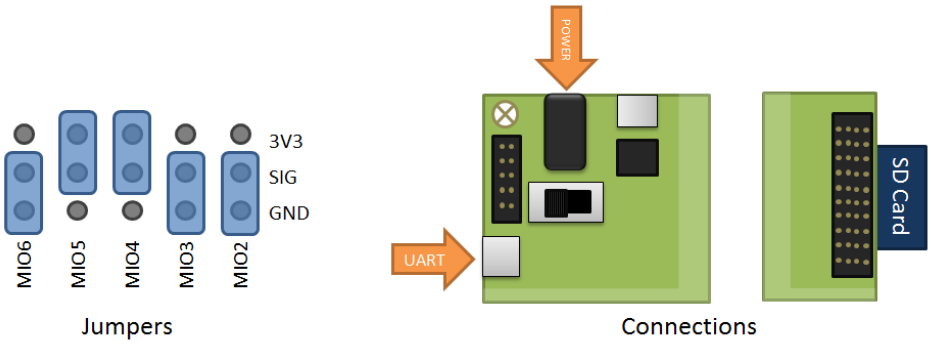
1 Booting Linux from SD Card

1.1 Primary reference(s):

Zynq design from scratch, Part 21 (`http://svenand.blogdrive.com/archive/180.html#.Va7yrvlViko`)

1.2 Configurations and Connections:



1.2.1 Extra Setup:

Install GTKterm terminal emulator, which will be the terminal over which the Zynq Processing System can be accessed, for communication via UART.

1.3 Notes:

The following are sample commands that may be executed directly from the Zynq terminal once the Linux OS is booted:

1. `zynq> uname -a` *Print information about the system (all)*
   ```
   Linux (none) 3.3.0-digilent-12.07-zed-beta #2 SMP PREEMPT Thu Jul 12 21:01:42 PDT 2012
   armv7l GNU/Linux
   ```
2. `zynq> read_sw` *Output state of switches in hexadecimal and decimal*
   ```
   0xff 255
   ```
3. `zynq> write_led 255` *Specify an LED configuration (0-255) to illuminate*
   `zynq> write_led 0`
4. `zynq> unload_oled` *Turn off/on the OLED display*
   `zynq> load_oled`
5. `zynq> poweroff` *Initiate proper powerdown sequence to preserve OLED display life*

1.4 Step-by-Step

1. Set up the ZedBoard according to the Configurations and Connections.
2. Open GTKterm and configure with the following settings, specific to the ZedBoard UART connection:

   Port : /dev/ttyACM0
   Baud Rate : 115200
   Bits : 8
   Stopbits : 1

Parity : none
Flow Control : none

3. Turn on the ZedBoard and wait for complete power up.

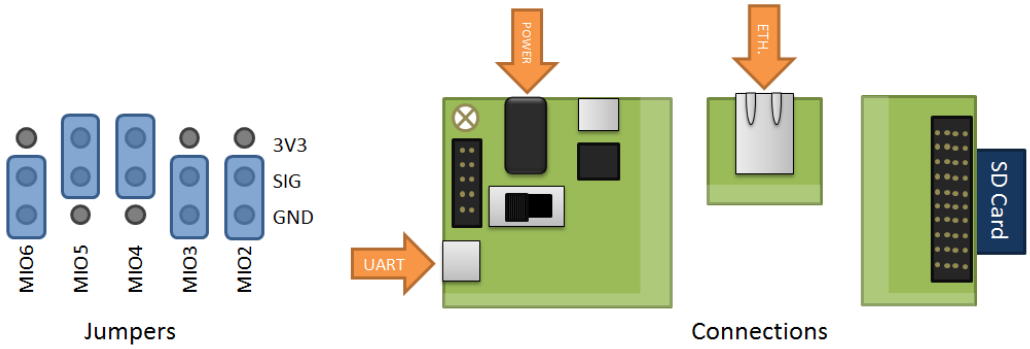
4. If `zynq>` prompt appears in GTKterm, Linux has booted and several commands can be executed to control LEDs, switches, and OLED display. If no prompt appears, type `help` in TGKterm, then enter `boot` following the `zed-boot>` prompt and wait for the boot to complete.

2 Establishing Ethernet Connection

2.1 Primary reference(s):

ZedBoard Getting Started Guide, Demo 5 - Ethernet

2.2 Configurations and Connections:



2.3 Extra Setup:

After configuring the host PC wired network connection, we added a resolv.conf file to the /etc directory of the Zynq Linux system which identified the host PC IP address as a default router, but this was ultimately not necessary for actions such as ssh from the host PC.
(Reference: `http://zedboard.org/content/zedboard-network-bring`)

Original directory:

```
zynq> cd /etc
zynq> ls
dropbear      init.d        inittab.orig  passwd
fstab         inittab       mtab          profile
```

Create and write to resolv.conf file:

```
zynq> cd /etc
zynq> touch resolv.conf
zynq> vi resolv.conf
```

And write

```
nameserver 192.168.1.1
```

Then, add the route of your router:

```
zynq> route add default gw 192.168.1.1 eth0
```

This identifies the host PC (with a manually set IP address of 192.168.1.1) as the default gateway (router) for the ZedBoard, and allows the ZedBoard to connect to the outside world via Ethernet. It may be relevant for communication *from* the Zynq in the future.

2.4 Step-by-Step

1. Connect standard Ethernet Cable between ZedBoard Gigabit Ethernet Port and host PC.

2. Set up a wired network connection with the following properties:

   IP address : 192.168.1.1
   Subnet mask : 255.255.255.0
   Default gateway : leave blank
   This configures the host PC to a known IP address that will be used by the ZedBoard.

3. Turn on the Zedboard with configurations and connections shown.

4. Verify the default IP address of ZedBoard Ethernet (192.168.1.10) using `ifconfig` command. It is also possible to view the ZedBoard embedded webpage by opening a web browser and entering `http://192.168.1.10` as the URL.

5. To ping from the host PC to the ZedBoard, enter `ping 192.168.1.10` in the host terminal, and ensure that packets are transmitted and received.

6. To SSH via ethernet connection, enter `ssh root@192.168.1.10` from the host PC and use "root" as the password. The `zynq>` prompt should now appear on the terminal window.

3 Project Development: General Procedure

3.1 Extra Setup:

Install Vivado Design Suite for hardware design, and SDK for developing software applications.
Install Digilent Adept JTAG Drivers to enable programming via JTAG.
(Reference: `http://svenand.blogdrive.com/archive/172.html#.VbBTYEY1GSo`)

3.2 Step-by-Step

1. A hardware design is initiated in the Vivado software. Here, IP cores such as the Zynq processing system, BRAM (Block RAM), and DMA (Direct Memory Access) Controllers are instantiated, and the appropriate connections established. Clock settings, pin assignments, and address mapping are also performed in Vivado.

2. The design is **Validated** and saved.

3. An **HDL wrapper** is created for the complete block design, which provides a top-level VHDL or Verilog file that can be interpreted by the design sythesizer.

4. Next, **Synthesis**, **Implementation**, and **Bitstream Generation** all occur sequentially when selecting **Generate Bitsream**. Once the bitstream is generated, opening the **Hardware Manager** allows for the device to be programmed.

5. Once the device is programmed (indicated by the blue "Done" LED of the ZedBoard), the hardware design and bitstream can be exported to the Software Development Kit (SDK), and SDK launched for software development.

6. Before developing a software application, a **Board Support Package** is created, which establishes the necessary drivers and functions compatible with the device's operating system so that communication between the OS and the hardware can occur.

7. A software application is created and may be executed by a light or a highly comprehensive operating system, depending on the demands of the design.

This general structure was followed in completing the several tutorials discussed next. The tutorial references are thorough in describing each project, but additional remarks were noted as reminders of good points or takeaways from the tutorial.

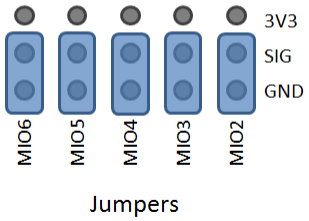
4 Hello World Application

Develop a simple hardware design for the Zynq PS and execute a C program to print "Hello World."

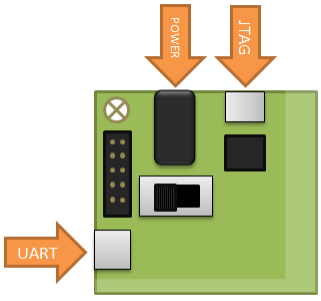
4.1 Relevant Tutorial(s):

Introduction to Zynq Hardware by Speedway Design Workshops, Labs 1-2

4.2 Configurations and Connections:



Jumpers



Connections

4.3 Observations and Remarks

1. This hardware design makes use only of the Zynq PS (processing system), not the PL (programmable logic).
2. The Hello World template is provided in SDK and may be modified for further experimenting with a simple PS design.

5 Enabling and Mapping PS Peripherals

Enable peripherals such as SPI Flash, Ethernet, USB, SD Card, and GPIO from the Zynq PS, allocate pins, and configure necessary clocks.

5.1 Relevant Tutorial(s):

Introduction to Zynq Hardware by Speedway Design Workshops, Lab 3

5.2 Observations and Remarks

1. This hardware design also makes use only of the Zynq PS (processing system), not the PL (programmable logic).
2. The hierarchy of mapping a variety of peripherals - SPI Flash, Ethernet, USB, SD Card, GPIO - based on the flexibility in MIO assignments is discussed.
3. Built-in test routines in SDK are executed to verify the memory addresses and peripheral connections.

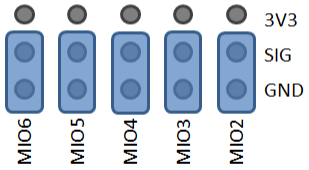
6 Using DMA for PL-PS Data Transfer - BRAM

Perform data transfer across memory on the PL (BRAM) and PS (DDR3), both with and without DMA (Direct Memory Access), to compare performance.

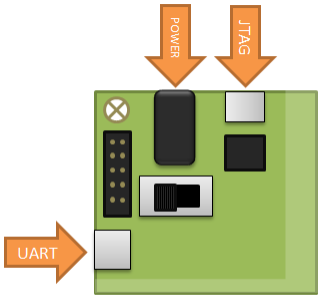
6.1 Relevant Tutorial(s):

Introduction to Zynq Hardware by Speedway Design Workshops, Labs 5-6

6.2 Configurations and Connections:



Jumpers



Connections

6.3 Observations and Remarks

About the DMA (Direct Memory Access) Controller [1]:
The DMA Controller is a hardware mechanism that alleviates the system processor of regulating data transfer between selected peripherals and memory components, which can increase the performance of the overall system. For data that is acquired asynchronously from a peripheral, the DMA issues an interrupt requesting access to the data bus from the processor so that the data may be transferred to the system memory. When the CPU acknowledges the request, a DMA buffer, or shared memory space, is allocated for receiving the incoming data. The data is written by the peripheral, and an interrupt signals when the task is complete.

The tutorial provides a C script that measures the time elapsed for data transfer (in clock cycles) when DMA is used and when it is inactive. This is useful in assessing the performance gains that can be achieved when performing data transfers within PL, within PS, or across PL and PS. The results obtained after implementing the design and running the test are presented in Table 1.

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.

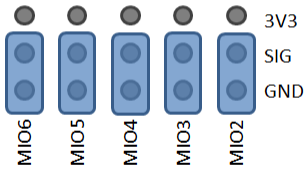
7 Using DMA for PL-PS Data Transfer - FIFO Loopback

Transfer data from DDR3 memory to a FIFO loopback on the Zynq PL, which returns the data to DDR3, through the use of DMA.

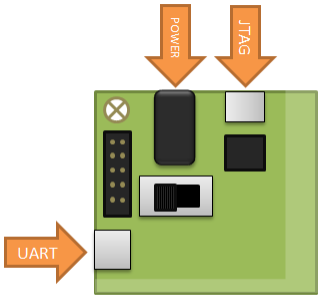
7.1 Relevant Tutorial(s):

Using the AXI DMA in Vivado - `http://www.fpgadeveloper.com/2014/08/using-the-axi-dma-in-vivado.html`

7.2 Configurations and Connections:



Jumpers



Connections

7.3 Observations and Remarks

1. Similarly to the previous design, this project incorporates a different IP core (FIFO Loopback) on the Zynq PL alongside the PS.
2. One future goal is to modify the provided C script to include the performance metric of the previous test, and extend the comparison of DMA vs. non-DMA data transfer to a range of different PL implementations as well.

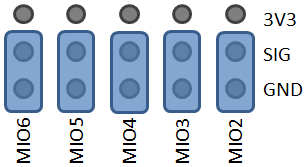
8 Switch Register - BRAM - DDR3 Data Transfer using DMA

Expand the data transfer paths explored by introducing a user input obtained through the ZedBoard switches, which is stored in BRAM and transferred to DDR3 memory using DMA.

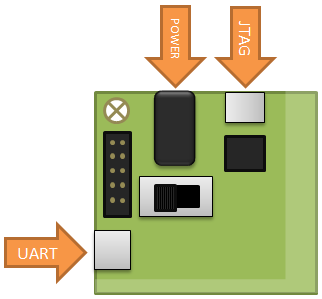
8.1 Relevant Guide(s):

Introduction to Zynq Hardware by Speedway Design Workshops, Labs 5-6
Adding Custom IP to the System by Xilinx -
`http://forums.xilinx.com/xlnx/attachments/xlnx/GenDis/16930/1/`
Writing Basic Software Application -
`http://forums.xilinx.com/xlnx/attachments/xlnx/GenDis/16930/2/adding%20ip%202.pdf`

8.2 Configurations and Connections:



Jumpers



Connections

8.3 Observations and Remarks

1. This was our first attempt at a custom design, which incorporated components from a few different existing procedures. The basic sequence is as follows:
   - The user selects an 8-bit value using the external switches on the ZedBoard.
   - The switch input, stored in a register, is transferred to Block RAM on the PL.
   - The BRAM data is transferred to DDR3, both with and without using DMA.
   - The value stored in the destination address is printed to the terminal to verify that it matches the switch input.
2. To introduce switches to a Vivado hardware design, a GPIO (General Purpose Input-Output) IP core is selected, and its input port is made **external** to indicate that an external component such as the switches will be used.
3. When composing the main C code in SDK that will execute a desired function on the hardware, it is a good idea to **"Clean Project"** after editing the code, which discards previous build problems and rebuilds the project. Additionally, executing `xil_printf("%c[2J",27);` within the main C code will clear the screen of the terminal each time the code is run.

References

[1] Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. *Linux device drivers*. " O'Reilly Media, Inc.", 2005.