

# Introduction to Systems-on-Chip Multi-Processor Integration

Abigail Butka, Muhammed Kawser Ahmed, Wade Fortney, Christophe Bobda

December 10, 2024

## 1 Objective

The learning objective of this module is to gain hands-on experience with inter-processor communication in Vivado and Vitis. In this module you will build a System-on-Chip system in Vivado 2021.1 with the Xilinx Zynq ARM Cortex-A9 hard-core processor, a soft-core MicroBlaze processor, and the AXI Mailbox IP. Then, perform inter-processor communication in Vitis by initializing the Mailbox IP and creating a custom run configuration that will run the Zynq and MicroBlaze applications at the same time.

For more conceptual details of this module, please refer to the accompanying PowerPoint presentation and video. In the unfortunate case that the following tutorial no longer produces a bitstream due to software updates to Vivado, the bitstream for this module is provided in the accompanying zip file for the PYNQ-Z2 FPGA.

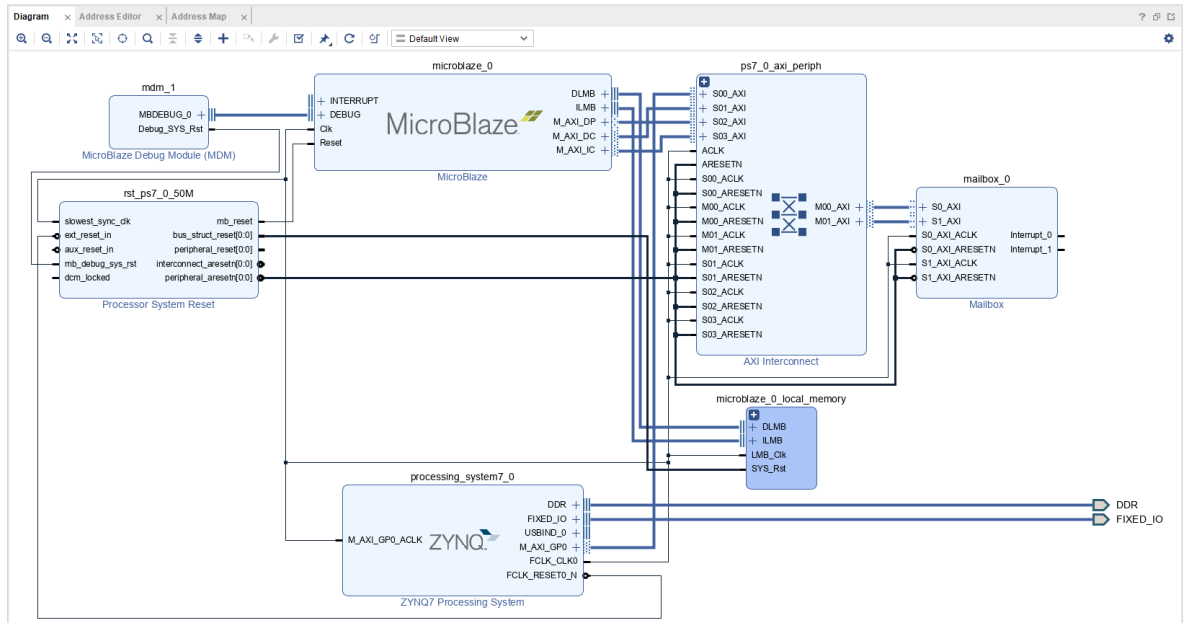
## 2 Equipment and Software Needed for this Module

1. Xilinx Vivado 2021.1.
2. An FPGA with ZYNQ7 processor.
  - (a) This module was tested using the PYNQ-Z2.
3. A USB-UART cable to program the board and display the output in the terminal.

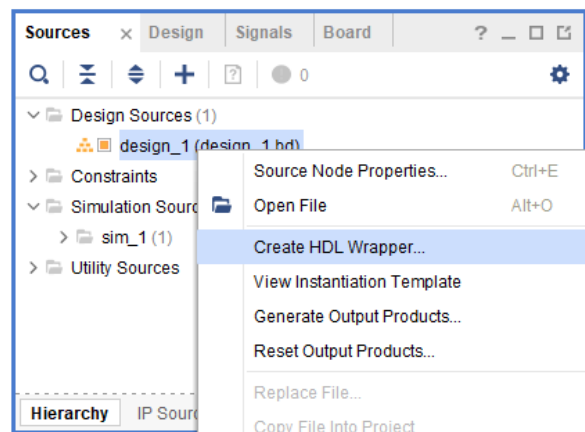
## 3 Module Description

### 3.1 Task 1: Zynq to MicroBlaze Communication

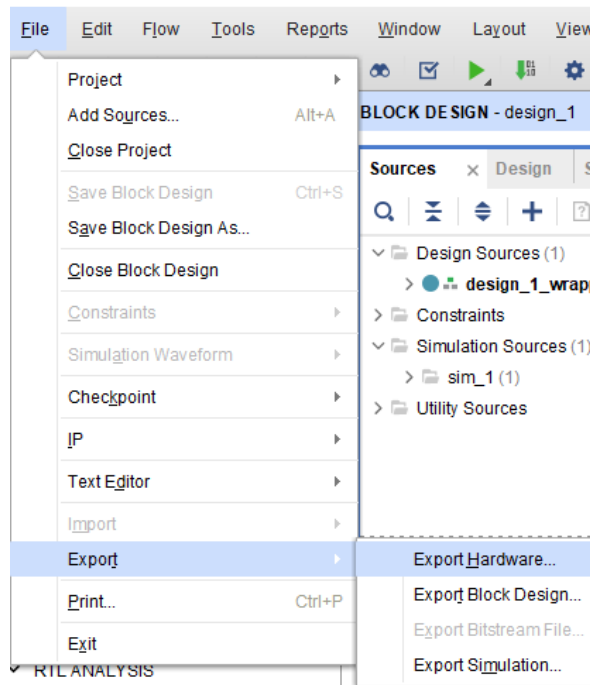
1. Open Vivado and create a new Vivado project. Make sure you choose the FPGA that you have. This tutorial uses the PYNQ-Z2.
2. Create Block Design.
3. Add IP Zynq7 Processing System to the Block Diagram.
4. Add IP MicroBlaze to the Block Diagram.
5. Run Block Automation, select all boxes, and click OK.
6. Next, add IP Mailbox.
7. Next, click Run Connection Automation, select all boxes, and click OK.
8. If the Run Connection Automation appears again, select all boxes, and click OK until the flag no longer appears.
9. Click Regenerate Layout. Your block diagram should look like the image below.



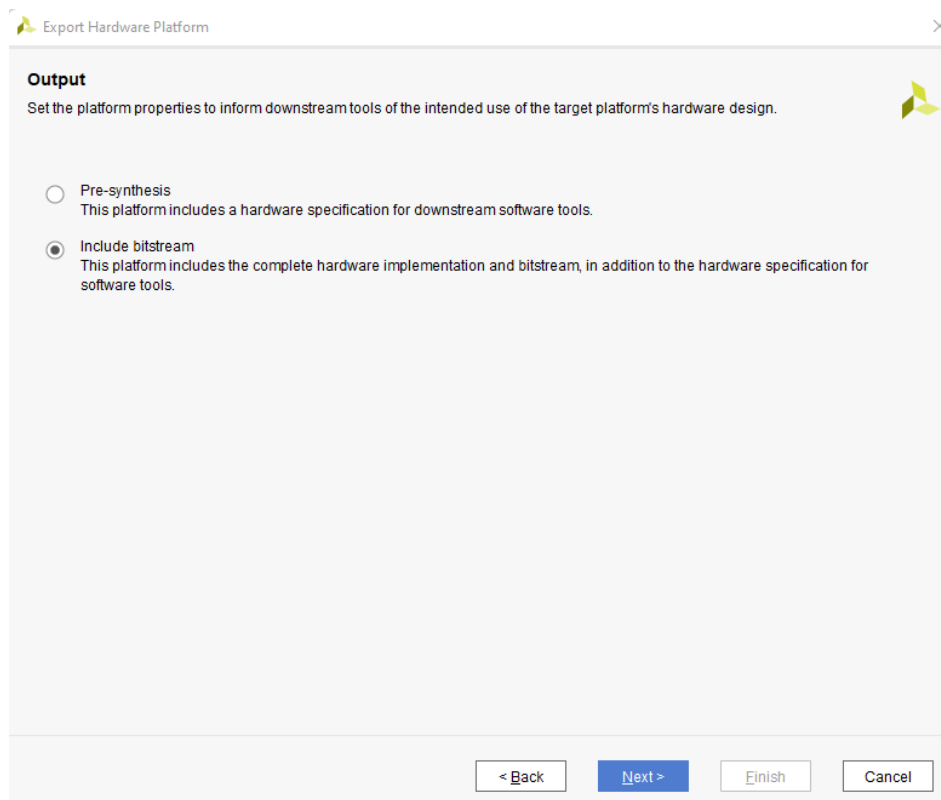
10. Finally, generate the bitstream by clicking on Generate Bitstream in the Flow Navigator.
11. When the window “Bitstream Generation successfully completed” pops up, click cancel.
  - (a) If you do not click cancel this will open the implemented design showing the FPGA pin placements.
  - (b) We do not need to see this information now.
12. Next, navigate to the Sources tab and expand Design Sources.



13. Right-click design\_1.bd and click Create HDL Wrapper. Then select Let Vivado Manage Wrapper and Auto-Update and click OK.
14. This will create a design\_1\_wrapper.v file that will now be the top file of the design.
15. Next, click on File/Export/Export Hardware.



16. Click Next, ensure that 'Include Bitstream' is selected, and click Next again.



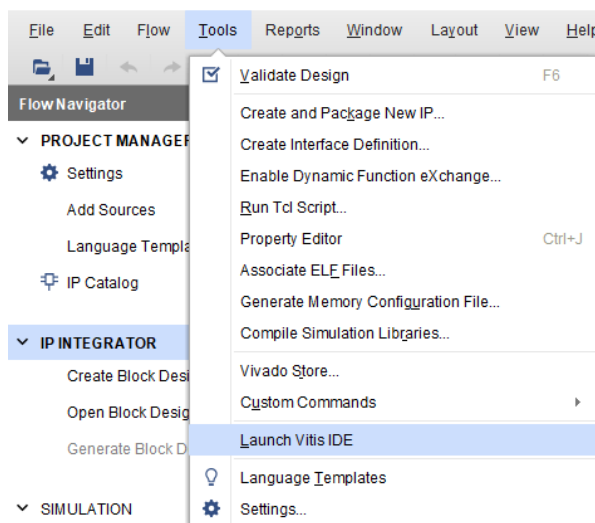
17. Leave the XSA file named design\_1\_wrapper, then click Next and Finish.

18. You have successfully created the hardware description for your FPGA.

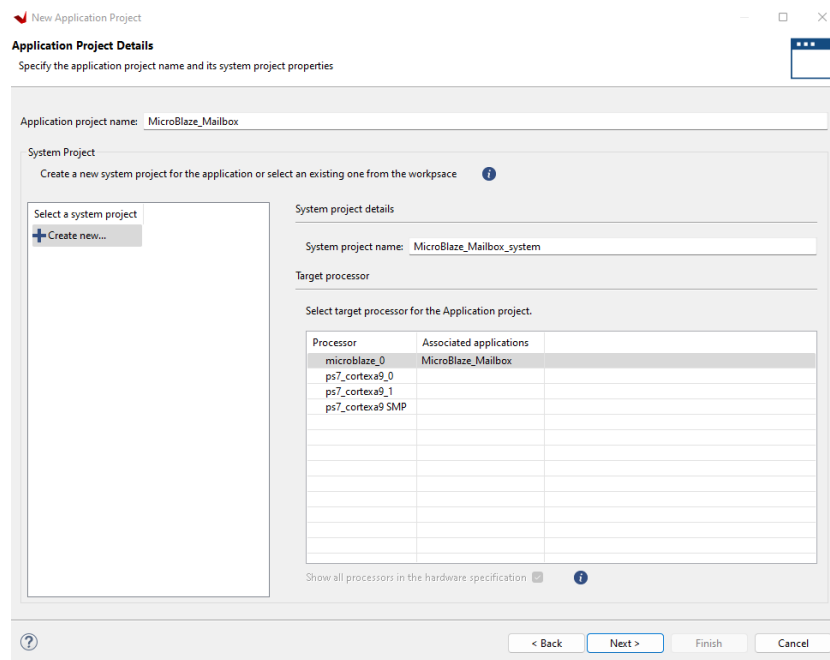
## 3.2 Task 2: Setting up the MicroBlaze Processor

In this task, you will create a simple PetaLinux image to run on the hardware description you created in the previous Task.

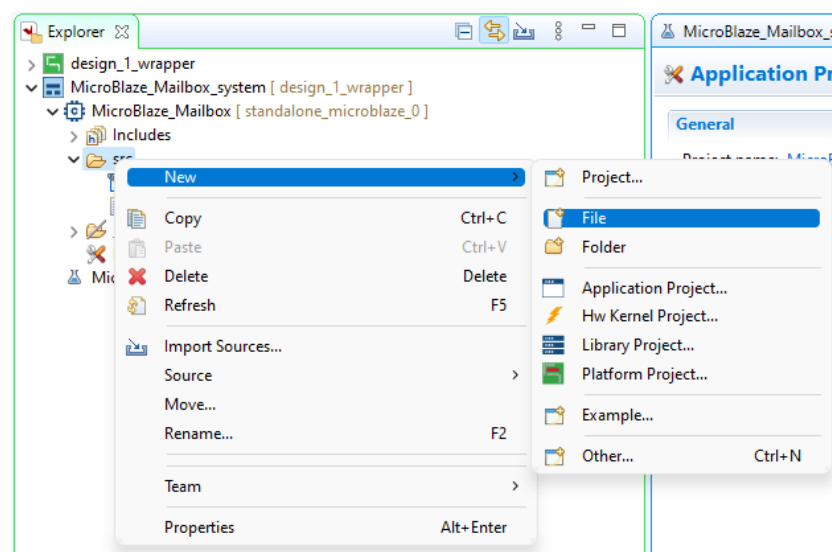
1. Click on Tools/Launch Vitis ID as seen in the image below.



2. When the new window loads titled “Select a directory as workspace”, click on browse.
3. Browse to the location of your Vivado project, you should see your design\_1\_wrapper.xsa.
4. Right-click and create a folder titled “Vitis\_WS.” This is where all the Vitis projects for this hardware design will be stored. Click Select Folder.
5. Click Launch
6. When Vitis loads, click Create Application Project.
7. You will open the home page showing the Welcome Page, click next.
8. You will now be on the Platform Page. Click the “Create a new platform from hardware (XSA)” tab.
9. Click browse, navigate to your Vivado project again, and select the design\_1\_wrapper.xsa file, click next.
10. Name the project MicroBlaze.Mailbox.
11. Ensure the selected processor is microblaze\_0 as seen in the image below.



12. On the Domain Page, click Next.
13. On the Template page, select Empty Application(C).
  - (a) The MicroBlaze does not have built-in access to the UART like the Zynq processor does.
  - (b) If you select Hello World, you will receive a red notification across the top of the screen stating that "This application requires an UART peripheral in the hardware."
  - (c) If you want to have the MicroBlaze talk over UART, you must add the AXI Uartlite IP to your design.
14. Click Finish. This will create both the platform project and application project in your Vitis Workspace.
15. Next, expand the folder MicroBlaze\_Mailbox\_system and right-click on the src folder to create a new file.



16. Name this new file MicroBlaze\_Mailbox.c.
17. The file will open after being created. Paste the following code:

```

#include <stdio.h>
#include "xil_printf.h"
#include "xparameters.h"
#include "sleep.h"
#include "xmbox.h"
#include "xmbox_hw.h"

#define DELAY 200000

int main()
{
    XMbox Mbox;
    XMbox_Config *ConfigPtr;

    u32 Tx_data;

    ConfigPtr = XMbox_LookupConfig(XPAR_MAILBOX_0_IF_0_DEVICE_ID);
    XMbox_CfgInitialize(&Mbox, ConfigPtr, ConfigPtr->BaseAddress);

    while(1){
        Tx_data = 30;
        XMbox_WriteBlocking(&Mbox, &Tx_data, 4);
        usleep(DELAY);
        Tx_data = 3;
        XMbox_WriteBlocking(&Mbox, &Tx_data, 4);
        usleep(DELAY);
    }

    return 0;
}

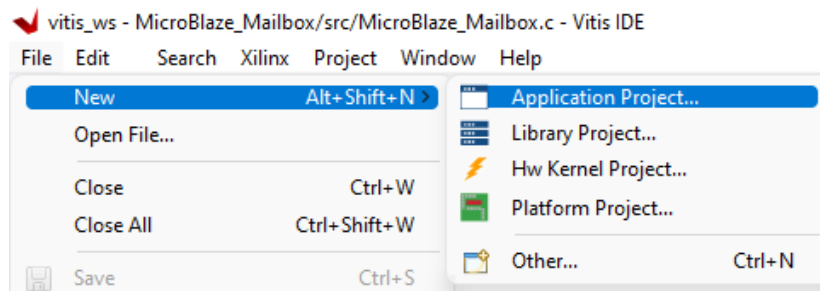
```

18. This code, initializes the MicroBlaze's pointers and configuration information for the Mailbox, then writes alternating data, '30' then '3', to the Mailbox.
19. Save the file, then right-click on the blue ZYNQ\_Mailbox\_System application project and select "Build Project."

### 3.3 Task 3: Setting up the Zynq Processor

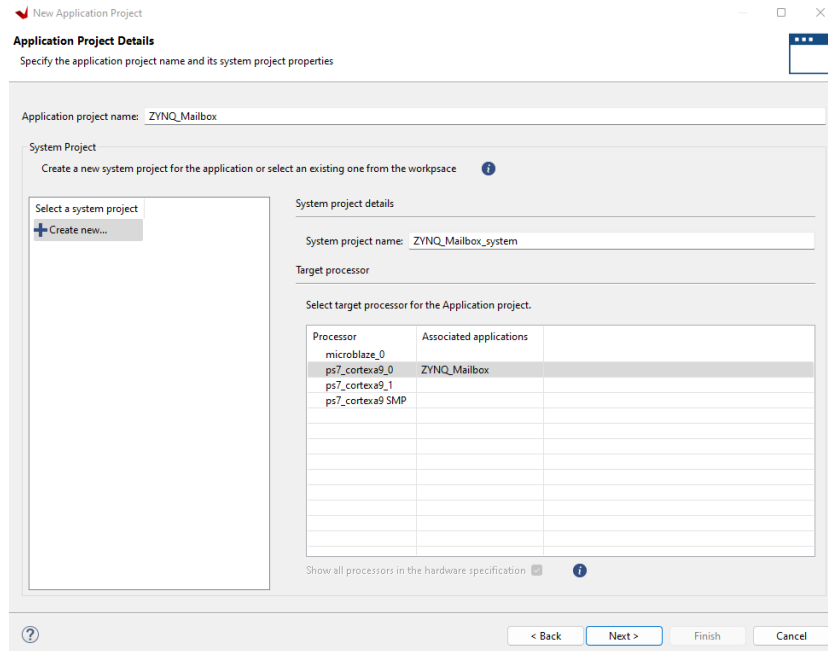
Now that the MicroBlaze is sending data to the Mailbox, we need the Zynq processor to read it.

1. Now, on the top left of the Vitis window, select File/New/Application Project.



2. You will now be on the Platform Page. Click the "Create a new platform from hardware (XSA)" tab.

3. Click browse, navigate to your Vivado project again, and select the design\_1\_wrapper.xsa file, click next.
4. Name the project ZYNQ\_Mailbox.
5. Ensure the selected processor is ps7\_cortexa9\_0 as seen in the image below. This selects Core 0 of the Zynq processor's two cores. Click Next.



6. On the Domain Page, click Next.
7. On the Template page, select Hello World.
  - (a) The Zynq has built-in access to the UART, unlike the MicroBlaze.
  - (b) If your Zynq board provides an error similar to that seen in Task 2, Instruction 13.b, add the AXI UART IP to your design. This should fix the issue.
8. Click Finish to create both the platform project and application project in your Vitis Workspace.
9. Next, locate the helloworld.c file under ZYNQ\_Mailbox\_System/HelloWorld/src/helloworld.c and double-click on to open it.
10. Paste the following code:

```
#include <stdio.h>
#include "platform.h"
#include "xparameters.h"
#include "xil_printf.h"
#include "xmbox.h"
#include "xmbox_hw.h"

#define MSGSIZ 4

int main()
{
    XMbox Mbox;
    XMbox_Config *ConfigPtr;
```

```

u32 RecvMsg;
init_platform();

print("Hello World\n\r");
print("Successfully ran Hello World application");

ConfigPtr = XMbox_LookupConfig(XPAR_MAILBOX_0_IF_1_DEVICE_ID);
XMbox_CfgInitialize(&Mbox, ConfigPtr, ConfigPtr->BaseAddress);
printf("Hello World, %d\n\r",strlen("hello"));
memset(&RecvMsg, 0, MSGSIZ);
XMbox_ReadBlocking(&Mbox, &RecvMsg, 4);
printf("Rcvd the message --> \r\n\r\n\t--[%lu]--\r\n\r\n", RecvMsg);

while(1){
    XMbox_ReadBlocking(&Mbox, &RecvMsg, 4);
    printf("Rcvd the message --> \r\n\r\n\t--[%lu]--\r\n\r\n", RecvMsg);
}
cleanup_platform();
return 0;
}

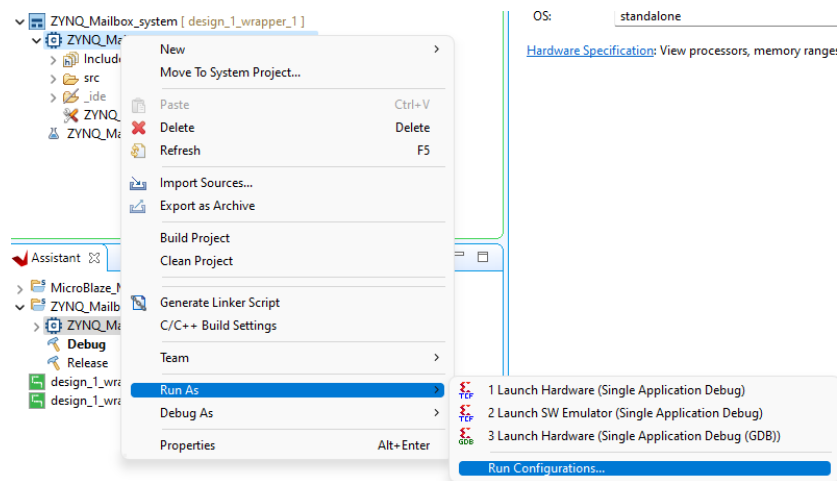
```

11. The above code initializes the Zynq's pointers and configuration information for the Mailbox. The while loop continuously runs the XMbox\_ReadBlocking() command which waits for data to enter the Mailbox, which then reads that data so it can be printed to the terminal.
12. Save the file, then right-click on the blue ZYNQ\_Mailbox\_System application project and select "Build Project"

### 3.4 Task 4: Vitis: Running Programs on Multiple Processors

Running a Vitis project with multiple Application and Platform projects is approached in a slightly different way than previous modules.

1. To begin, right-click on ZYNQ\_Mailbox\_system/ZYNQ\_Mailbox.
2. Then, select Run As/Run Configurations...



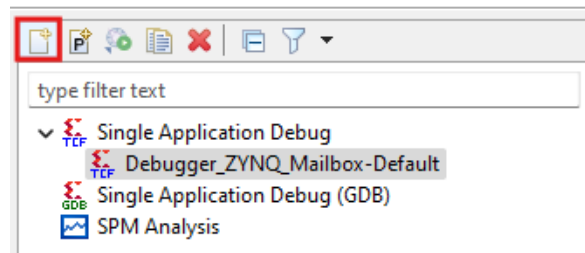
3. In the Debug Configurations window, select Single Application Debug and click on the sheet of paper with plus sign in the top left.



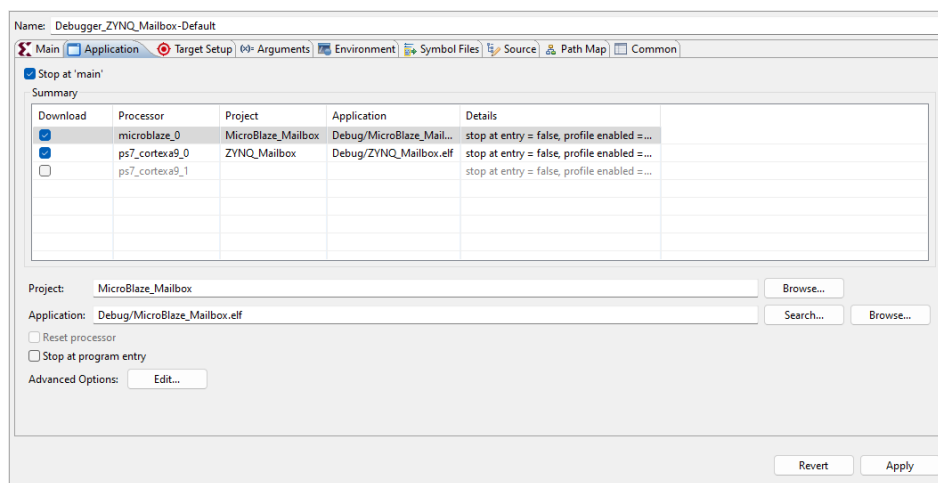


## Create, manage, and run configurations

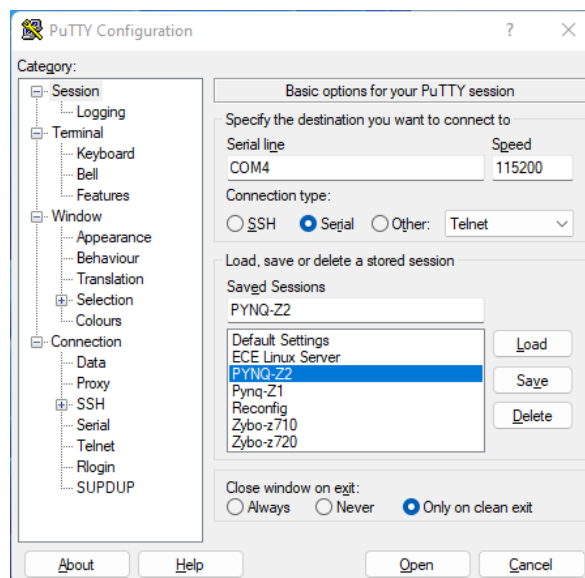
Debug a program using Application Debugger.



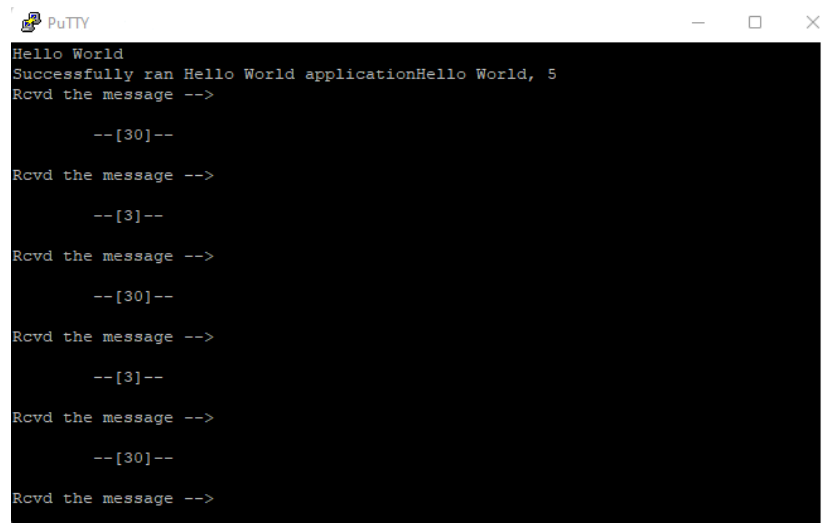
- Next, click on the Application tab. Click on the checkbox next to microblaze\_0. Both microblaze\_0 and ps7.cortexa9\_0 should be selected.



- Click Apply to save this configuration.
- Now, plug your PYNQ-Z2 or other FPGA in and turn it on.
- Open PuTTY, select Serial, set the COM port to your identified COM port, and set the speed to 115200, as seen in the image below.



8. Once your PuTTY has connected, go back to the Vitis Run Configuration window and click run. The data of repeating 30's and 3's will display in the terminal as shown below.



```
PuTTY
Hello World
Successfully ran Hello World applicationHello World, 5
Rcvd the message -->
    --[30]--
Rcvd the message -->
    --[3]--
Rcvd the message -->
    --[30]--
Rcvd the message -->
    --[3]--
Rcvd the message -->
    --[30]--
Rcvd the message -->
```