060FFFFU
#define XPAR_AXI_UARTLITE_0_BAUDRATE 9600U
#define XPAR_AXI_UARTLITE_0_USE_PARITY 0U
#define XPAR_AXI_UARTLITE_0_ODD_PARITY 0U
#define XPAR_AXI_UARTLITE_0_DATA_BITS 8U
```

Definitions for peripherals from the xparameters.h file.

```
#include <stdio.h>
#include "xparameters.h"
#include "xil_cache.h"
#include "xtmrctr.h"
#include "tmrctr_header.h"
```

Including files that are being used.

```
{
   int status;

   print("\r\n Running TmrCtrSelfTestExample() for axi_timer_0...\r\n");

   status = TmrCtrSelfTestExample(XPAR_AXI_TIMER_0_DEVICE_ID, 0x0);

   if (status == 0) {
      print("PASSED for axi_timer\r\n");
   }
   else {
      print("FAILED for axi_timer\r\n");
   }
}
```

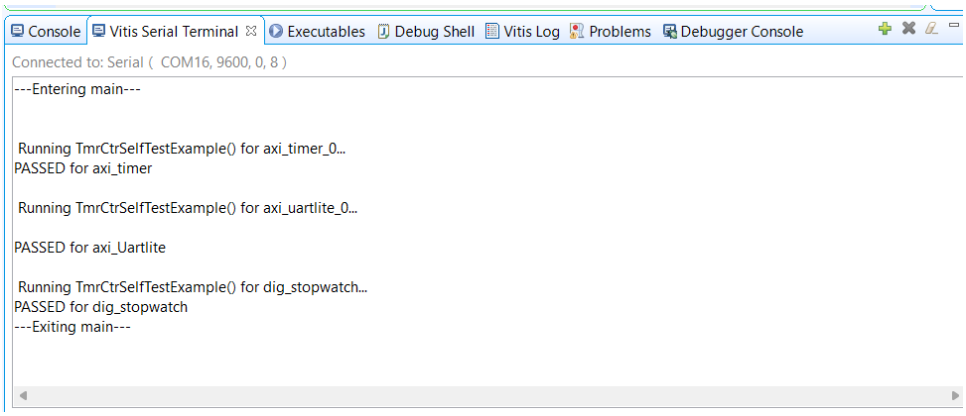
Status to check for axi_timer_0 in testperiph.c

```
{
    int status1;

    print("\r\n Running TmrCtrSelfTestExample() for axi_uartlite_0...\r\n");

    status1 = TmrCtrSelfTestExample(XPAR_AXI_UARTLITE_0_DEVICE_ID, 0x0);

    if (status1 == 0) {
        print("PASSED for axi_Uartlite\r\n");
    }
    else {
        print("FAILED for axi_Uartlite\r\n");
    }
}
```

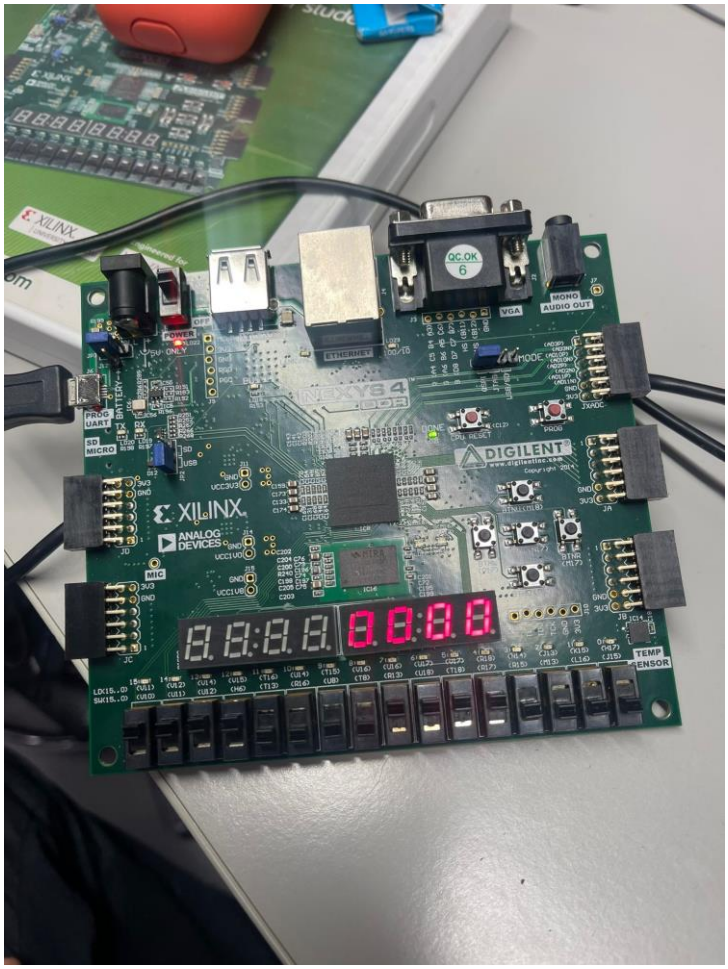
Status1 to check for AXI_UARTLITE in testperiph.c

```
{
    int status2;

    print("\r\n Running TmrCtrSelfTestExample() for dig_stopwatch...\r\n");

    status2 = TmrCtrSelfTestExample(XPAR_DIG_STOPWATCH_0_DEVICE_ID, 0x0);

    if (status2 == 0) {
        print("PASSED for dig_stopwatch\r\n");
    }
    else {
        print("FAILED for dig_stopwatch\r\n");
    }
}
```

Status2 to check for dig_stopwatch in testperiph.c

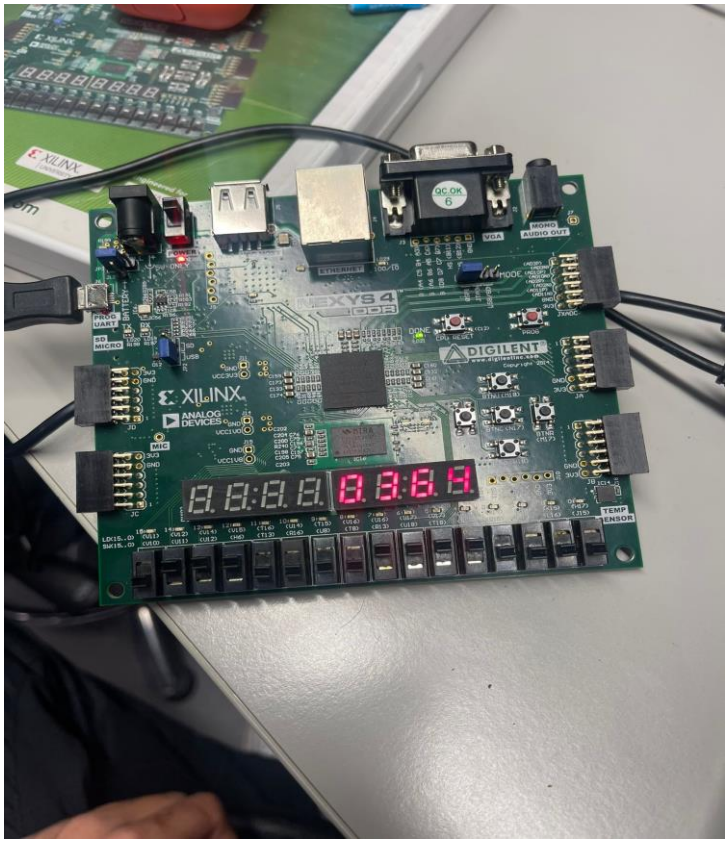


Output on Vitis serial terminal showing

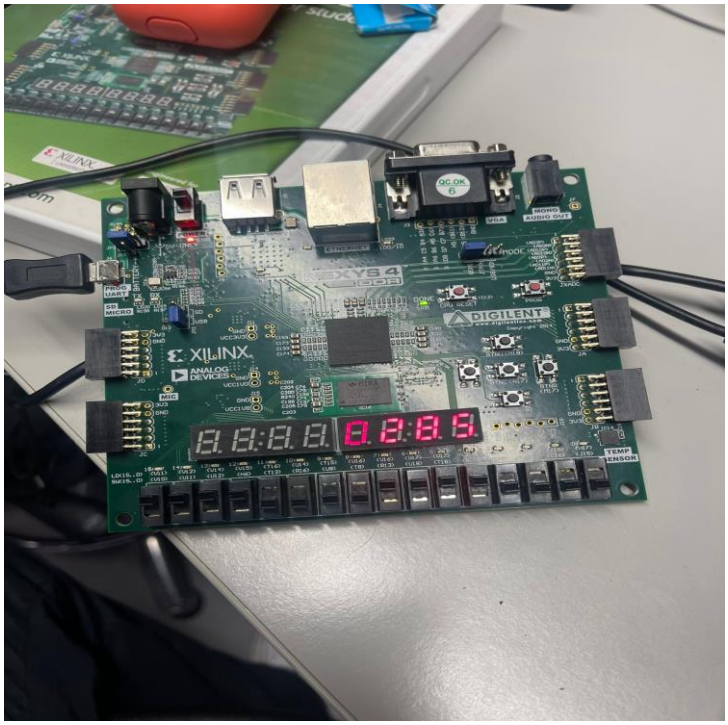


Nexys board being programmed from vitis.

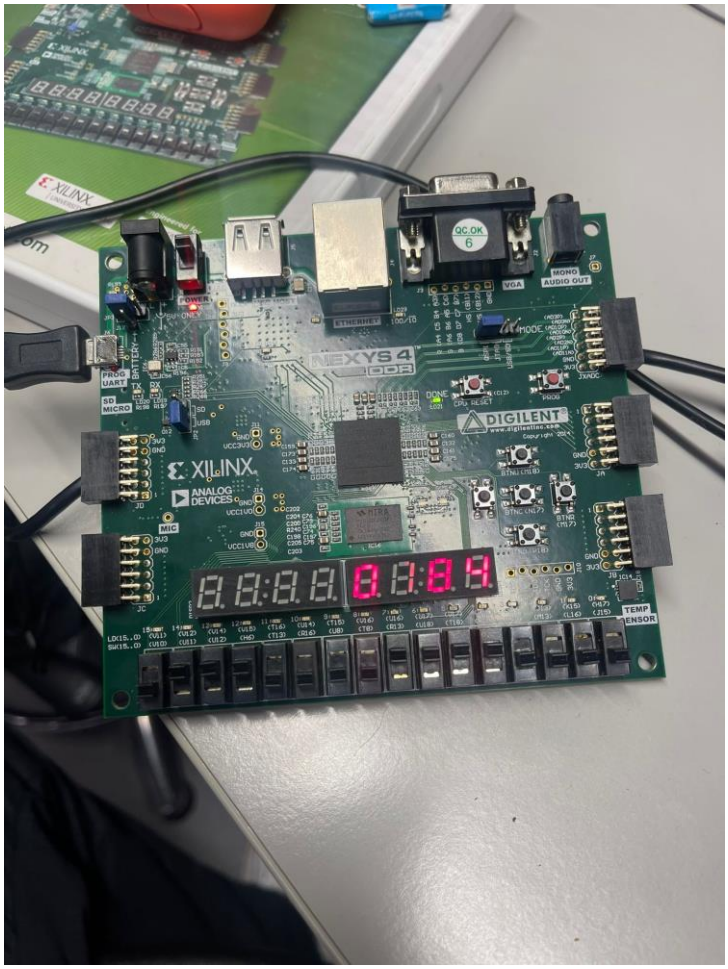
Pause(switch V10) set to high



Pause(switch V10) set to low



Pause(switch V10) set to high



Pause(switch V10) set to low

4. Conclusion

In conclusion, the development and integration of a custom AXI digital stopwatch with a MicroBlaze processor was successful. A key aspect to take away from this project is being able to ensure that the IP repository is correctly configured, and its location is specified in your project settings to make sure they are accurate. Also verify that the IP repository has been correctly initialized. For certain tools to identify the IP repository, there needs to be an initial setup or configuration procedure. One such problem faced was when building the pheriperal.c file and having the “parameters.h” error whereby to solve it, the makefile of the custom digital stopwatch had to be changed which allowed the ip path to be correct leading to a successful build.