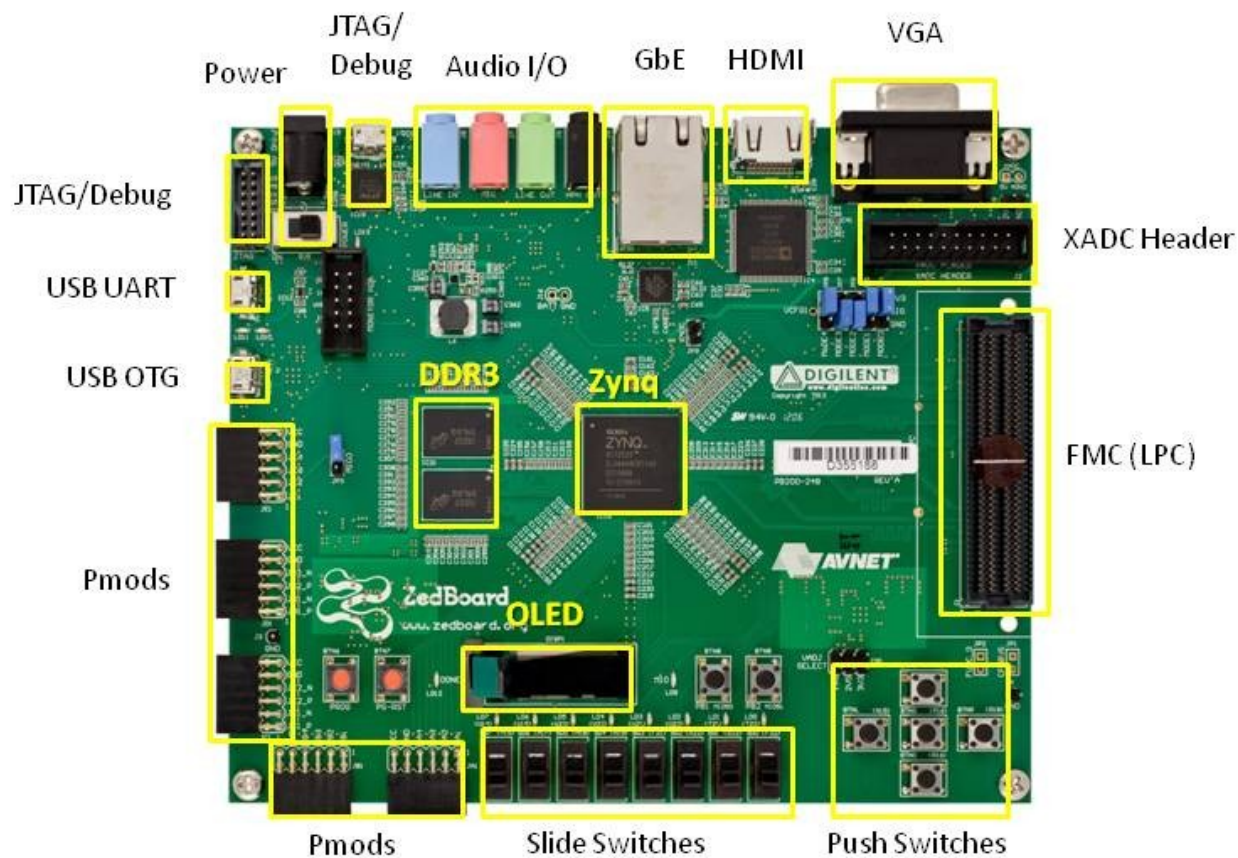


ZedBoard DAQ

Jia Fu Low
James Bueghly
Jovana Andrejevic

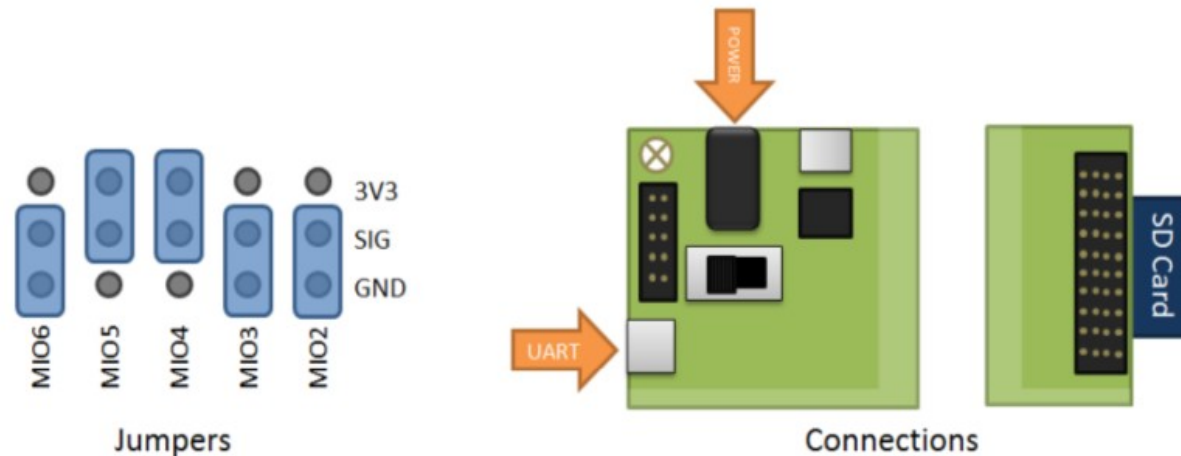
July 27, 2015

ZedBoard Schematic



* SD card cage and QSPI Flash reside on backside of board

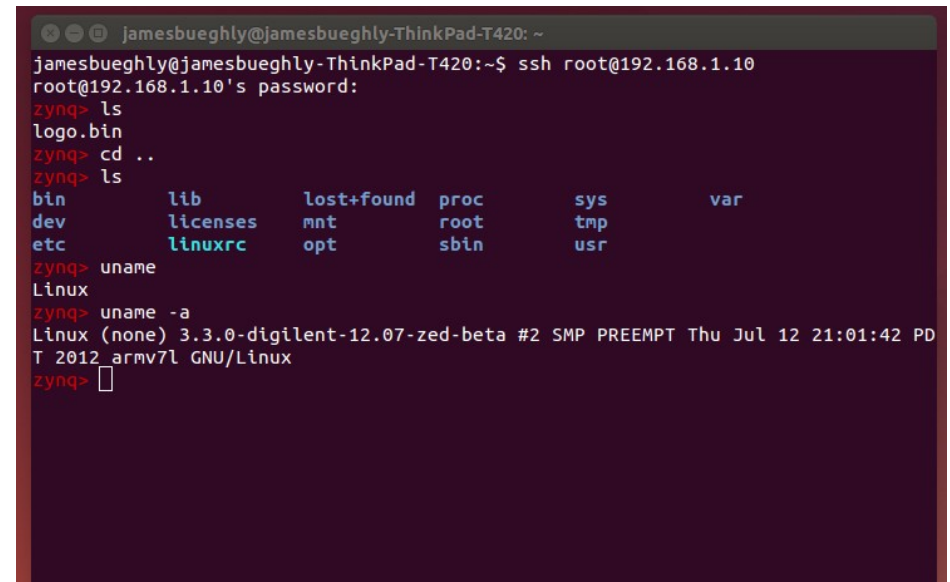
Booting Linux on Zynq



- Boot from SD card using GTKterm terminal emulator
- UART connection settings:
 - Port: `/dev/ttyACM0`
 - Baud rate: 115200
- Simple built-in commands to access peripherals
 - `write_led`

Connecting Via Ethernet

- Standard ethernet connection between ZedBoard and PC
- Network properties
 - IP address: 192.168.1.1
 - Subnet mask: 255.255.255.0
 - Gateway: blank
- Connect via SSH → file transfer via SCP



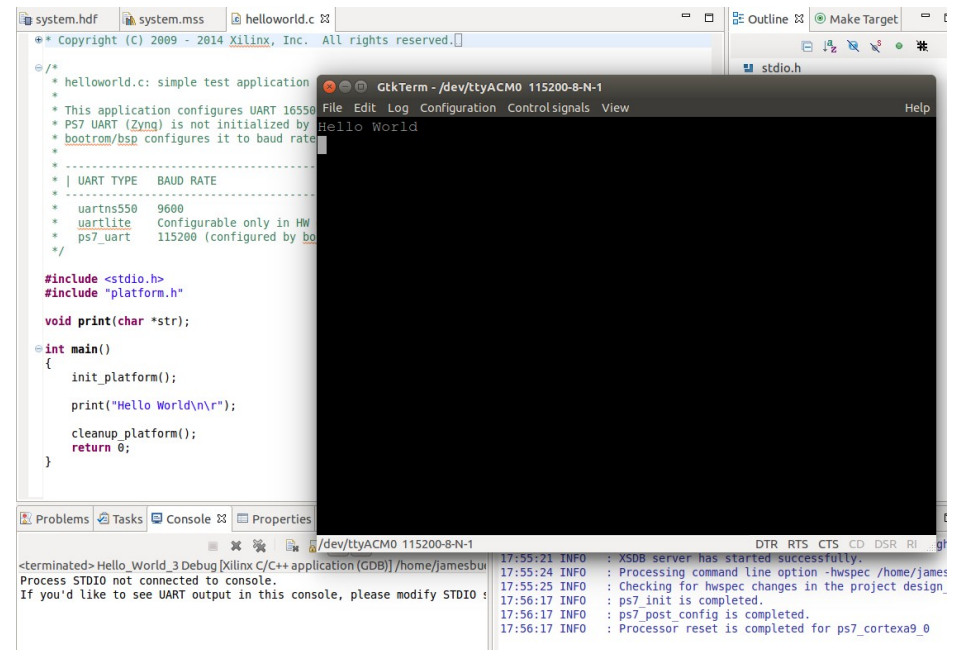
```
jamesbueghly@jamesbueghly-ThinkPad-T420: ~  
jamesbueghly@jamesbueghly-ThinkPad-T420:~$ ssh root@192.168.1.10  
root@192.168.1.10's password:  
zynq> ls  
logo.bin  
zynq> cd ..  
zynq> ls  
bin          lib          lost+found   proc         sys          var  
dev          licenses     mnt          root         tmp  
etc          linuxrc      opt          sbin         usr  
zynq> uname  
Linux  
zynq> uname -a  
Linux (none) 3.3.0-digilent-12.07-zed-beta #2 SMP PREEMPT Thu Jul 12 21:01:42 PD  
T 2012 armv7l GNU/Linux  
zynq> 
```

Vivado & SDK Procedure

- 1) Create Vivado project
- 2) Develop block design with IP cores
- 3) Modify connections/settings and validate design
- 4) Generate an HDL wrapper
- 5) Synthesis, implementation and bitstream generation
- 6) Program the ZedBoard hardware
- 7) Export to SDK and develop board support package
- 8) Develop software application to execute on the board

Hello World

- Jumper pins in cascaded JTAG mode
- Basic hardware design in Vivado including only PS
- Write helloworld.c in SDK and run on ZedBoard
- Output displayed in GTKterm window



The screenshot displays the Xilinx IDE interface. The main editor window shows the source code for `helloworld.c`, which is a simple test application that configures the UART and prints "Hello World". The code includes comments about the application's purpose and configuration details. Below the editor, the `Console` window shows the output of the program, which is "Hello World". The `GTKterm` window is also visible, showing the same output. The `Console` window also displays system messages related to the XSD8 server and the processing of command line options.

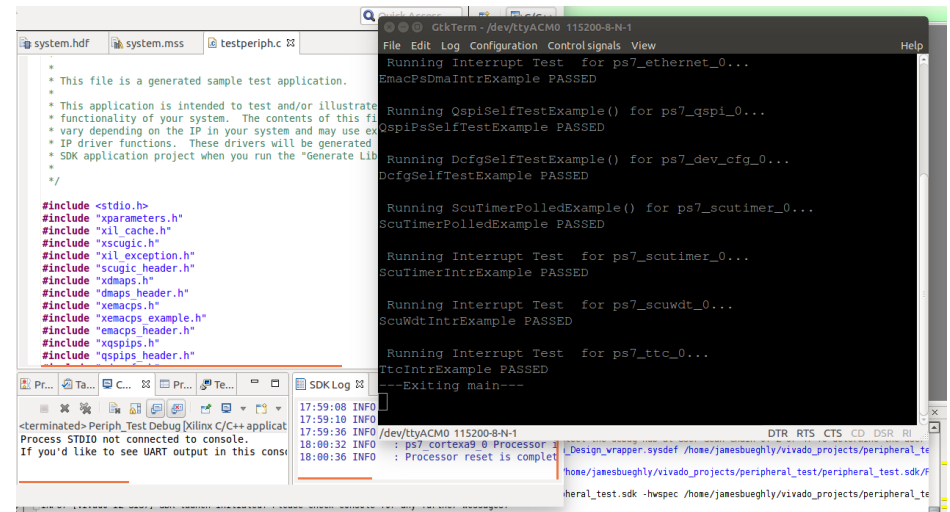
```
/*  
 * helloworld.c: simple test application  
 * This application configures UART 16550  
 * PS7 UART (Zynq) is not initialized by  
 * bootrom/bsp configures it to baud rate  
 */  
-----  
| UART TYPE  BAUD RATE  
-----  
 * uarts550   9600  
 * uartlite   Configurable only in HW  
 * ps7_uart   115200 (configured by bo  
 */  
  
#include <stdio.h>  
#include "platform.h"  
  
void print(char *str);  
  
int main()  
{  
    init_platform();  
    print("Hello World\n\r");  
    cleanup_platform();  
    return 0;  
}
```

GTKterm - /dev/ttyACM0 115200-8-N-1
Hello World

<terminated> Hello_World_3 Debug [Xilinx C/C++ application (GDB)] /home/jamesbu
Process STDIO not connected to console.
If you'd like to see UART output in this console, please modify STDIO :
17:55:21 INFO : XSD8 server has started successfully.
17:55:24 INFO : Processing command line option -hwspec /home/james
17:55:25 INFO : Checking for hwspec changes in the project design
17:56:17 INFO : ps7_init is completed.
17:56:17 INFO : ps7_post_config is completed.
17:56:17 INFO : Processor reset is completed for ps7_cortexa9_0

Memory and Peripherals

- Hardware design utilizes only PS
- Use SDK templates to test memory and peripherals
- Relevant information displayed in GTKterm window



The screenshot displays a development environment with two main windows. The left window shows a C source file named `testperiph.c` with the following content:

```
* This file is a generated sample test application.
* This application is intended to test and/or illustrate
* functionality of your system. The contents of this fi
* vary depending on the IP in your system and may use ex
* IP driver functions. These drivers will be generated
* SDK application project when you run the "Generate Lib
*/

#include <stdio.h>
#include "xparameters.h"
#include "xil_cache.h"
#include "xscugic.h"
#include "xil_exception.h"
#include "xscugic_header.h"
#include "xdmapi.h"
#include "dmaps_header.h"
#include "xemacps.h"
#include "xemacps_example.h"
#include "xemacps_header.h"
#include "xqspips.h"
#include "xqspips_header.h"
```

The right window is a terminal titled `GtkTerm - /dev/ttyACM0 115200-8-N-1`. It shows the output of the application, which includes several test results:

```
Running Interrupt Test for ps7_ethernet_0...
EmacPsDmaIntrExample PASSED

Running QspiSelfTestExample() for ps7_qspi_0...
QspiPsSelfTestExample PASSED

Running DcfgSelfTestExample() for ps7_dev_cfg_0...
DcfgSelfTestExample PASSED

Running ScuTimerPolledExample() for ps7_scutimer_0...
ScuTimerPolledExample PASSED

Running Interrupt Test for ps7_scutimer_0...
ScuTimerIntrExample PASSED

Running Interrupt Test for ps7_scuwdt_0...
ScuWdtIntrExample PASSED

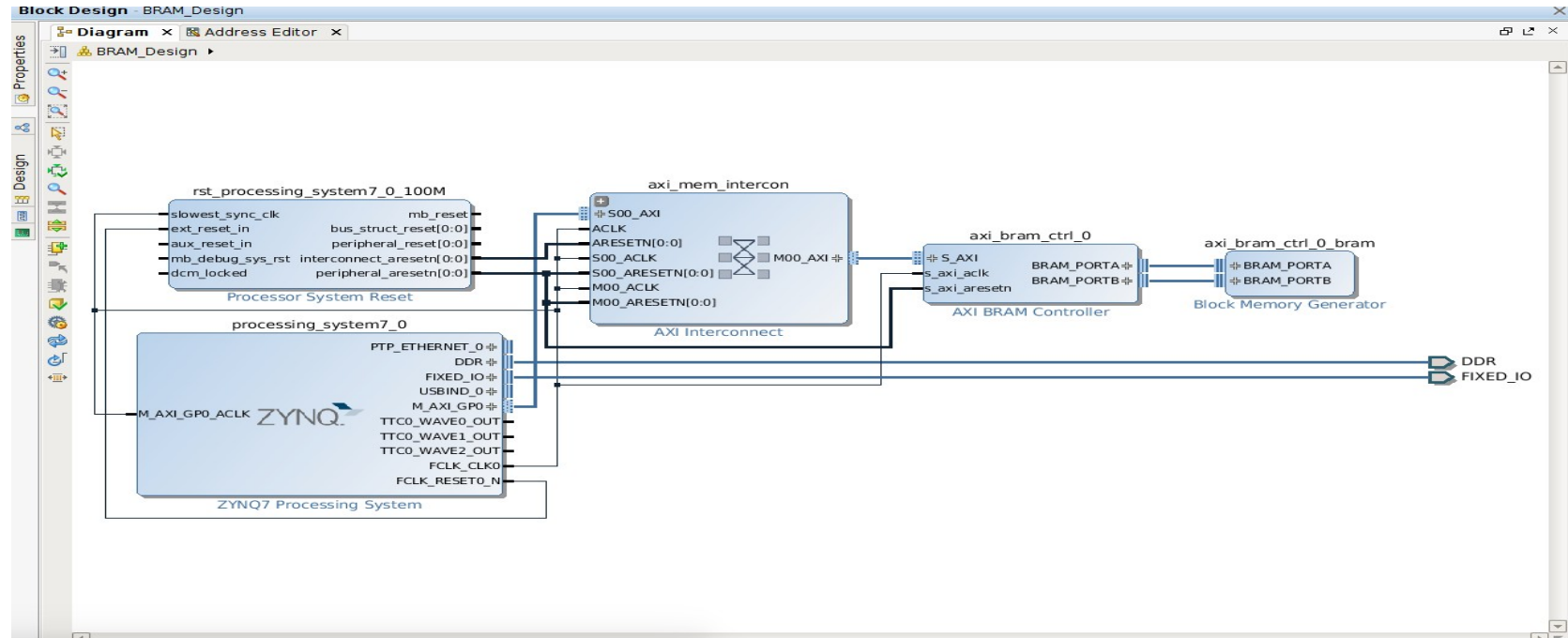
Running Interrupt Test for ps7_ttc_0...
TtcIntrExample PASSED

---Exiting main---
```

At the bottom of the terminal, there is a log of system events:

```
17:59:08 INFO /dev/ttyACM0 115200-8-N-1
17:59:10 INFO : ps7_cortexa9 @ Processor
18:00:32 INFO : Processor reset is complet
18:00:36 INFO
```

Data Transfer Tests



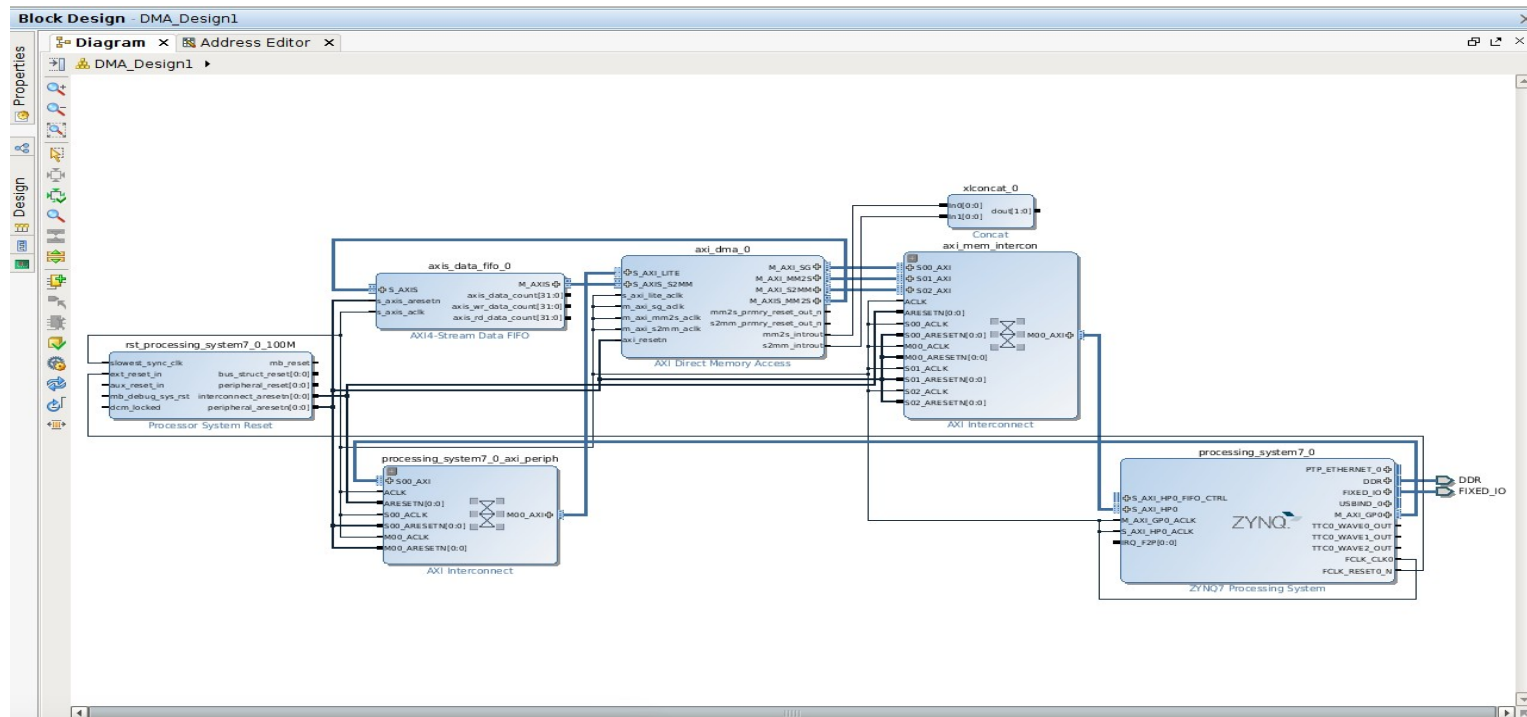
- Add BRAM, BRAM controller, and AXI-interconnect in Vivado block design
- Execute C script to test transfers between memory locations
 - BRAM → BRAM
 - BRAM → DDR3
 - DDR3 → DDR3

Transfer Test Results

Bytes transferred	BRAM/BRAM	BRAM/DDR3	DDR3/DDR3
256	18	10	2
512	20	11	2
1024	20	11	2
2048	21	11	2
4096	21	11	3
8192	21	11	3

Table 1: **Gain in data transfer rate** (ratio of clock cycles elapsed without DMA/clock cycles elapsed with DMA). This table compares data transfer rates with and without the use of a DMA Controller, varying the number of bytes transferred as well as the source and destination pairs (specified by the notation "source/destination"). We can observe that DMA is particularly advantageous for large data transfers involving PL components, such as the BRAM.

Data Transfer with FIFO

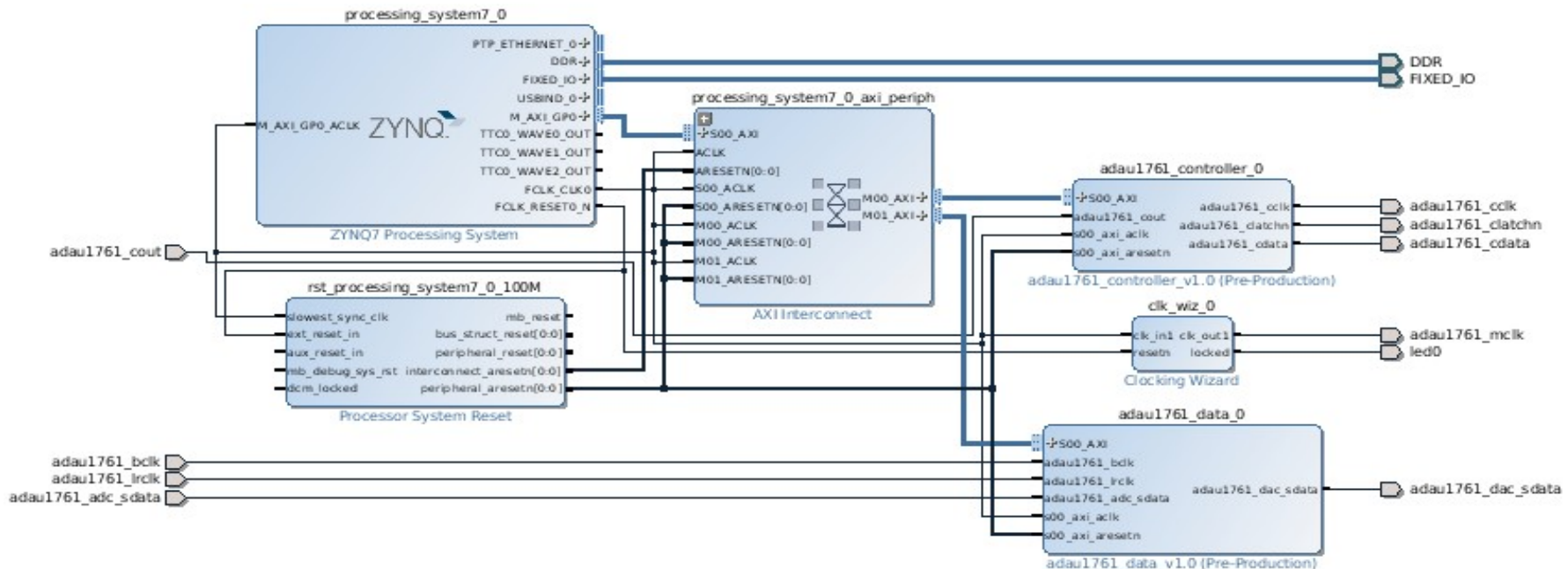


- Add FIFO loopback IP in PL
- Transfer: DDR3 → DDR3 using DMA

Data Transfer from Switch Register

- Input 8-bit data using switches on ZedBoard
- Include GPIO IP core in Vivado block design
 - Make connections external to access switches
- Transfer: Switches → BRAM → DDR3
 - Transfer both with and without DMA
- Read input data out in GTKterm window

Audio with ZedBoard



- Create custom IP: adau1761 controller and data
- Create and add to constraints file to assign external pins
- Audio in from computer → ZedBoard → audio out to headphones