Lab 3: Creating Custom IPs

Abhinav Subramanian

ECEN 449-502

Due date: 2/11/2020

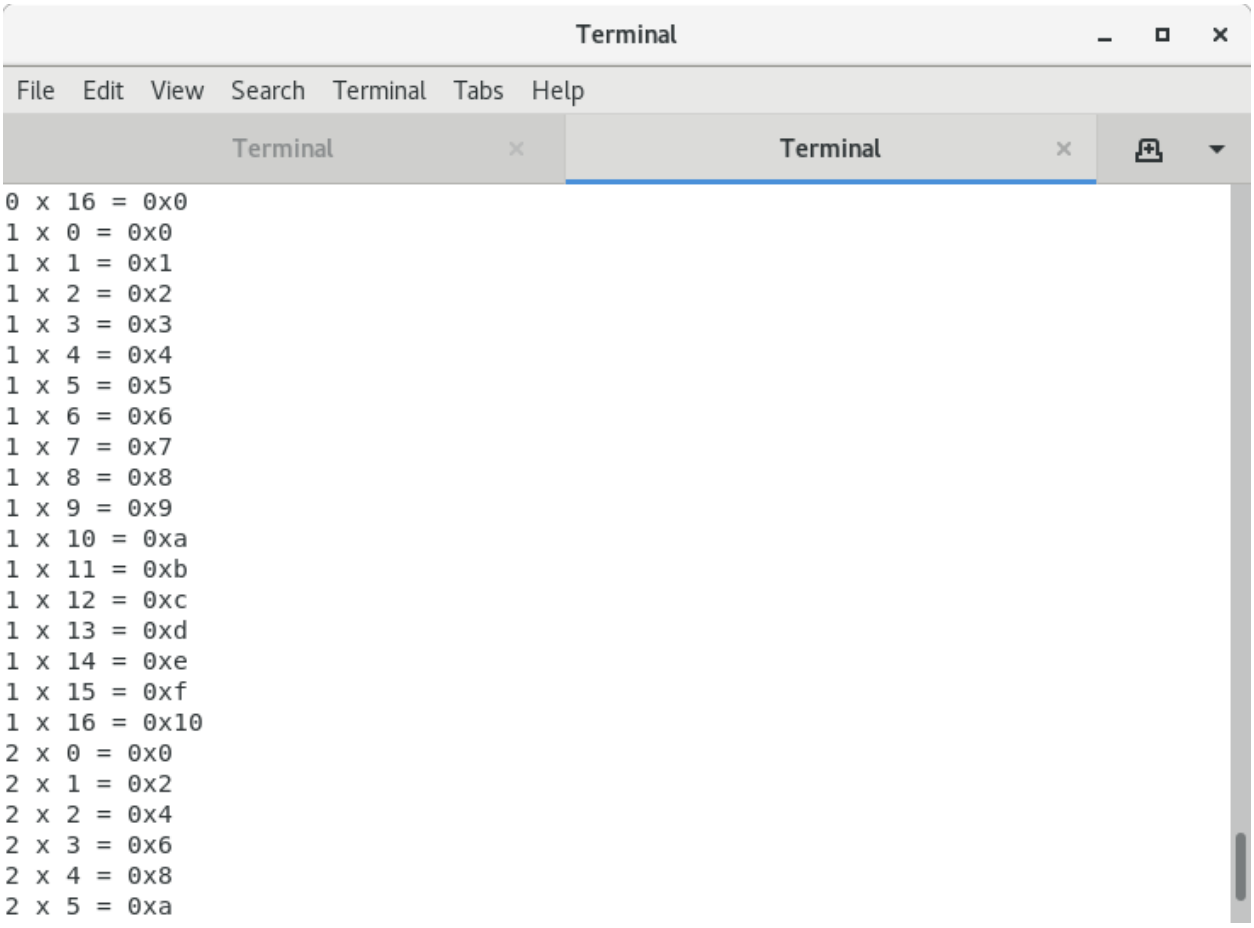
Introduction:

The aim of this lab was to learn how to create a custom IP block in Vivado and write software interacting with it. Firstly, I created an IP that would take in two inputs and output the product of the two inputs. Then, I wrote software that would input various numbers into the IP I programmed and output the result each time the numbers changed.

Procedure:

- Open Vivado and create a new project. But this time, select the ZYBO Z7-10 from the "boards" tab instead.
- Create a new block design and add the Processing System IP to it. Configure it using the TCL file and disable all the unneeded I/O pins.
- Create the multiply IP. Comment out anything writing to slv_reg2 that already exists in the Verilog file. Then copy the code in the lab manual to enable the multiplication ablilities.
- Repackage the IP, and now connect it to the rest of the system, generate the bitstream and export it to the SDK.
- Now, create a new hello world project in the SDK. Copy and paste the code given in the appendix into the helloworld.c file.
- Open a new terminal and start picocom.
- Program the board and run helloworld.c on the ZYBO.
- You should now see the result of two numbers being multiplied being printed to the picocom.

Results:



Conclusion:

In this lab, I learned how to create a custom IP block in Vivado and write software interacting with it. Firstly, I created an IP that would take in two inputs and output the product of the two inputs. Then, I wrote software that would input various numbers into the IP I programmed and output the result each time the numbers changed. The program worked exactly as expected, and I didn't really run into any big issues. The only problem I had was that I couldn't find the board on my machine, so I had to configure the project the normal way.

Postlab questions:

a. The temp_reg is used to store the output temporarily, so it can be written to the output after the I/O is done. If it's removed, it could cause incorrect output/errors, as there would be a delay on the result.
b. Any set of inputs that causes the output's answer to go above 32 bits would result in an overflow error. To deal with this, either increase the output size or split the results between two registers.

Appendix:

helloworld.c

```
/******************************************************************************
**
*
* Copyright (C) 2009 - 2014 Xilinx, Inc.  All rights reserved.
*
* Permission is hereby granted, free of charge, to any person obtaining a
copy
* of this software and associated documentation files (the "Software"), to
deal
* in the Software without restriction, including without limitation the
rights
* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
* copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included in
* all copies or substantial portions of the Software.
*
* Use of the Software is limited solely to applications:
* (a) running on a Xilinx device, or
* (b) that interact with a Xilinx device through a bus or interconnect.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
* XILINX  BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
* WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF
* OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*
* Except as contained in this notice, the name of the Xilinx shall not be
used
* in advertising or otherwise to promote the sale, use or other dealings in
* this Software without prior written authorization from Xilinx.
*
******************************************************************************
*/

/*
 * helloworld.c: simple test application
 *
 * This application configures UART 16550 to baud rate 9600.
 * PS7 UART (Zynq) is not initialized by this application, since
 * bootrom/bsp configures it to baud rate 115200
```

```
 *
 * ------------------------------------------------
 * | UART TYPE   BAUD RATE                        |
 * ------------------------------------------------
 *   uartns550   9600
 *   uartlite    Configurable only in HW design
 *   ps7_uart    115200 (configured by bootrom/bsp)
 */

#include <stdio.h>
#include "platform.h"
#include "xparameters.h"
#include "multiply.h"

void print(char *str);

int main()
{
    init_platform();

    int i=0;
    int j=0;
    for(i=0;i<=16;i++){
        for(j=0; j<=16;j++){
                MULTIPLY_mWriteReg(XPAR_MULTIPLY_0_S00_AXI_BASEADDR,
MULTIPLY_S00_AXI_SLV_REG0_OFFSET,(u32)i); //write to reg1
                MULTIPLY_mWriteReg(XPAR_MULTIPLY_0_S00_AXI_BASEADDR,
MULTIPLY_S00_AXI_SLV_REG1_OFFSET,(u32)j); //write to reg2
                u32 ans=MULTIPLY_mReadReg(XPAR_MULTIPLY_0_S00_AXI_BASEADDR,
MULTIPLY_S00_AXI_SLV_REG2_OFFSET); //read from output
                int actualans=(int)ans;
                printf("%d x %d = 0x%x\n\r",i,j,actualans); //print to
console.
        }
    }
    printf("\n\r");

    cleanup_platform();
    return 0;

}
```