Producing an Animation Using AXI Timer Interrupt

Goal

- Integrate AXI Timer into the design from VGA tutorial
- Enable and use interrupt from AXI Timer to control animation

Requirements

- Xilinx Vivado software
- Xilinx Nexys 4 board and a programming cable
- Enough disk space for the project files
- Existing VGA tutorial project

Background

Recall from the VGA tutorial that we used C loops to delay the drawing process in order to produce an animation. The software will loop for multiple cycles doing no effective work to achieve a frame rate close to 30 FPS. This is not effective as we are consuming unnecessary processing power. Note that this is similar to the polling method used in devices. An alternative method to achieve delays in animation is to use a device counter and produce an **interrupt** signal to the processor and only update the drawings when interrupted. During the counting process, the processor can carryout other meaningful computations instead of polling.

Introduction

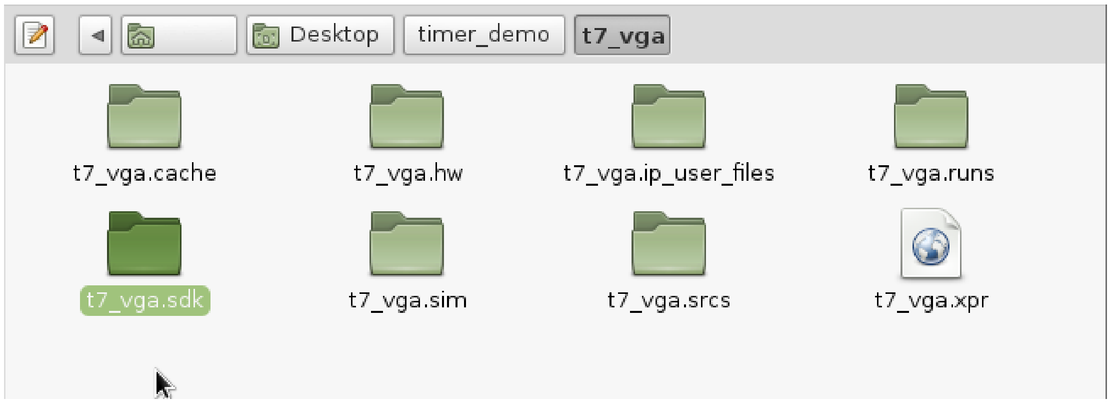
In this tutorial you will reproduce the animation done in the VGA tutorial using a device timer and interrupts. Instead of creating a new project, we will work on top of the old VGA project to save time.

1. **Update the Hardware**

In this section, we will update the hardware aspect of the VGA tutorial to include an AXI Timer

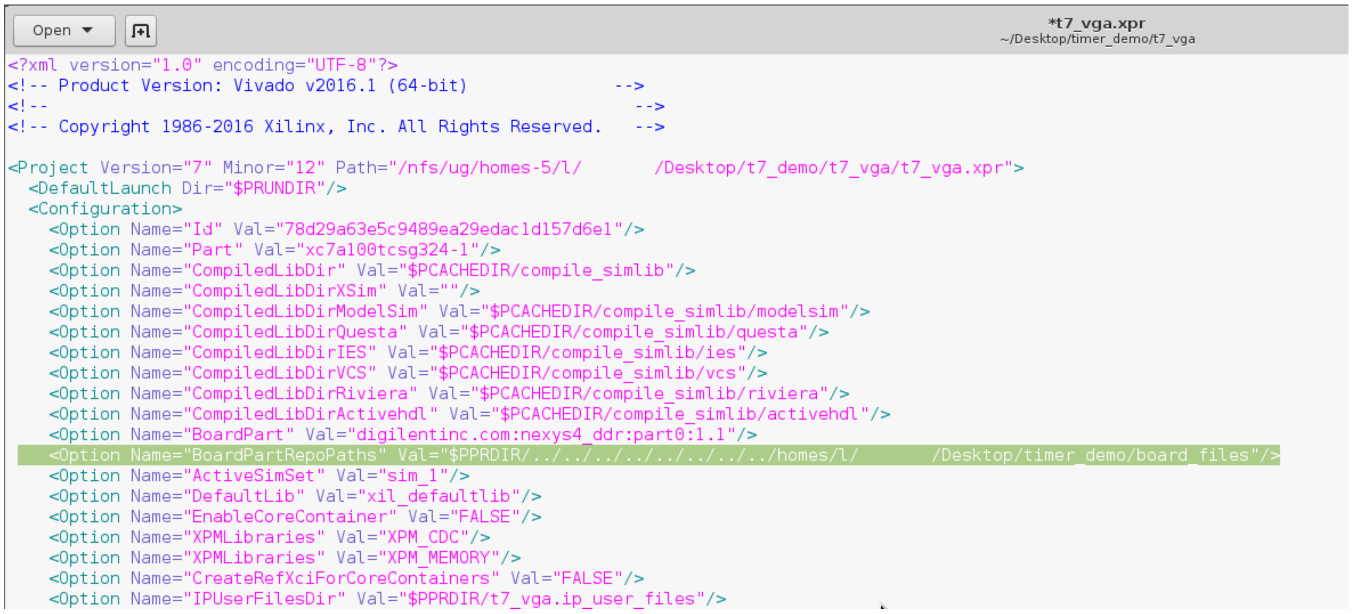
1.1 Create a backup of the VGA lab

1.2 Open the VGA lab project folder and delete the SDK file



1.3 In the project folder find the .xpr project file and use a text editor to open it

1.4 Go to line 20, BoardPartRepoPaths, make sure that the directory to your board file Nexys 4 DDR is still effective.



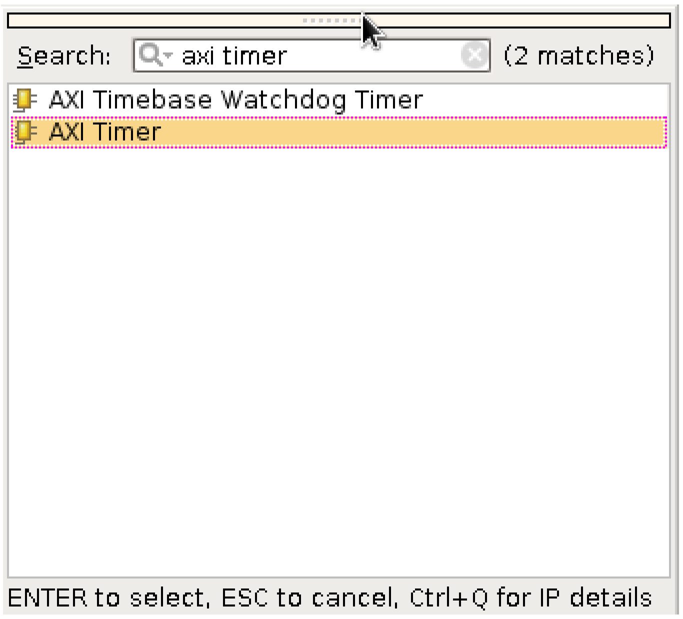
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Product Version: Vivado v2016.1 (64-bit)              -->
<!--                                                         -->
<!-- Copyright 1986-2016 Xilinx, Inc. All Rights Reserved.   -->

<Project Version="7" Minor="12" Path="/nfs/ug/homes-5/l/       /Desktop/t7_demo/t7_vga/t7_vga.xpr">
  <DefaultLaunch Dir="$PRUNDIR"/>
  <Configuration>
    <Option Name="Id" Val="78d29a63e5c9489ea29edac1d157d6e1"/>
    <Option Name="Part" Val="xc7a100tcsg324-1"/>
    <Option Name="CompiledLibDir" Val="$PCACHEDIR/compile_simlib"/>
    <Option Name="CompiledLibDirXSim" Val=""/>
    <Option Name="CompiledLibDirModelSim" Val="$PCACHEDIR/compile_simlib/modelsim"/>
    <Option Name="CompiledLibDirQuesta" Val="$PCACHEDIR/compile_simlib/questa"/>
    <Option Name="CompiledLibDirIES" Val="$PCACHEDIR/compile_simlib/ies"/>
    <Option Name="CompiledLibDirVCS" Val="$PCACHEDIR/compile_simlib/vcs"/>
    <Option Name="CompiledLibDirRiviera" Val="$PCACHEDIR/compile_simlib/riviera"/>
    <Option Name="CompiledLibDirActivehdl" Val="$PCACHEDIR/compile_simlib/activehdl"/>
    <Option Name="BoardPart" Val="digilentinc.com:nexys4_ddr:part0:1.1"/>
    <Option Name="BoardPartRepoPaths" Val="$PPRDIR/../../../../../../../../homes/l/       /Desktop/timer_demo/board_files"/>
    <Option Name="ActiveSimSet" Val="sim_1"/>
    <Option Name="DefaultLib" Val="xil_defaultlib"/>
    <Option Name="EnableCoreContainer" Val="FALSE"/>
    <Option Name="XPMLibraries" Val="XPM_CDC"/>
    <Option Name="XPMLibraries" Val="XPM_MEMORY"/>
    <Option Name="CreateRefXciForCoreContainers" Val="FALSE"/>
    <Option Name="IPUserFilesDir" Val="$PPRDIR/t7_vga.ip_user_files"/>
```

1.5 Open Vivado and open the new VGA project

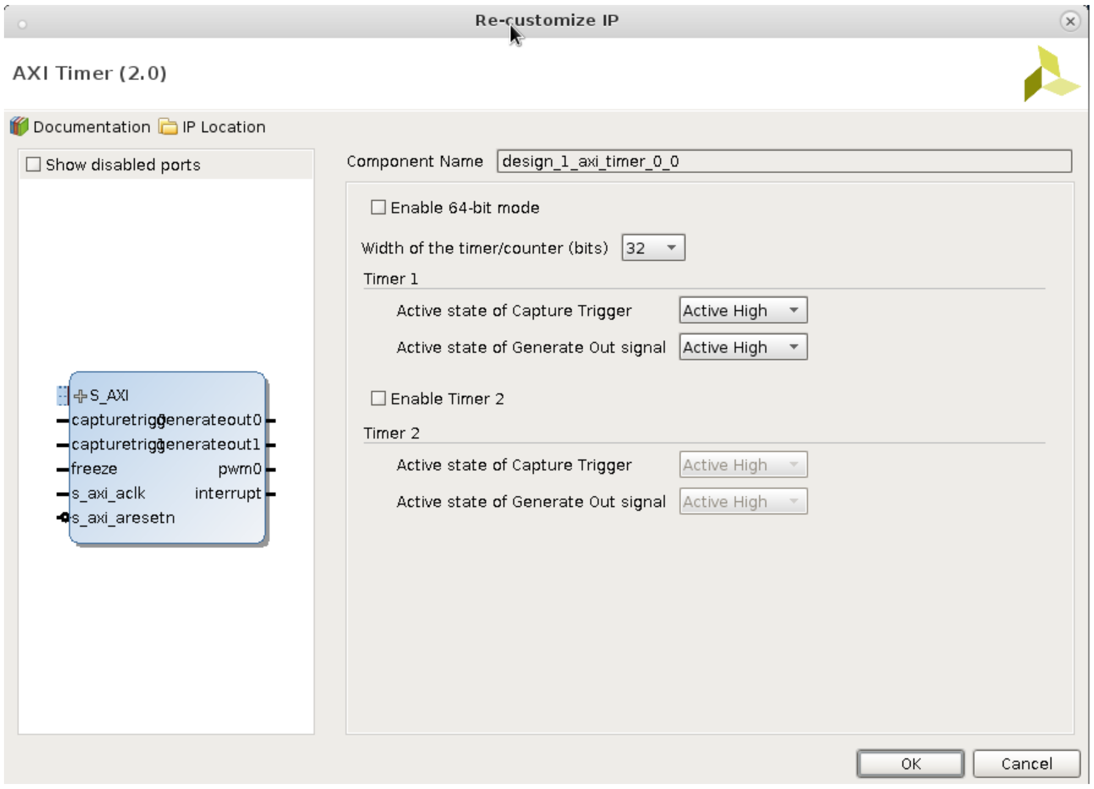
1.6 Open the block design

1.7 Right click anywhere in the block design and select Add IP to add AXI Timer


Search:
axi timer
(2 matches)
AXI Timebase Watchdog Timer
AXI Timer
ENTER to select, ESC to cancel, Ctrl+Q for IP details

1.8 Double click the AXI Timer to reconfigure

1.9 Uncheck Enable Timer 2 as we only will only use one timer and click OK.


Re-customize IP

AXI Timer (2.0)

Documentation | IP Location

☐ Show disabled ports

Component Name: design_1_axi_timer_0_0

☐ Enable 64-bit mode

Width of the timer/counter (bits): 32

Timer 1

Active state of Capture Trigger: Active High

Active state of Generate Out signal: Active High

☐ Enable Timer 2

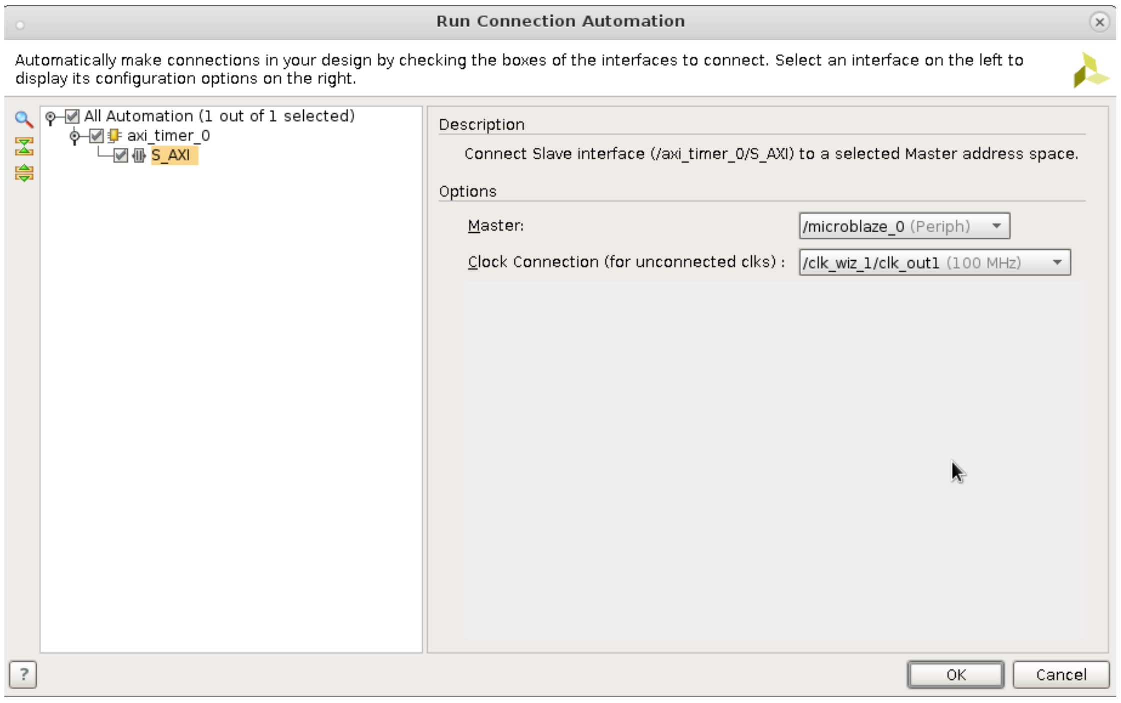
Timer 2

Active state of Capture Trigger: Active High

Active state of Generate Out signal: Active High

1.10 Run connection automation

1.11 For the clock connection in AXI Timer, select the 100Mhz system clock and click OK


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 (1 out of 1 selected)
  ☑ axi_timer_0
    ☑ S_AXI

Description

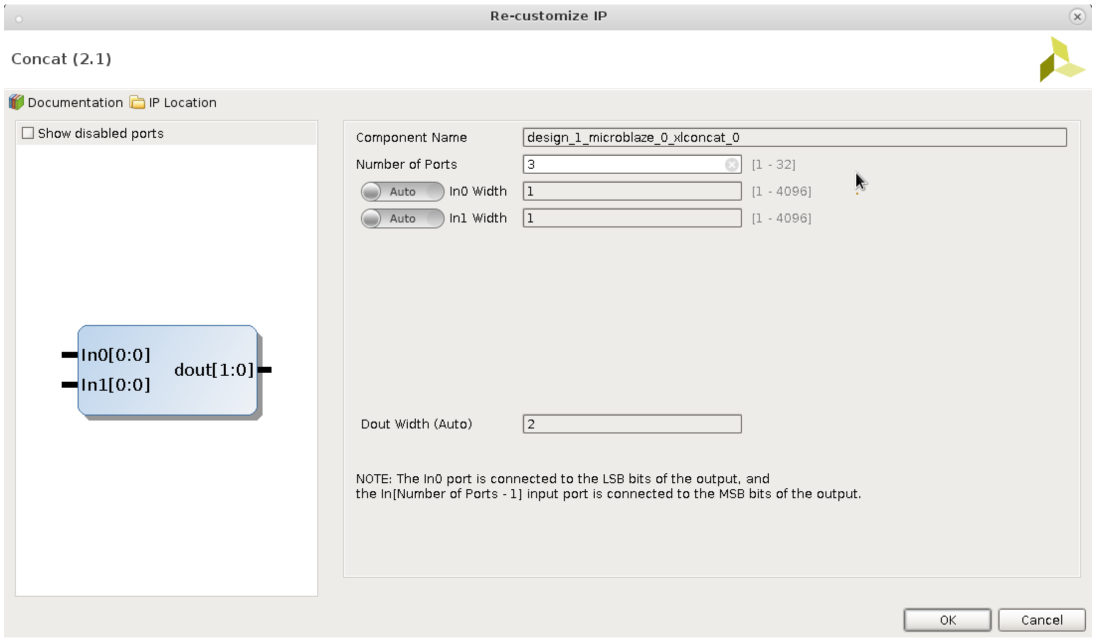
Connect Slave interface (/axi_timer_0/S_AXI) to a selected Master address space.

Options

Master: /microblaze_0 (Periph)

Clock Connection (for unconnected clks) : /clk_wiz_1/clk_out1 (100 MHz)

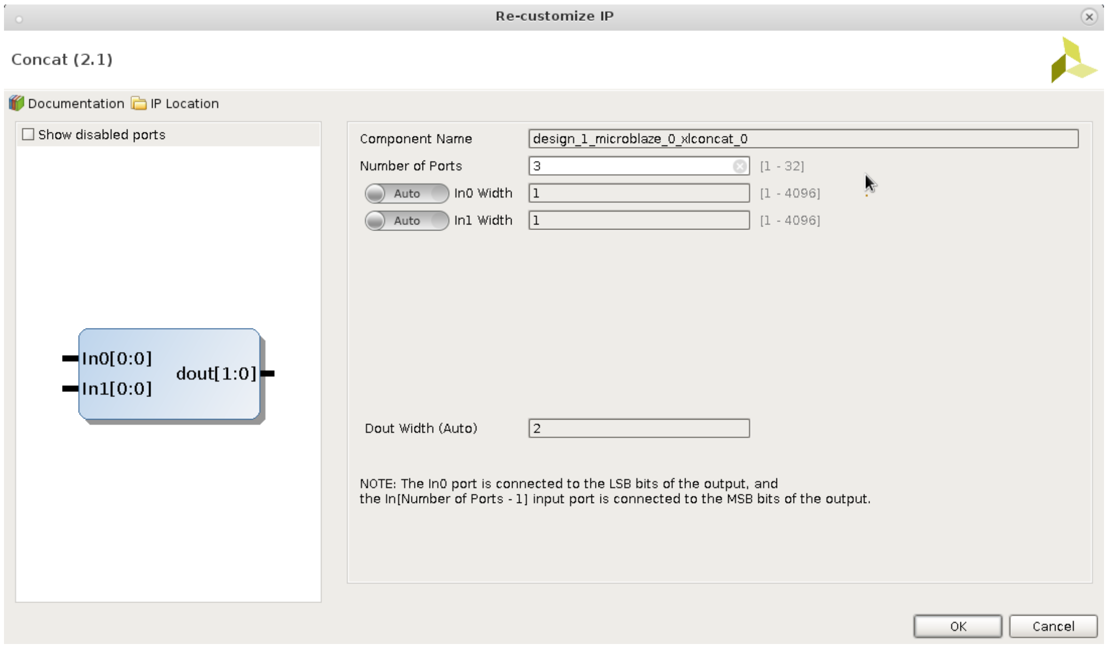
1.12 Double click the concat IP and add in additional input wire: In2



Component Name: design_1_microblaze_0_xlconcat_0
Number of Ports: 3 [1 - 32]
In0 Width: 1 [1 - 4096]
In1 Width: 1 [1 - 4096]
Dout Width (Auto): 2

NOTE: The In0 port is connected to the LSB bits of the output, and
the In[Number of Ports - 1] input port is connected to the MSB bits of the output.

1.13 Manually connect the interrupt of the AXI Timer to In2 of concat



Component Name: design_1_microblaze_0_xlconcat_0
Number of Ports: 3 [1 - 32]
In0 Width: 1 [1 - 4096]
In1 Width: 1 [1 - 4096]
Dout Width (Auto): 2

NOTE: The In0 port is connected to the LSB bits of the output, and
the In[Number of Ports - 1] input port is connected to the MSB bits of the output.

1.14 Run Validate and ensure that it was successful

1.15 Regenerate bit stream

1.16 Export to hardware including the bit stream

1.17 Run SDK to set up the software

2. Setup the Software

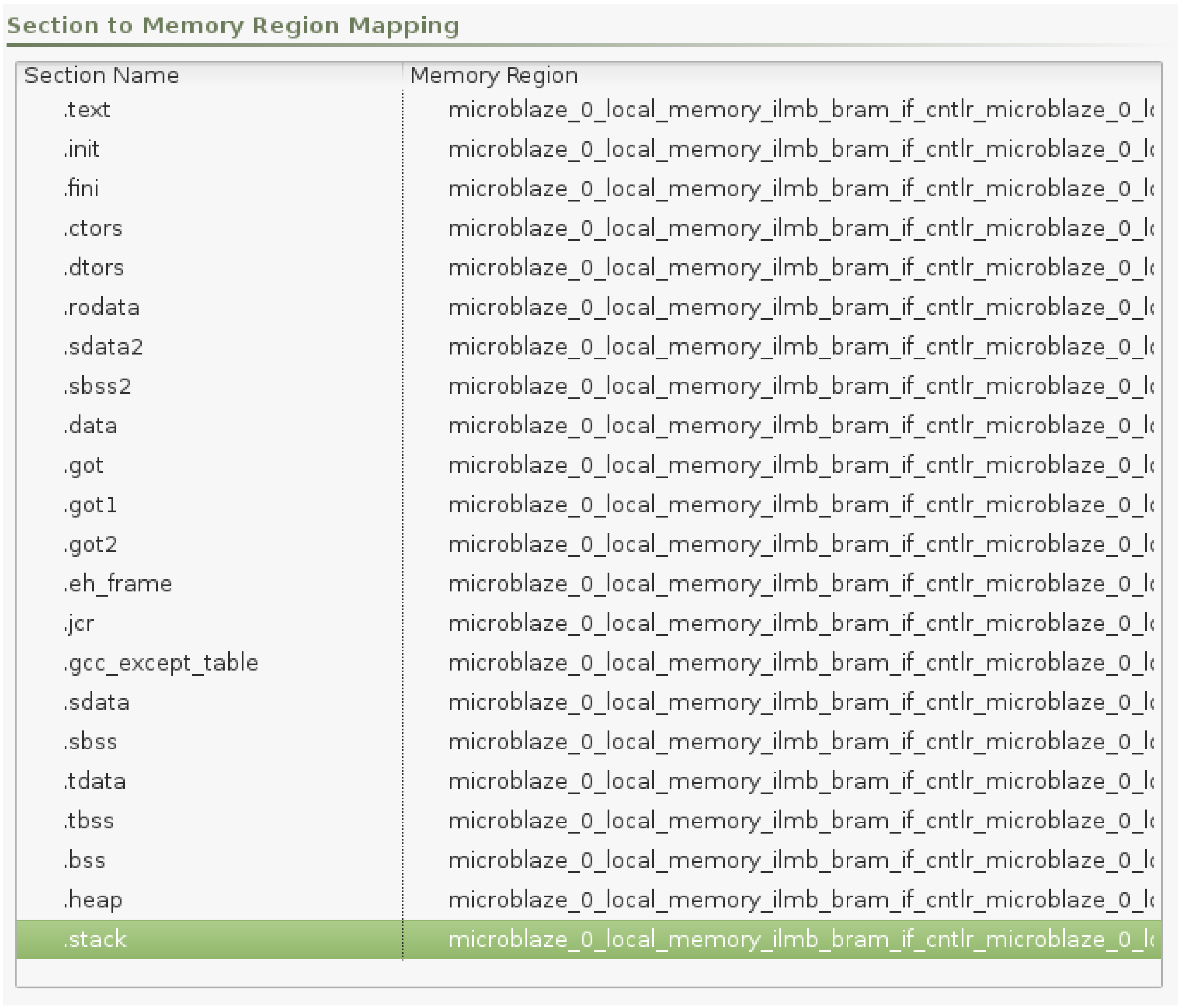
In this section, we will setup the software aspect to enable AXI Timer and allow the processor to accept interrupt from AXI Timer

2.1 Create a new empty Xilinx Application

2.2 Under Source, create a new C source

2.3 Copy and paste the provided source code into your new C source file

2.4 Double click the linker file: lscript.ld and change all values under Memory Region Mapping to MicroBlaze local memory



Section to Memory Region Mapping

Section Name	Memory Region
.text	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.init	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.fini	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.ctors	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.dtors	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.rodata	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.sdata2	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.sbss2	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.data	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.got	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.got1	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.got2	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.eh_frame	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.jcr	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.gcc_except_table	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.sdata	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.sbss	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.tdata	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.tbss	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.bss	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.heap	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l
.stack	microblaze_0_local_memory_ilmb_bram_if_cntlr_microblaze_0_l

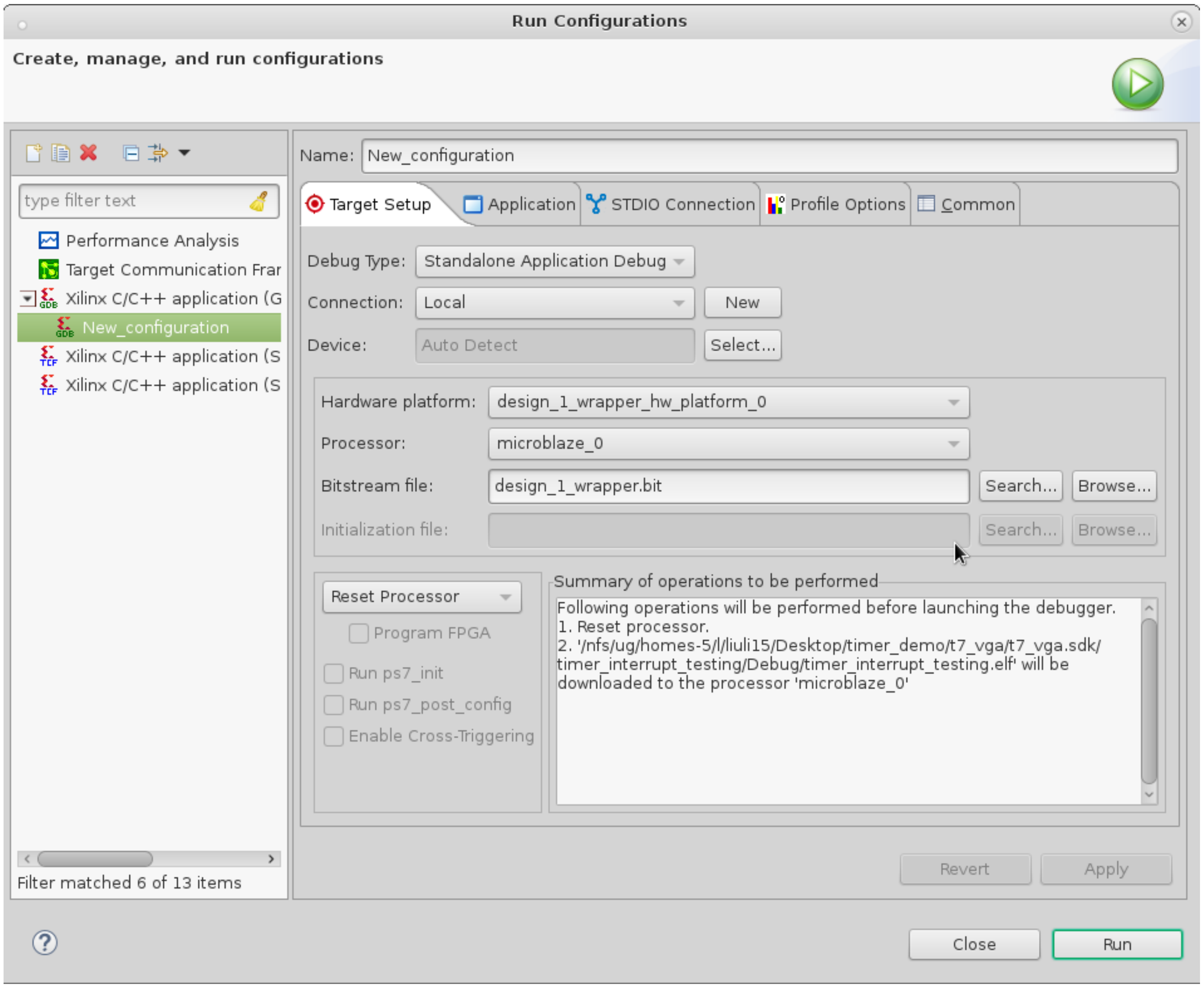
2.5 Save and compile

2.6 Under Xilinx Tools, click Program FPGA. Make sure your board is connected to the computer and is ON.

2.7 Under Run, click Run Configuration

2.8 Double click on Xilinx Application GDB

2.9 Under Target Setup select Reset Entire System and your Xilinx application



Name: New_configuration

Debug Type: Standalone Application Debug

Connection: Local

Device: Auto Detect

Hardware platform: design_1_wrapper_hw_platform_0

Processor: microblaze_0

Bitstream file: design_1_wrapper.bit

Initialization file:

Reset Processor

Summary of operations to be performed

```
Following operations will be performed before launching the debugger.
1. Reset processor.
2. '/nfs/ug/homes-5/l/liuli15/Desktop/timer_demo/t7_vga/t7_vga.sdk/
timer_interrupt_testing/Debug/timer_interrupt_testing.elf' will be
downloaded to the processor 'microblaze_0'
```

2.10 Make sure that the CPU reset (SW5) is in high state and click Run

3. Exercise

Carefully examine the provided source code and the AXI Timer documentation. Instead of using interrupt, change the operating mode of the Timer to polling mode instead.