

Remote Lab Demonstration

Tutorial

November 2010

Revision 002

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Intel, the Intel Atom family of processors, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2010, Intel Corporation. All rights reserved.

Copyright © 2010, Gleichmann Electronics Research. All rights reserved.

Contents

1	Introduction	6
1.1	Goals	6
1.2	Referenced Documents	6
2	System Overview	7
2.1	Block Diagram	7
2.2	Configurations	7
2.3	Connectors.....	10
2.4	Inside the FPGA	12
3	Starting Up	14
3.1	Connect Via SSH	14
3.2	Create Hello World! Message	16
3.3	Command: lspci	17
3.4	Command: lsfpfga.....	17
3.5	Command lsmod	17
3.6	Creating a Running Light	18
3.7	Data Exchange	19
4	Hpe® Desk	21
4.1	Loading a Design	21
4.2	Creating a New Design.....	21
4.3	Clock Factory	25
4.3.1	Clock Distribution.....	25
4.3.2	Board Configuration.....	27
4.3.3	FPGA Configuration	28
4.4	Creating an FPGA Configuration File.....	28
4.5	Running the New FPGA Configuration.....	29
5	Summary	33

Figures

Figure 1.	Hpe® IRP Panel – Front.....	10
Figure 2.	Hpe® IRP Panel – Back	10
Figure 3.	Hpe® IRP Main Board – Top.....	11
Figure 4.	Hpe® IRP Main Board – Bottom.....	12
Figure 5.	FPGA Configuration.....	13
Figure 6.	Web Cam View of the Hpe® IRP	15
Figure 7.	PuTTY Configuration.....	16
Figure 8.	Print a Hello World! Message on the LCD.....	16
Figure 9.	Examine the Contents of the FPGA With lsfpfga.....	17
Figure 10.	Examine the Loaded Drivers	18
Figure 11.	Complete Script for Creating a Running Light	19
Figure 12.	Mounting the IRP User's Home Directory on a Windows Machine	20

Figure 13. Opening Demo1 in Hpe® Desk 21
Figure 14. Hpe® IRP Clock Factory Configuration 26
Figure 15. Configuring the Clock Distribution in Hpe® Desk 27
Figure 16. FPGA Configuration with UART IP Core 29
Figure 17. Driver for the UART IP Core 30
Figure 18. Configuring the Driver for the UART IP Core..... 31

Tables

Table 1. Hpe_IRP Firmware 8
Table 2. Software Packages installed on the Hpe_IRP 8
Table 3. Windows software 9
Table 4. IP Cores 9
Table 5. FPGA Design Components 13

Revision History

Document Number	Revision Number	Description	Revision Date
451161	1.0	Initial release.	June 2010
324614	002	Update to Section 2.1	November 2010

§

1 Introduction

1.1 Goals

After this demonstration/lesson you will understand:

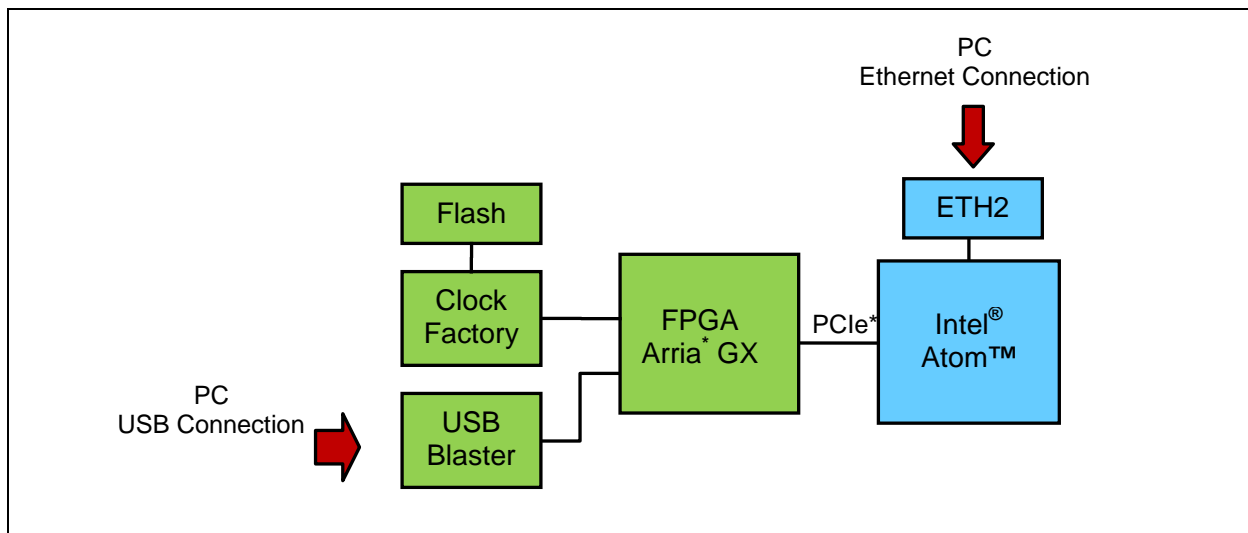
- The hardware architecture of the Hpe[®] Industrial Reference Platform (IRP).
- The Linux* environment of the Hpe[®] IRP.
- How to create your own Field Programmable Gate Array (FPGA) design.

1.2 Referenced Documents

Document	Document Number/Location
<i>Intel[®] Remote Lab Access Guide</i>	Document will be emailed to individual customers.
<i>Hpe[®] Industrial Reference Platform (IRP) User Manual</i>	http://www.ge-research.com/attach/UserManual-Hpe_IRP.pdf
<i>Hpe[®] desk Basic Manual</i>	http://www.ge-research.com/attach/UserManual_HpeDesk_basic.pdf

2 System Overview

2.1 Block Diagram



The Hpe® IRP consists of:

- Intel® Atom™ processor on a Qseven* module
- Intel® Atom™ Z530 processor (up to 1.6 GHz)
- Intel® System Controller Hub US15W chipset with integrated Intel® Graphics Media Accelerator (Intel® GMA) 500
- 1 GB DDR2 SDRAM (soldered)
- Hard disk 250GB SATA
- Gentoo Linux* OS
- OSADL real-time kernel
- Altera* Arria GX FPGA:
 - Programmable hardware
 - Contains all the IP cores
 - Connected with the Intel® Atom™ processor via PCIe*
 - Programmable with an onboard USB Blaster or automatically from a Flash on system start-up.
 - Clocks are provided from a configurable Clock Factory.

2.2 Configurations

The configurations for the testing conducted by Gleichmann Electronic Research GmbH are listed in subsequent tables.

Table 1. Hpe_IRP Firmware

Firmware	Version
BIOS	X1.00n
EC SW	1.1

Table 2. Software Packages installed on the Hpe_IRP

Category	Package	Version
app-examples	atem-demo	1.2
app-examples	kpa-master-demo	1.4.7.0-r1
app-examples	uio-python-base	1.0
app-examples	can-utils	0.8
app-examples	dma-demo	1.0
app-examples	gpio-demo	1.0
app-examples	hmi-demo	1.3
app-examples	hmi-eth-demo	1.1
app-examples	i2c-demo	1.1
app-examples	interrupt-latency-demo	0.1
app-examples	sercos-III-master-demo	1.0
app-examples	throughput-demo	0.2
app-examples	webserver-demo	0.5-r2
app-examples	ahb-ram-test	0.1
app-examples	uart-demo	1.0
sys-apps	installkernel	0.9
sys-apps	irpselftest	0.4
sys-apps	button-watch	0.8
sys-apps	fpgautils	0.7
sys-apps	pciutils	3.1.4
sys-devel	gcc	4.1.2
sys-drivers	pcie-base	1.6-r1
sys-drivers	ahb-histogram-uio	1.0
sys-drivers	ahb-ram-uio	1.5
sys-drivers	ahb-trace-uio	1.4
sys-drivers	can	1.4
sys-drivers	com-crc32-uio	0.9
sys-drivers	dma	0.5-r1
sys-drivers	gpio-uio	1.5
sys-drivers	hmi	1.4

Category	Package	Version
sys-drivers	i2c	1.5
sys-drivers	mtip100	1.6
sys-drivers	sercos3	1.4
sys-drivers	timer-counter-uio	1.5
sys-drivers	uart	1.6
sys-libs	glibc	2.6.1
sys-kernel	osadl-sources	2.6.31.12

Table 3. Windows software

Name	Version
Hpe_desk basic	v3.1pretest13
Quartus	9.1SP2 Web Edition

Table 4. IP Cores

Name	Version
AHB_ARBITER	v1/20090930
OSS_ROM	v1/20081028
AHB-PCIe	v1/20090420
AHB-APB	v1/20081028
AHB_DMA	v1/20090930
AHBRAM	v1/20080924
ZBTSRAM	v1/20080924
CAN_GER	v1/20090415
10/100ETHMAC	1.2/20090526
Sercos III	v1/20090713
HMI-IRP	v1/20090608
I2C-Asics	v1/20090608
UART	v1/20090703
GPIO	v1/20080917
Timer	v1/20090529
AHBHistogram	v1/20100622
AHBTrace	v1/20090923
Com CRC32	v1/20100719

2.3 Connectors

The Hpe® IRP offers a lot of connectors, most of them are on the back panel. For additional or custom connectors, there are two internal high speed (up to 5 Gb/s) connectors for child boards.

Figure 1. Hpe® IRP Panel – Front

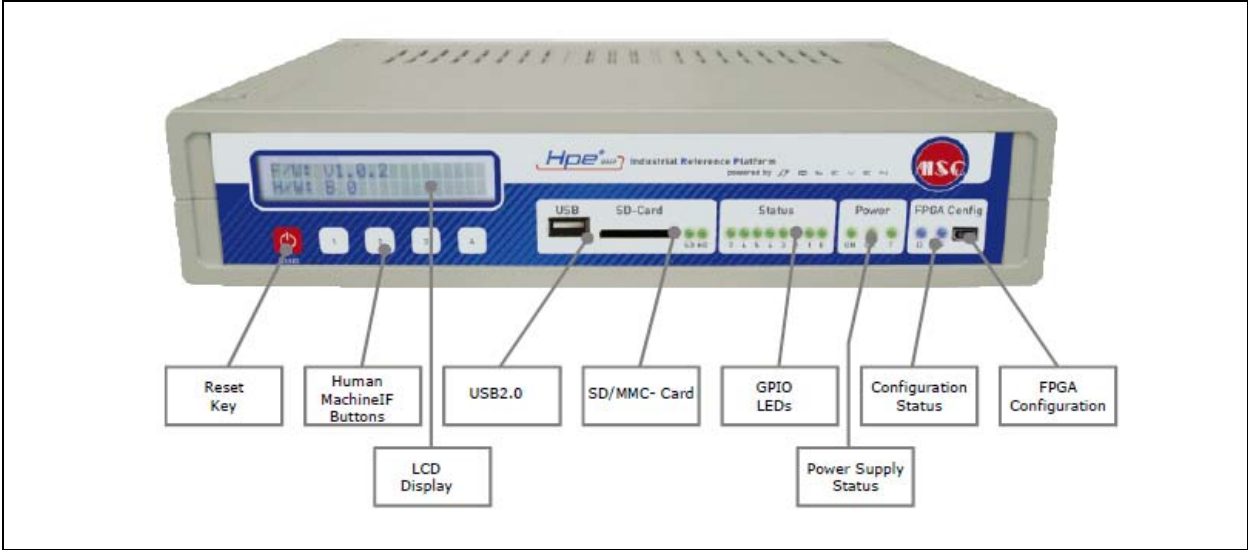


Figure 2. Hpe® IRP Panel – Back

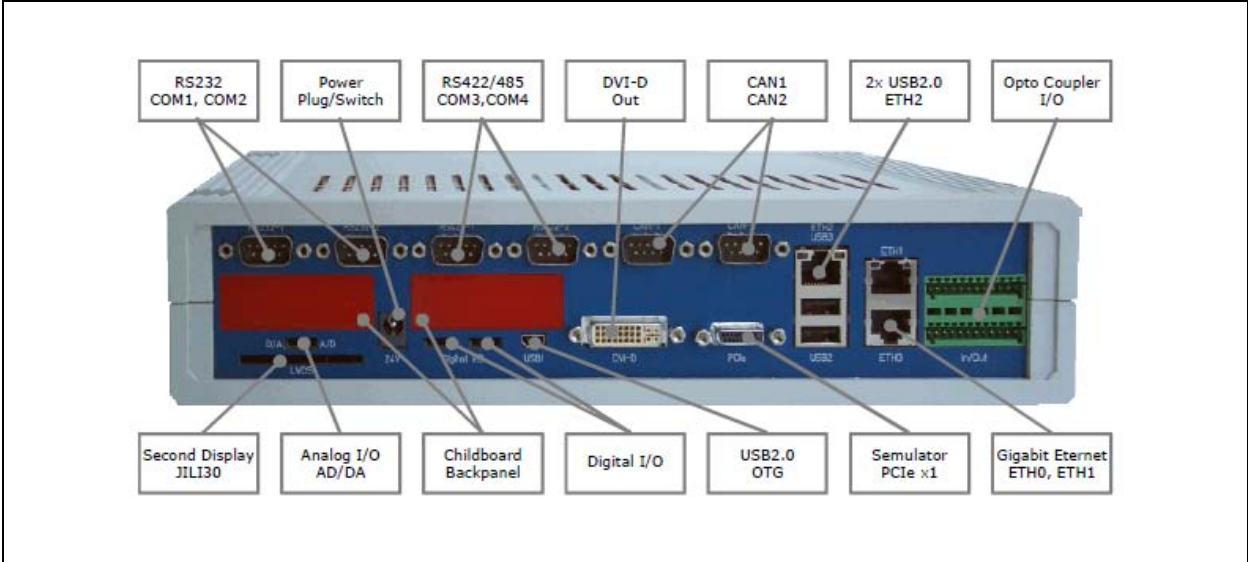


Figure 3. Hpe® IRP Main Board – Top

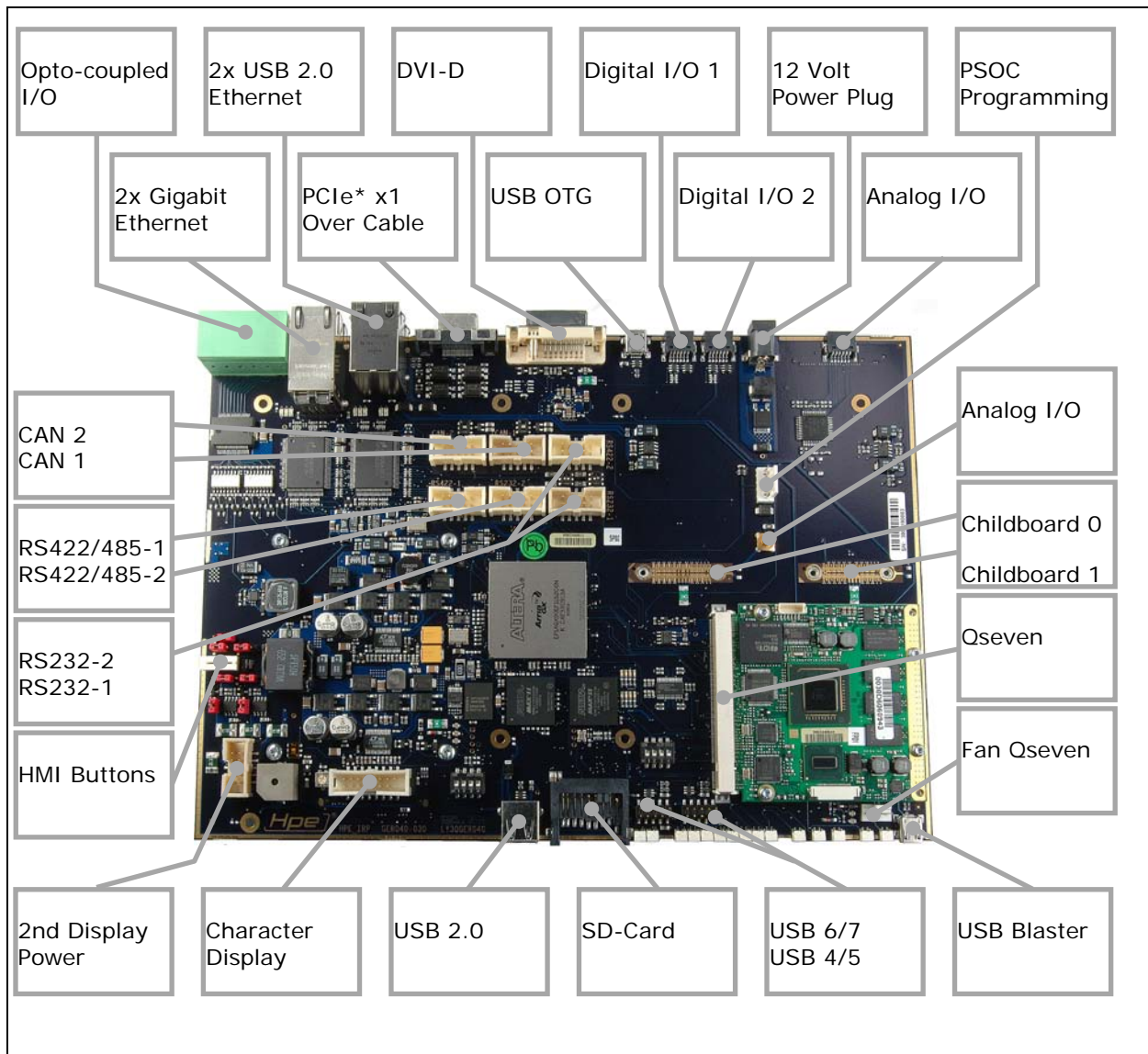
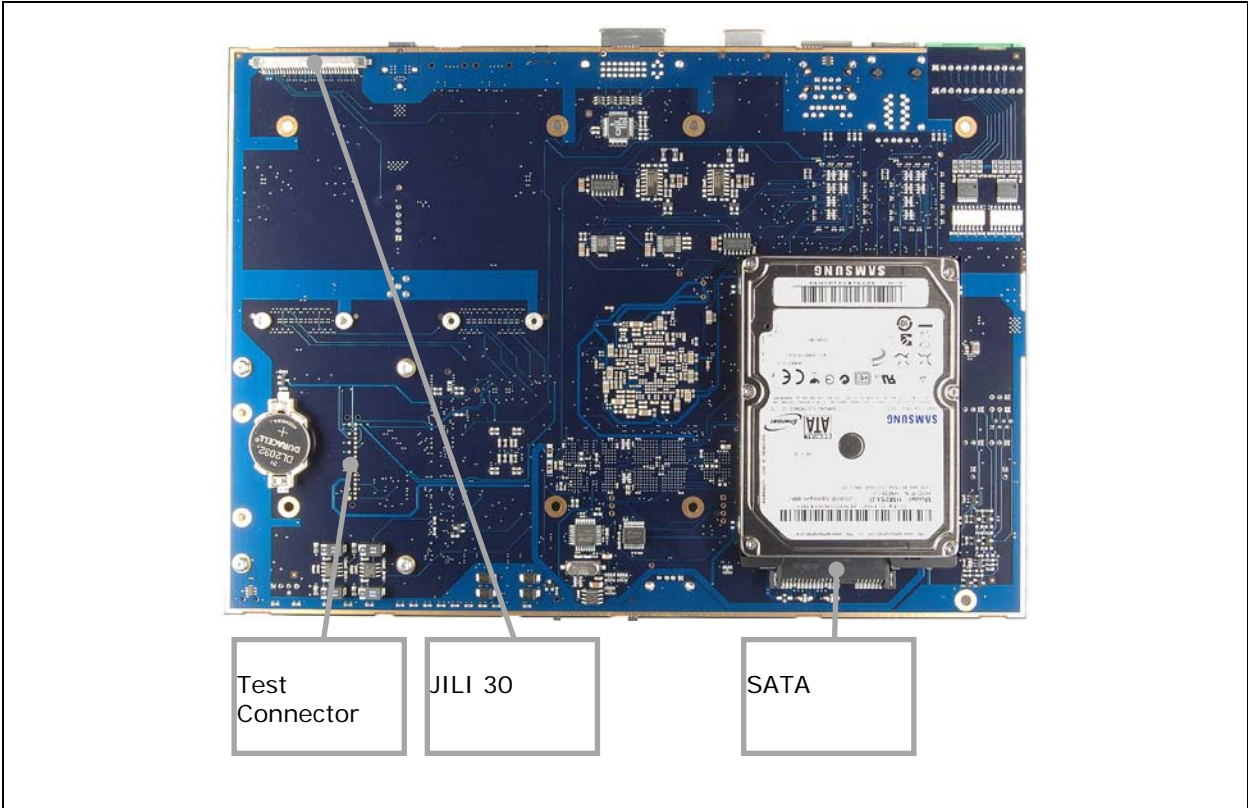


Figure 4. Hpe® IRP Main Board – Bottom

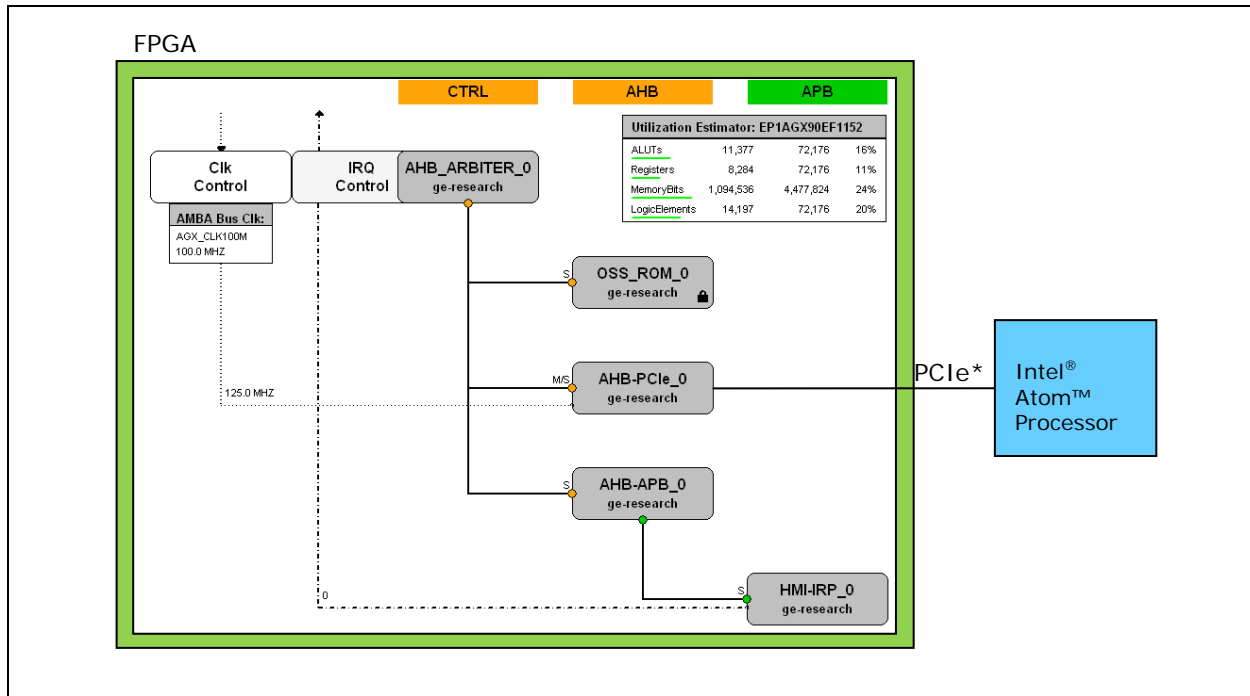


For more detailed information refer to the manual; the shortcut "IRP Samba Server" on the desktop of Virtual Lab's Windows* machine contains "Hpe_IRP_support\Manual\manual-1.0.3.pdf".

2.4 Inside the FPGA

Figure 5 shows the easiest configuration of the FPGA. This configuration is used in the delivery status of the Hpe® IRP and in the beginning of this tutorial. Later on we will use the Hpe_desk software to add further IP cores to the design.

Figure 5. FPGA Configuration



The IP cores inside the FPGA are connected by the Advanced High-performance Bus (AMBA AHB) and Advanced Peripheral Bus (AMBA APB). The IP cores shown in [Figure 5](#) are explained in [Table 1](#).

Table 5. FPGA Design Components

Component	Function
AHB Arbiter	Controls the AHB.
AHB-PCIe* bridge	Connects the AHB the Intel® Atom™ processor via PCIe.
OSS ROM	Stores the address ranges of the IP Cores, is needed to load the correct drivers.
AHB-APB bridge	Connects the AHB with the slower APB for peripherals.
HMI IRP	Human machine interface controls the character - LCD, LEDs, keys on the front and internal DIPs.

3 Starting Up

There are two ways to log into the IRP:

1. With a USB keyboard and a DVI-D display or
2. Remotely via a SSH network connection.

3.1 Connect Via SSH

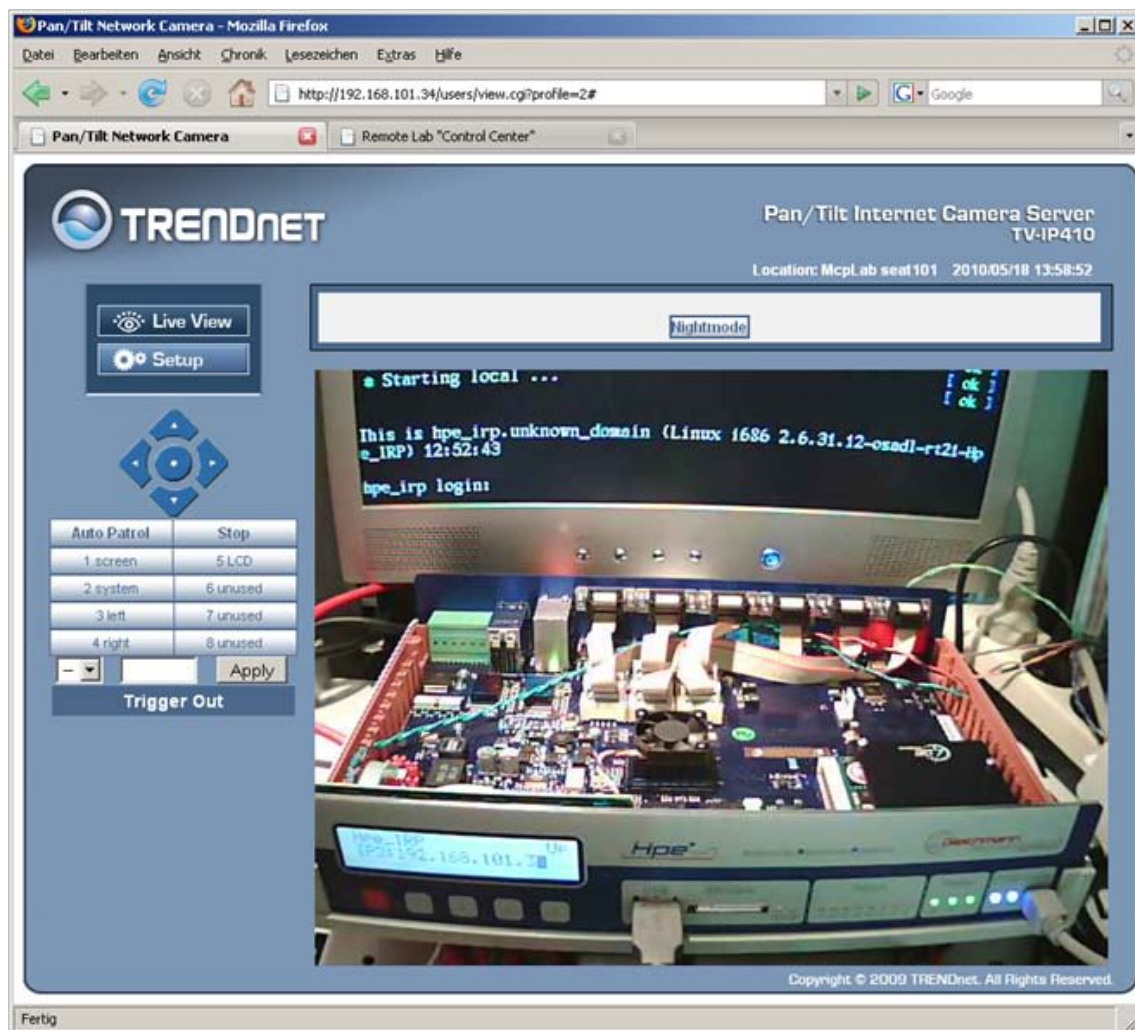
In the Virtual Lab we will use a Secure Shell (SSH) connection. The Internet Protocol (IP) address of the IRP is stated in the login credentials you received during the Virtual Lab registration process. In the rest of this document we assume the IP address 192.168.101.3.

Step 1

Establish a connection to the Virtual Lab as described in [TODO]. The result should look like [Figure 6](#). Note that you can also find out the IRP's IP address by looking at the LCD or at the screen. You can use the preset positions of the webcam for this.

The two blue LEDs in the FPGA Config section of the front panel indicate that the D (Device under Test) in the FPGA and the CF (clock factory) are successfully configured.

Figure 6. Web Cam View of the Hpe® IRP

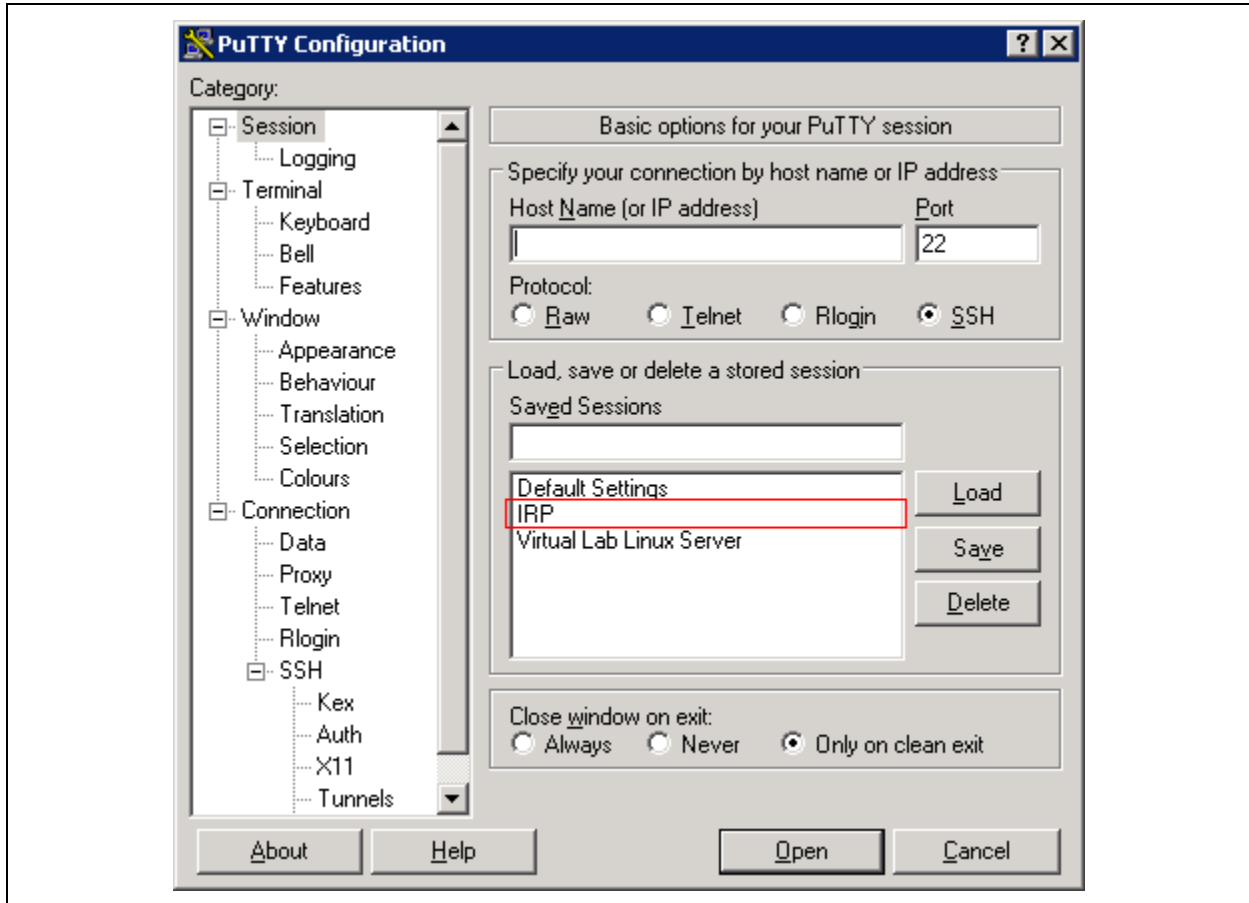


Step 2

1. Start up PuTTY (there is a shortcut on your desktop).
2. Type the IP address in the Host Name field (Figure 7), using the IP address from the LCD display (see Figure 6).
3. Name the session.
4. Save the settings.
5. Open a connection and login using the following:

Login user: irp
password: irp

Figure 7. PuTTY Configuration



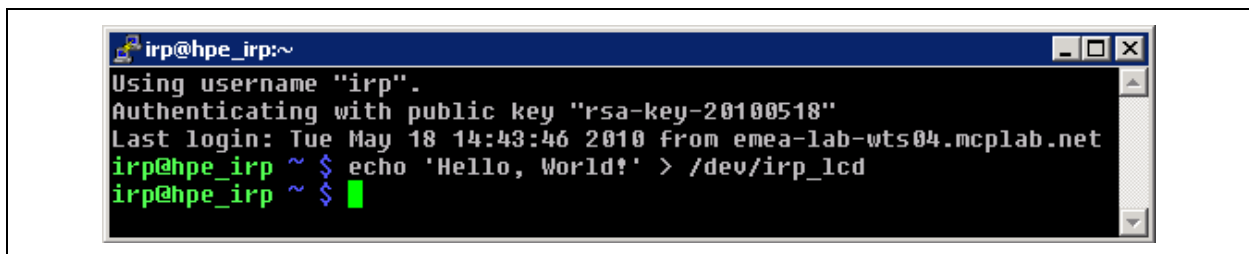
3.2 Create Hello World! Message

First let us try to output a "Hello, World!" message on the LCD character display on the front panel of the IRP. The command to do so is shown in [Figure 8](#).

Note: You must enclose the message in single quotes. Otherwise the shell (bash in this case) would interpret the exclamation mark and try to do a history substitution.

When you look at the web cam you will see the output.

Figure 8. Print a Hello World! Message on the LCD



3.3 Command: lspci

But how does the system work and what devices are available? With the command 'lspci' we get a list of all available PCI devices.

```
"/usr/sbin/lspci"
```

In this list we can find

```
"02:00.0 Class ff00: MSC Vertriebs GmbH Device 0004 (rev 01)"
```

This is our FPGA.

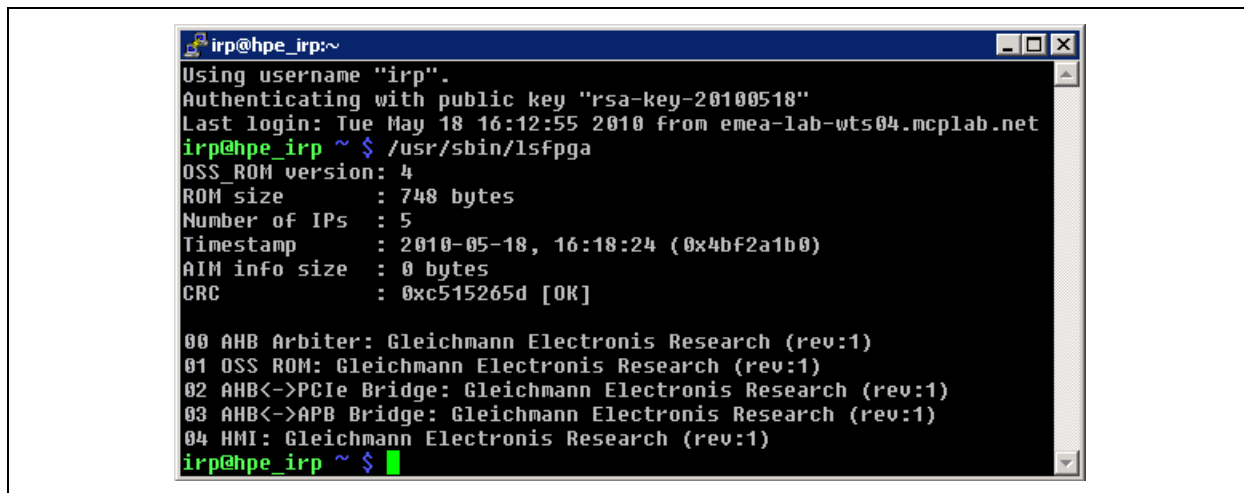
3.4 Command: lsfpfga

With the command 'lsfpfga' we can look into the FPGA to see its current configuration (see [Figure 9](#)). The option "-v" gives us a more detailed list.

```
/usr/sbin/lsfpfga  
/usr/sbin/lsfpfga -v
```

In the "Hello, World!" example above we used the human machine interface (HMI) IP core (number 04 in [Figure 9](#)). The HMI IP core drives the LCD and the LEDs and reads the four buttons on the front panel.

Figure 9. Examine the Contents of the FPGA With lsfpfga



3.5 Command lsmod

With the command "lsmod" we get a list of currently loaded kernel modules (drivers) as shown in [Figure 10](#). The module "hpe_irp_hmi" is the driver for HMI IP core. This driver provides the special file "/dev/irp_lcd" which we used to output the "Hello, World!" message.

Figure 10. Examine the Loaded Drivers

```

irp@hpe_irp:~$ lsmod
Module                  Size  Used by
hpe_irp_hmi             6788  0
pcspkr                  2524  0
sdhci_pci               7548  0
sdhci                   18592  1 sdhci_pci
e1000e                  110028  0
hpe_irp_bridge          10012  0
hpe_irp_bus             4380  2 hpe_irp_hmi,hpe_irp_bridge

```

3.6 Creating a Running Light

For a first project we will create a simple running light by using a shell script.

Step 1

1. Create a new folder in your work directory: `mkdir led1`
2. Change to this directory: `cd led1`
3. Create a new file named `led1.sh`: `touch led1.sh`
4. Now edit this file with the nano editor: `nano led1.sh`

Step 2

Write the script:

1. Always start a bash shell script with the comment line (also known as Shebang):

```
#!/bin/bash
```

2. Write some comment explaining the script.
3. Then we need a variable "leds" pointing to the file standing for the LEDs.

```
leds=`echo -n
/sys/bus/hpe_irp/drivers/hpe_irp_hmi/00*/leds`
```

4. Check to see if this file exists:

```
if [[ ! -e ${leds} ]] ; then
    echo 'ERROR: LEDs not available (Driver not
    loaded?)'
    exit 1
fi
```

5. Writing the values 0x01, 0x02, 0x04, ... to `${leds}`

```
for ((i=0;i<8;++i)) ; do
    printf 0x%x $((1<<i)) > ${leds}
    sleep 1
done
```

6. Write the final output:

```
echo Done
```

7. Click **CTRL +O** to save the file.

8. Click **CTRL + X** to close the editor.

The script should like [Figure 11](#):

Figure 11. Complete Script for Creating a Running Light

```
#!/bin/bash
#
# Shell script to demonstrate control the LEDs on the Hpe_IRP

leds=`echo -n /sys/bus/hpe_irp/drivers/hpe_irp_hmi/00*/leds`

if [[ ! -e ${leds} ]] ; then
    echo 'ERROR: LEDs not available (Driver not loaded?)'
    exit 1
fi

echo "Running 1 pattern on LEDs..."

# Simply write the hex numbers 0x1 0x2 0x4 ... to ${leds}

for ((i=0;i<8;++i)) ; do
    printf 0x%x $((1<<i)) > ${leds}
    sleep 1
done

echo Done
```

Step 3

To make the script an executable file you must change the permission of the script file:

```
chmod u+x led1.sh
```

Step 4

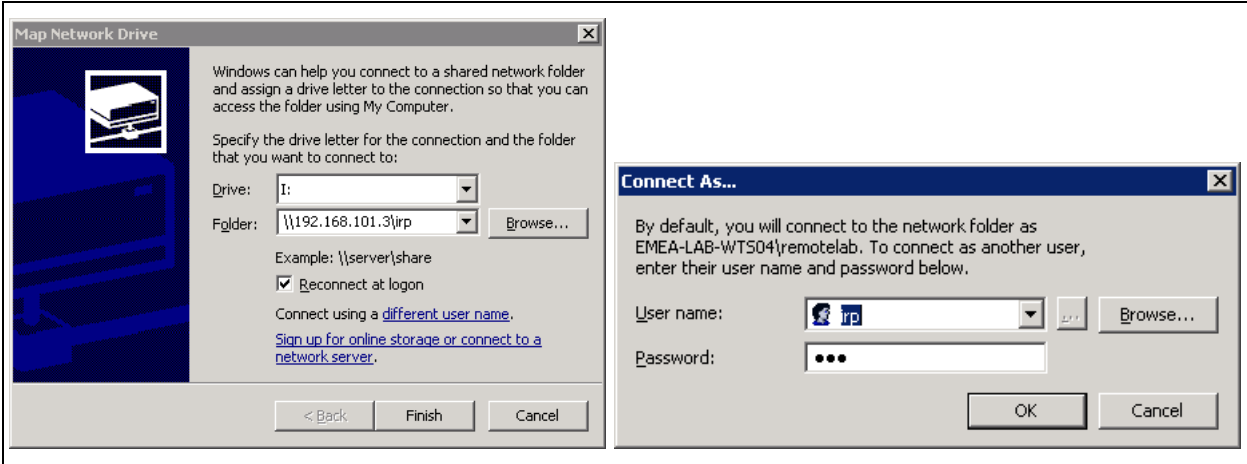
Now you can run the script **led1.sh**.

If your script does not work you can find a final draft under `"/home/irp/work/drafts/led1/led1.sh"`.

3.7 Data Exchange

For easier data modification and data exchange you can mount a network drive on your computer. Use the IP address shown on the IRP LCD after booting or use the command `ifconfig` to find out the actual IP address of the IRP.

Figure 12. Mounting the IRP User's Home Directory on a Windows Machine



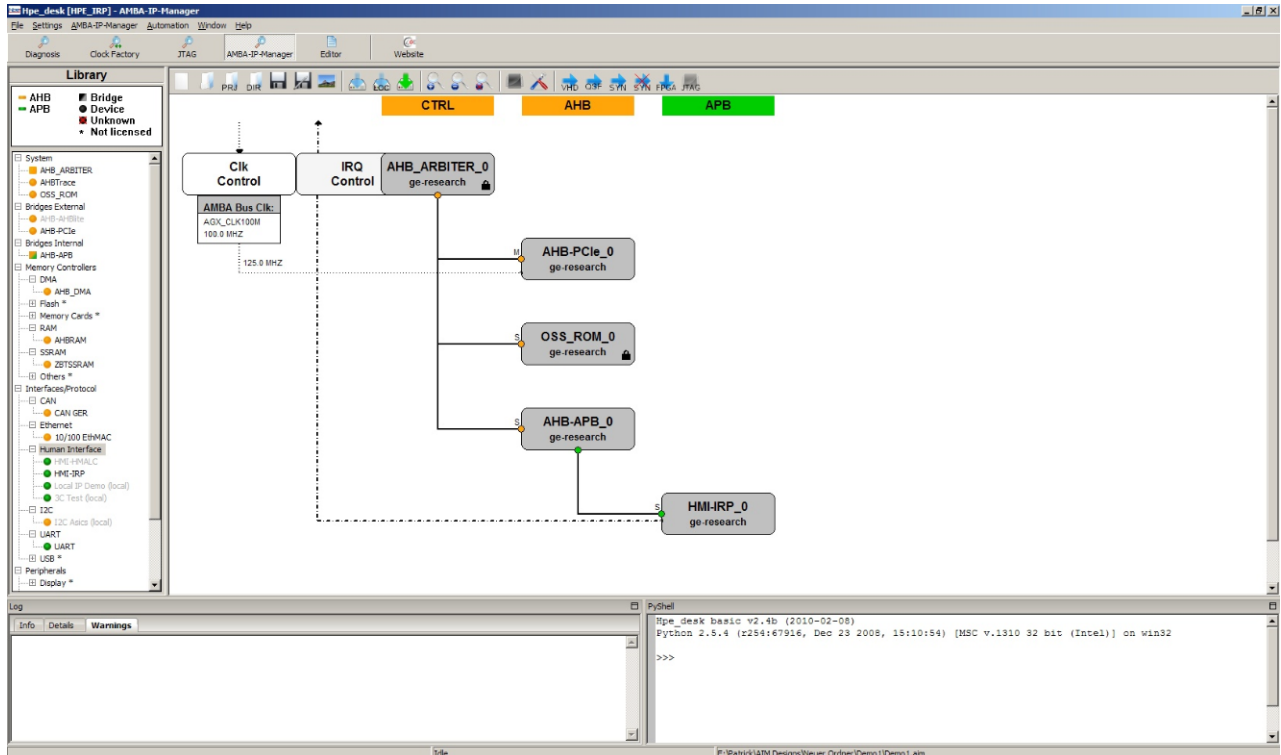
4 Hpe® Desk

Now we will discuss how to create FPGA designs using the Hpe® Desk software. More specifically we will use the AMBA-IP-Manager which is part of the Hpe® Desk.

4.1 Loading a Design

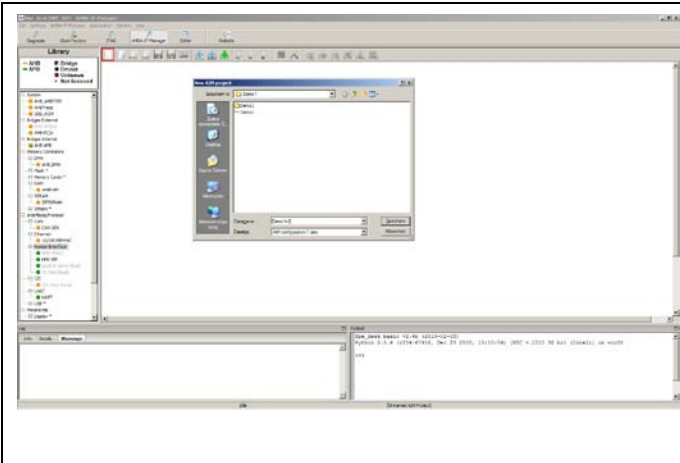
Use the desktop shortcut to locate the file "AIM-Designs\hmi-demo\hmi-demo.aim". Double-click this file and Hpe® Desk will start. Now you should see a design as shown in [Figure 13](#). This is the design we used in the "Hello World!" and running light projects.

Figure 13. Opening Demo1 in Hpe® Desk



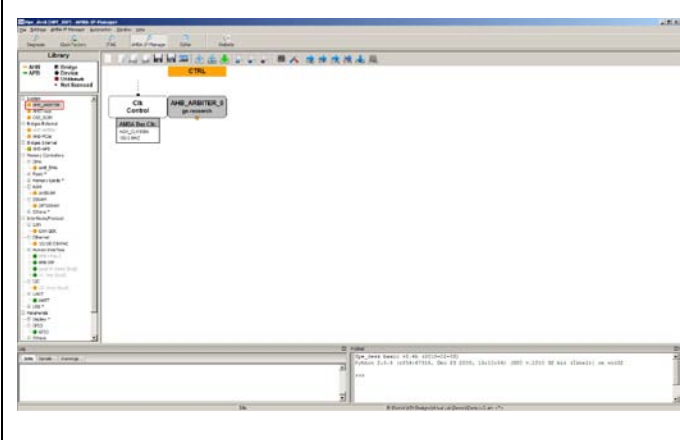
4.2 Creating a New Design

Now we want to add a Universal Asynchronous Receiver Transmitter (UART) to the design. Therefore, we will start with an empty design to learn more about the needed components:



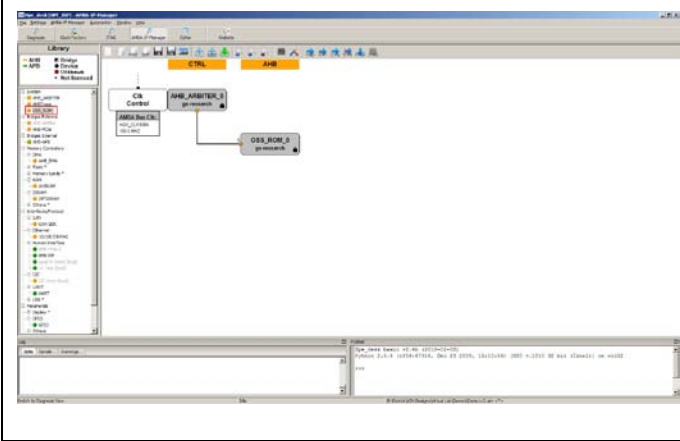
Step 1

Create a new design and save it as Demo1v2 in the same folder.



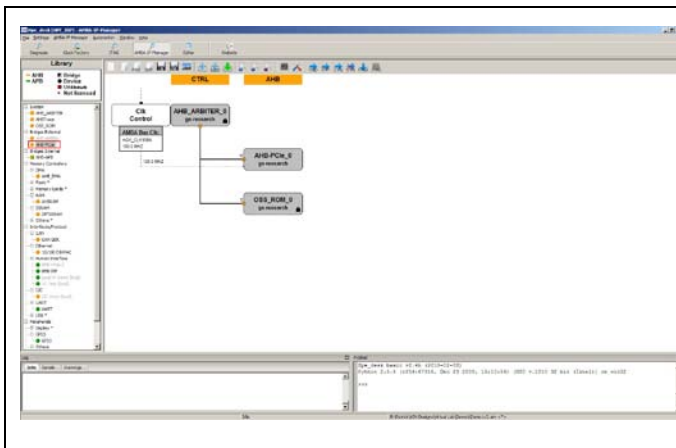
Step 2

We need an arbiter for the =Advanced High-performance Bus (AHB). The arbiter schedules the accesses to the bus and controls the multiplexors for the address and data signals of the AHB.



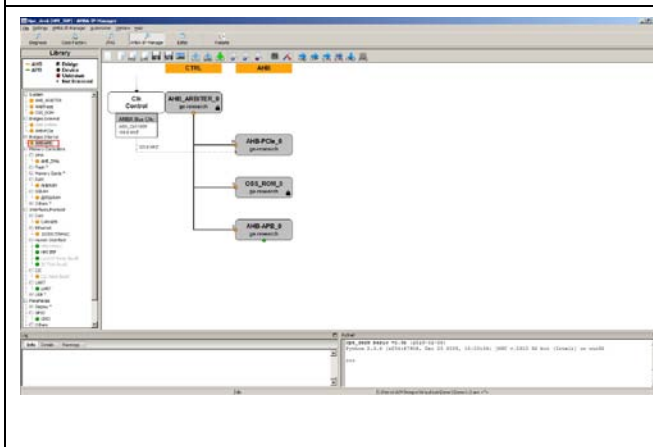
Step 3

We always need an OSS ROM. In the OSS ROM the address range and other important information from all used IP cores is stored automatically. This information is needed to load and configure the Linux drivers automatically. The content of the OSS ROM is generated by Hpe® Desk before the synthesis.



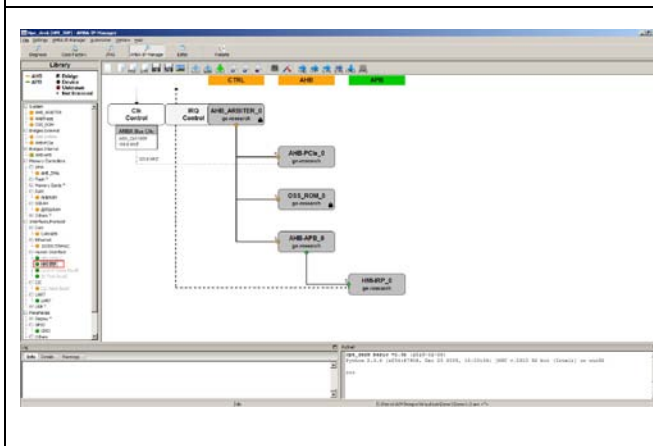
Step 4

The AHB-PCIe* bridge is the connection between the Intel® Atom™ processor and the AHB, which is the internally-used bus in the FPGA.



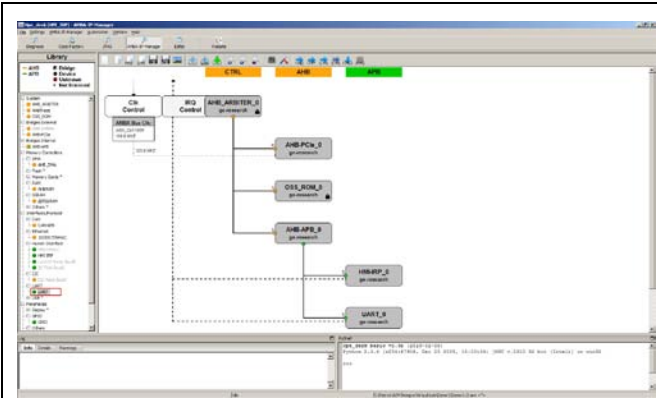
Step 5

Because the UART and HMI use the Advanced Peripheral Bus (APB) we need a bridge between the AHB and the APB. The APB is used to integrate low speed peripherals to the AHB. The APB offers a simpler interface, latched address and control and consumes less power.



Step 6

Again we integrated the HMI IP core to our design to have access to the LCD, the LEDs, and the keys.



Step 7

Last we integrate the UART. Hover the mouse over the inserted IP cores to show detailed information. By double-clicking the IP cores you can access IP details.

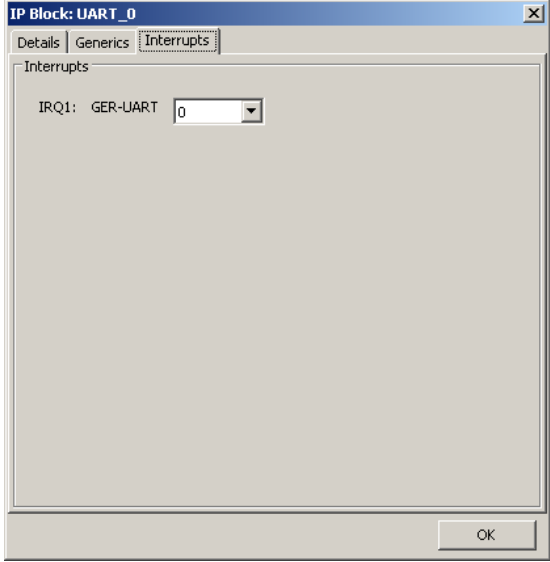
The screenshot shows a dialog box titled 'IP Block: UART_0'. It has three tabs: 'Details', 'Generics', and 'Interrupts'. The 'Details' tab is active and contains the following information:

- Summary**
Name: uart_ger_v1
Vendor: Gleichmann Electronics Research
Version: v1.1/20090703
- Description**
The GERA-UART (Gleichmann Electronic Research AMBA UART) is a macro, which is functional compatible to the NS16550A UART, except for the fact that it supports fifo mode only.
- Addressing**
0x02100000..0x021FFFFFF
- Pin Assignment**
Preset: COM1 [dropdown menu] [Edit Pin Assignment button]
- Resource Usage (approx. based on defaults)**
LogicElements: 533, ALUTs: 457, Registers: 310, MemoryBits: 128

An 'OK' button is located at the bottom right of the dialog box.

Step 8

Double-click on the UART IP core and set the Pin Assignment to COM1 by choosing COM1 from the drop down menu. Click **Edit Pin Assignment** to fully customize the Pin Assignment¹; however, it is strongly recommended to use the predefined assignments.

	<p>Step 9</p> <p>Select the Interrupts tab. Select Interrupt 0 from the drop-down box.</p>
---	---

NOTES:

1. The pin assignment defines the association of the top level ports with the physical FPGA pins.

4.3 Clock Factory

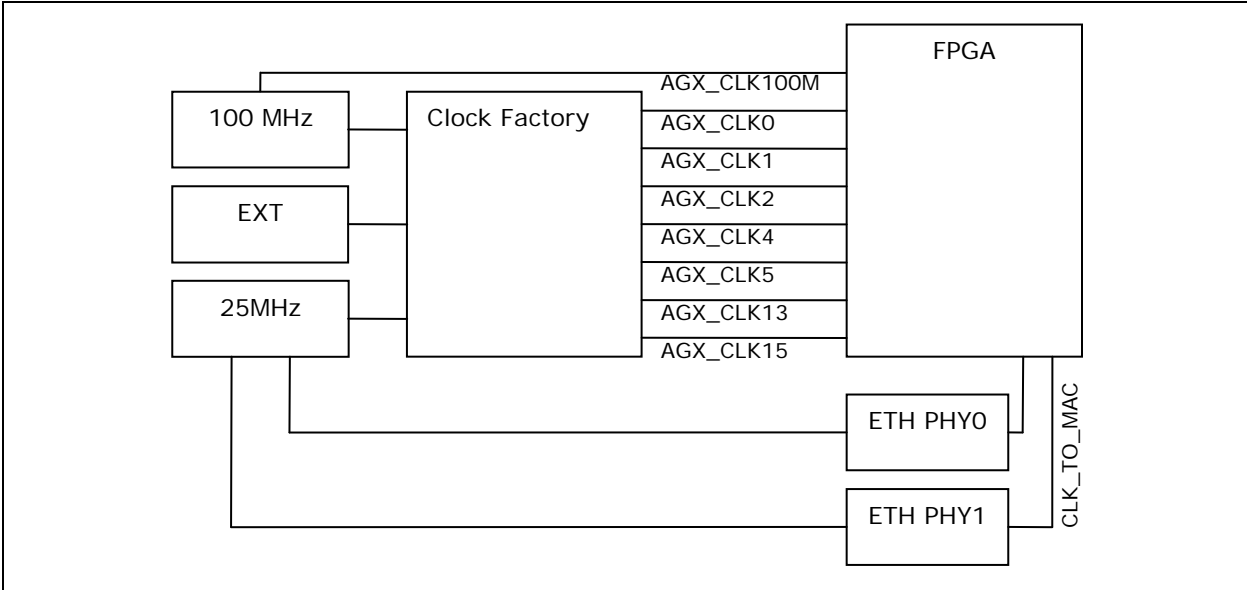
There is a dedicated CPLD (complex programmable logic device) called clock factory on the Hpe® IRP. This CPLD handles three tasks:

- Clock Distribution
- Board Configuration
- FPGA Configuration

4.3.1 Clock Distribution

On the Hpe® IRP a CPLD does the clock management. As you can see in [Figure 14](#) three clocks are delivered on the main board: 100MHz, 25MHz and one from an external connector.

Figure 14. Hpe® IRP Clock Factory Configuration

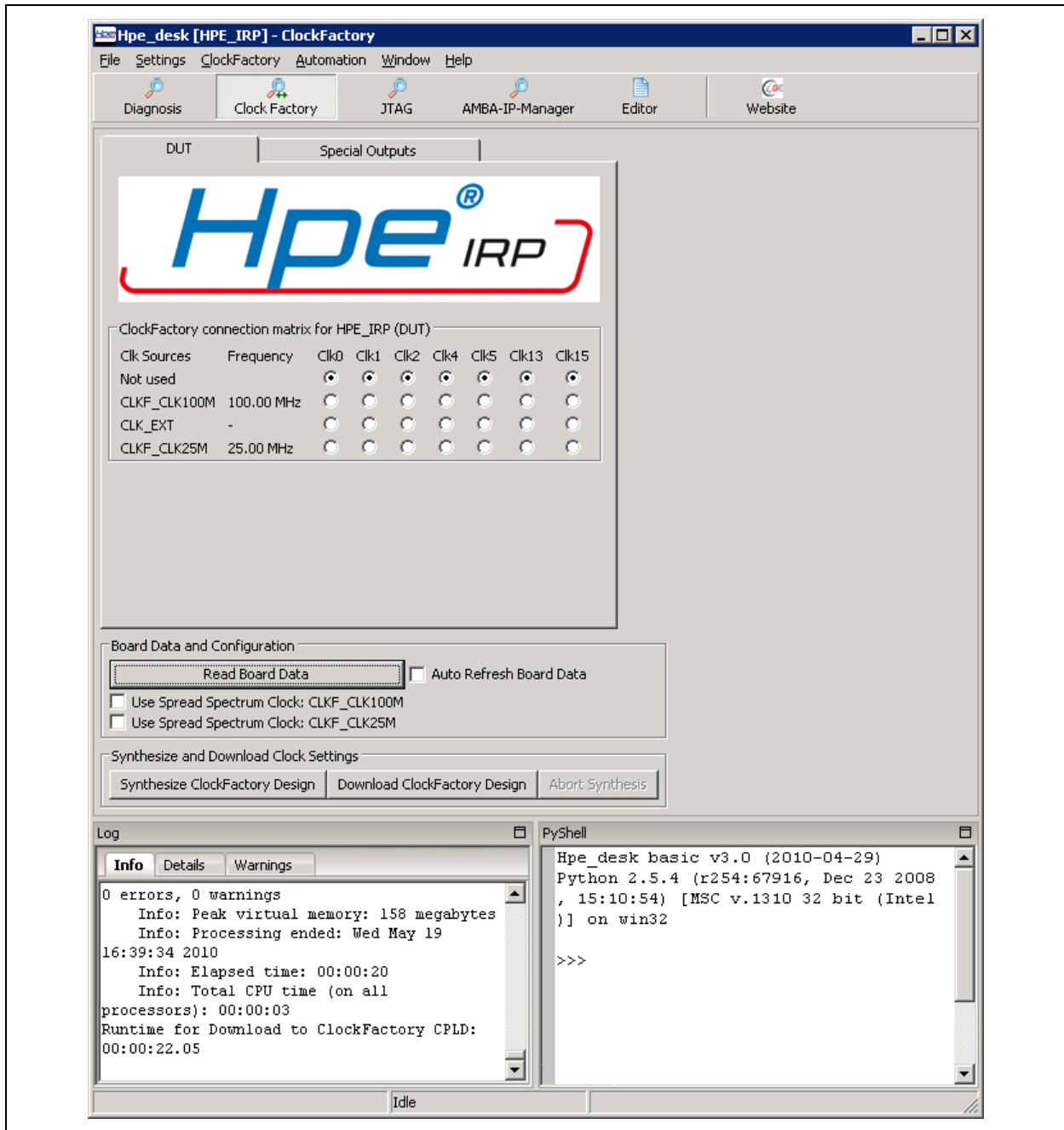


The clocks can be distributed to most of the clock pins of the FPGA. Use the Hpe® Desk to set the rules for how the clocks are distributed (see [Figure 14](#)) and to configure the clock distribution. The columns represent the FPGA clock input pins and the rows represent the clock sources. You can see the currently active configuration by clicking on "Read Board Data". Reading the board includes measuring the actual frequencies at the inputs of the clock factory. The results of this measurement are shown in the "Frequency" column.

You may also generate further clocks inside the FPGA by using the phase locked loops (PLL) in the FPGA.

In our design we do not need special clocking requirements and do not need to make any changes.

Figure 15. Configuring the Clock Distribution in Hpe® Desk



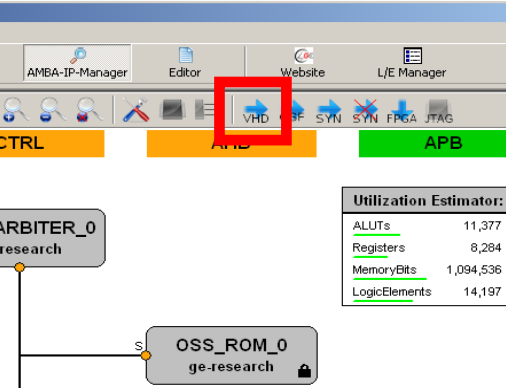
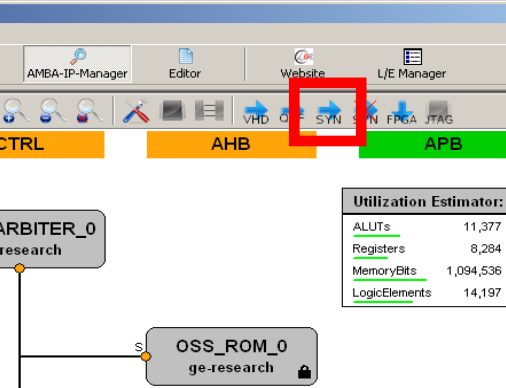
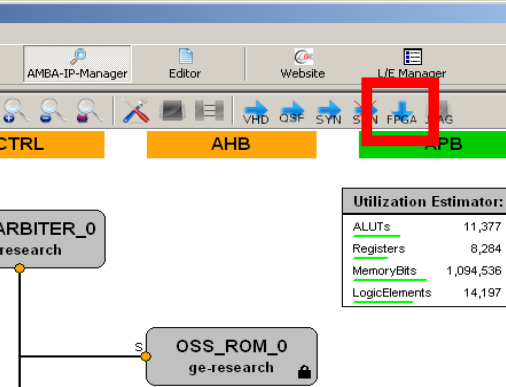
4.3.2 Board Configuration

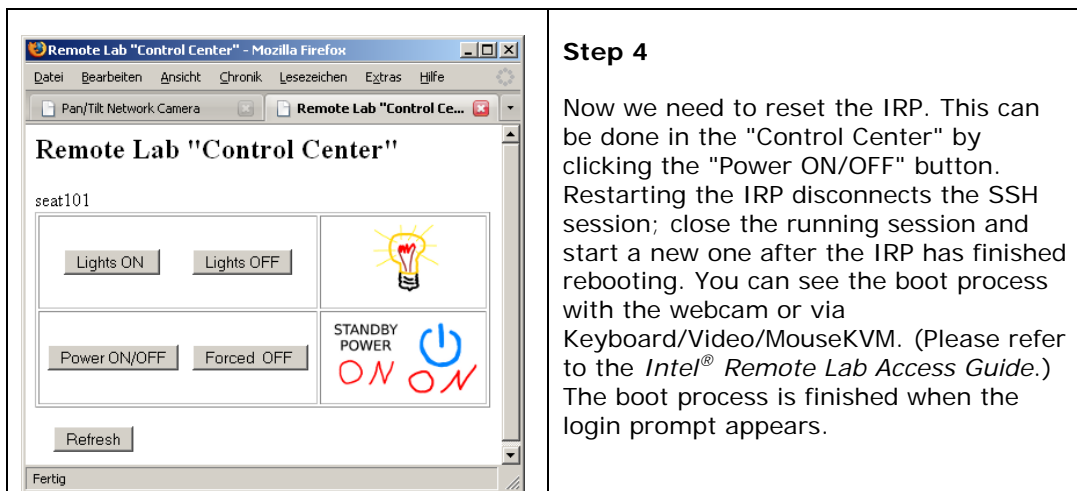
As you can see in [Figure 15](#) there is a second tab called "Special Outputs". In this tab you can configure the onboard Ethernet PHYs and details about the oscillators which provide the 25MHz and the 100MHz clocks. For this tutorial we can use the default settings.

4.3.3 FPGA Configuration

The clock factory CPLD configures the FPGA on every system start. It loads a configuration stored in an on-board FLASH memory into the FPGA.

4.4 Creating an FPGA Configuration File

	<p>Step 1</p> <p>Clicking on the VHD icon will generate a Top Level Entity and a new folder Demo1v2 in which the Top Level Entity can be found. If you want to see the generate VHDL code, open the file "toplevel.vhd" in the new folder.</p>
	<p>Step 2</p> <p>Save the project and start the synthesis by clicking on the SYN icon. During this process the HDL sources (Hardware Description Language) will be compiled, mapped to the available FPGA resources and finally place-and-route will be performed. The result of this is a file which can be downloaded to the FPGA. The whole process will take about half an hour.</p>
	<p>Step 3</p> <p>No we can download the design by clicking on the FPGA icon. If you carefully watch the webcam you will notice how the blue LED (the left one in Figure 1 and Figure 6) turns off during the reconfiguration.</p>



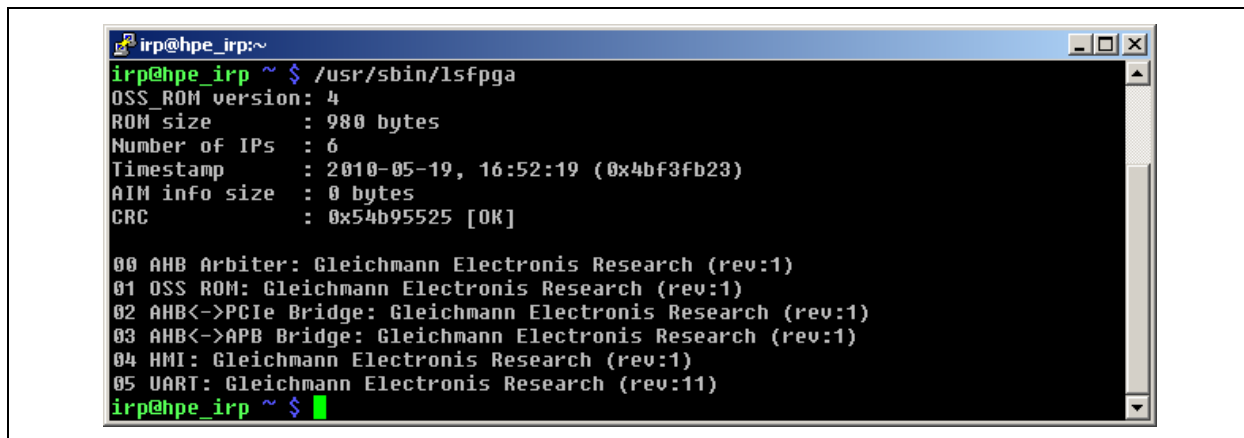
4.5 Running the New FPGA Configuration

Try to use the UART IP core. The RS232-1 interface of the Hpe IRP is connected to the COM2 of your working machine by a null modem cable, so you can send and receive data to/from the PC.

First, check if the new configuration is really active by using the "lsfpga" command. The output should contain the new UART IP core (see [Figure 16](#)).

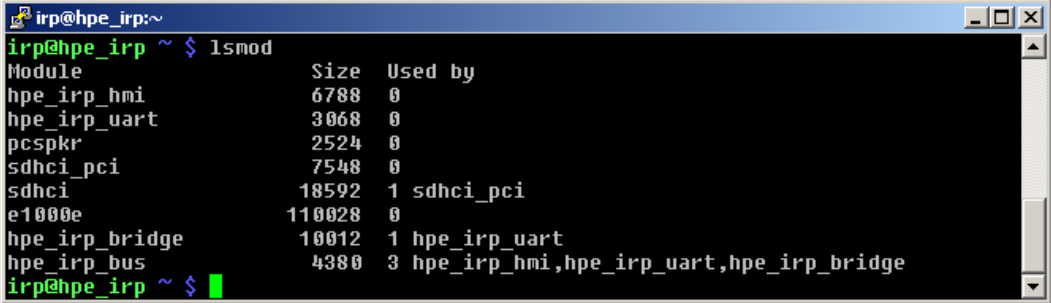
Clicking on the VHD icon (see [Section 4.4](#)) will generate a Top Level Entity and a new folder Demo1v2 in which the Top Level Entity can be found. If you want to see the generate VHDL code, open the file in the new folder.

Figure 16. FPGA Configuration with UART IP Core



Check if the proper driver is loaded by using the "lsmod" command. The output should contain the module "hpe_irp_uart" (see [Figure 17](#)). Note that the drivers are loaded on demand, i.e., only those drivers are loaded which are actually required by the active FPGA configuration.

Figure 17. Driver for the UART IP Core



```
irp@hpe_irp:~$ lsmod
Module                  Size  Used by
hpe_irp_hmi             6788  0
hpe_irp_uart           3068  0
pcspkr                 2524  0
sdhci_pci              7548  0
sdhci                  18592  1 sdhci_pci
e1000e                 110028 0
hpe_irp_bridge         10012  1 hpe_irp_uart
hpe_irp_bus            4380  3 hpe_irp_hmi,hpe_irp_uart,hpe_irp_bridge
irp@hpe_irp:~$
```

Go to the directory `cd /home/irp/tutorial1/solution/UartDemo`. Find t a file called `UartDemo.c` i. Open the file with nano ("`nano UartDemo.c`").

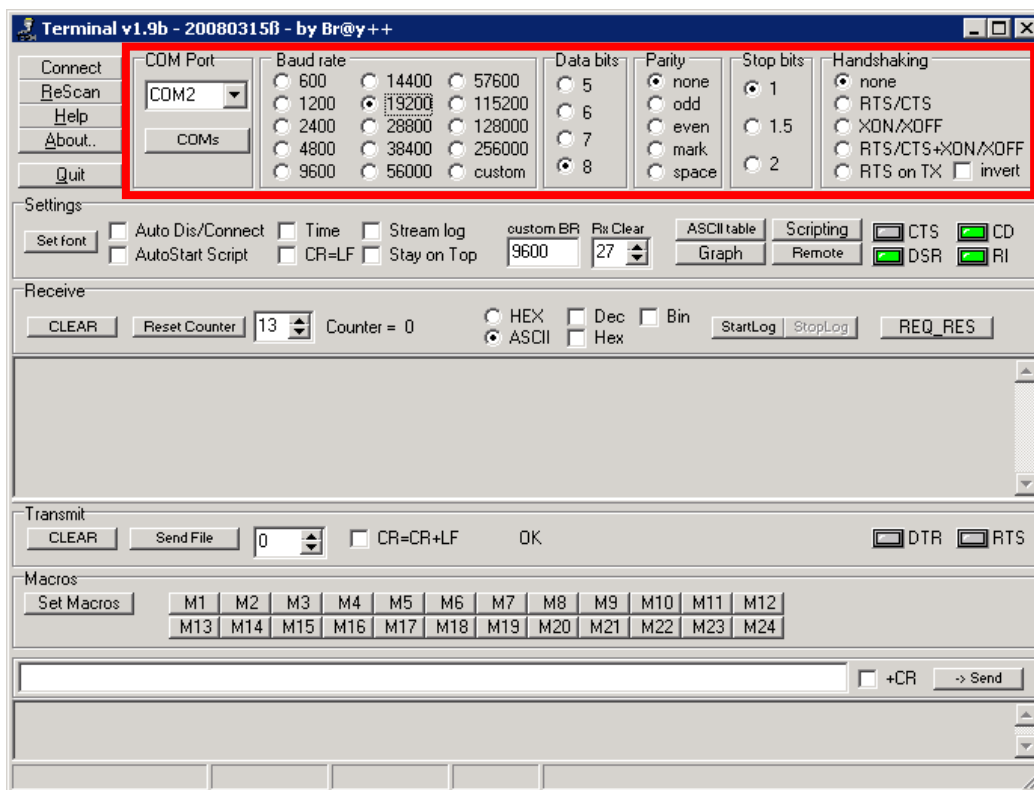
You can see the c-File by doing the following things:

1. Open a serial Port `ttyS0`.
2. Configure the Port to 19200 Baud:
8 databits
no parity bit
1 stop bit
3. Writes "Hello, World!" to the serial port.
4. Close the Port.

Click **CTRL +X** to close the editor. To run this program you must compile it, therefore you would use the gcc compiler "`gcc -o uart1 uart1.c`"i.e., type "make". There should be no errors and you should get a `uart1.oUartDemo` file.

To see the transmitted output, open a terminal program on the Windows machine in the Virtual Lab. Start the `terminal.exe` on your desktop and configure it as shown in [Figure 18](#). Click **Connect**.

Figure 18. Configuring the Driver for the UART IP Core



Switch back to the PuTTY session, type the command `./UartDemo` and you will see the output on your terminal program. If you do not have the permission to open the serial port, use the command "su" to log in as root. The password is **irp**.

```
#include <stdio.h> /* Standard input/output definitions */
#include <string.h> /* String function definitions */
#include <unistd.h> /* UNIX standard function definitions */
#include <fcntl.h> /* File control definitions */
#include <errno.h> /* Error number definitions */
#include <termios.h> /* POSIX terminal control definitions */

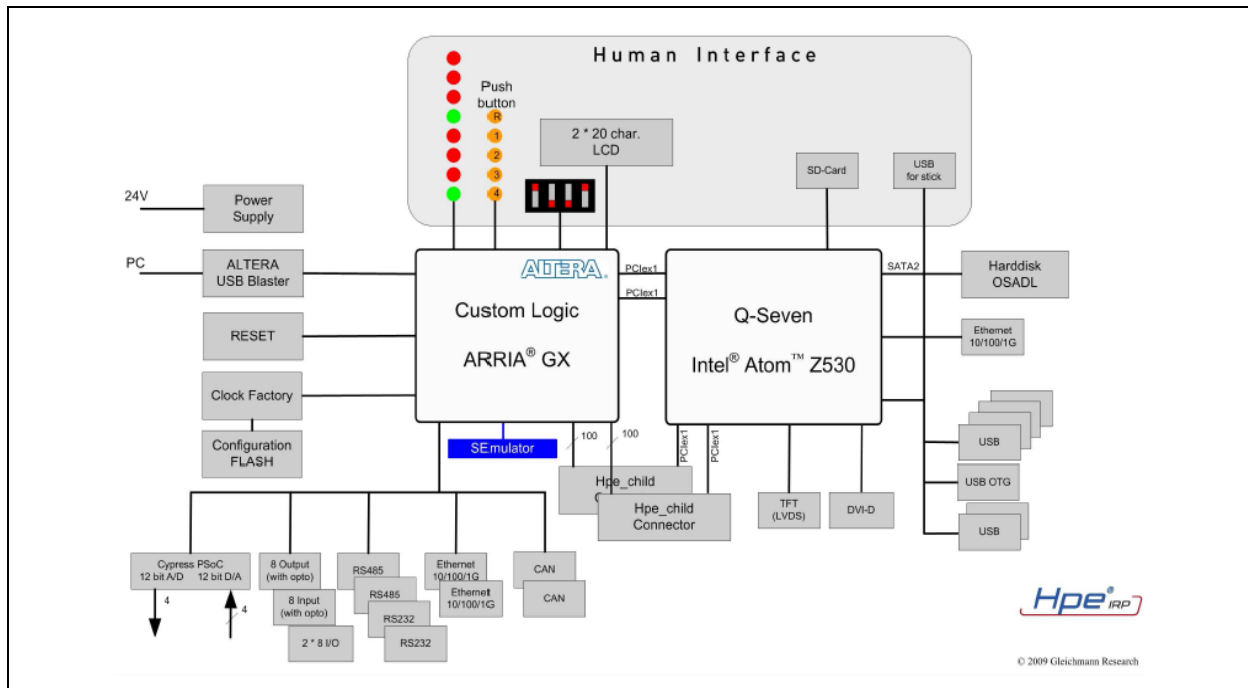
int main(void)
{
    int fd; /* File descriptor for the port */
    struct termios options;
    /* Open a Port
    *****/
    fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NDELAY);
    if (fd == -1)
    {
        /* Could not open the port. */
        perror("open_port: Unable to open /dev/ttyS0 - ");
    }
    else
    {
        fcntl(fd, F_SETFL, 0);
    }
}
```

```
/* Configure the Port
*****/
tcgetattr(fd, &options);          /* read options */
cfsetispeed(&options, B19200);    /* set 19200 Baud */
cfsetospeed(&options, B19200);
/* Enable the receiver and set local mode... */
options.c_cflag |= (CLOCAL | CREAD);
options.c_cflag &= ~PARENB      /* no Parity */
options.c_cflag &= ~CSTOPB     /* 1 stopbit */
options.c_cflag &= ~CSIZE;     /* 8 databits */
options.c_cflag |= CS8;
tcsetattr(fd, TCSANOW, &options); /* store options */
/* Write to the Port
*****/
n = write(fd, "Hello, World!\r", 14);
if (n < 0)
{
    fputs("write() of 14 bytes failed!\n", stderr);
}
/* Close the Port
*****/
close(fd);
return 0;
}
```

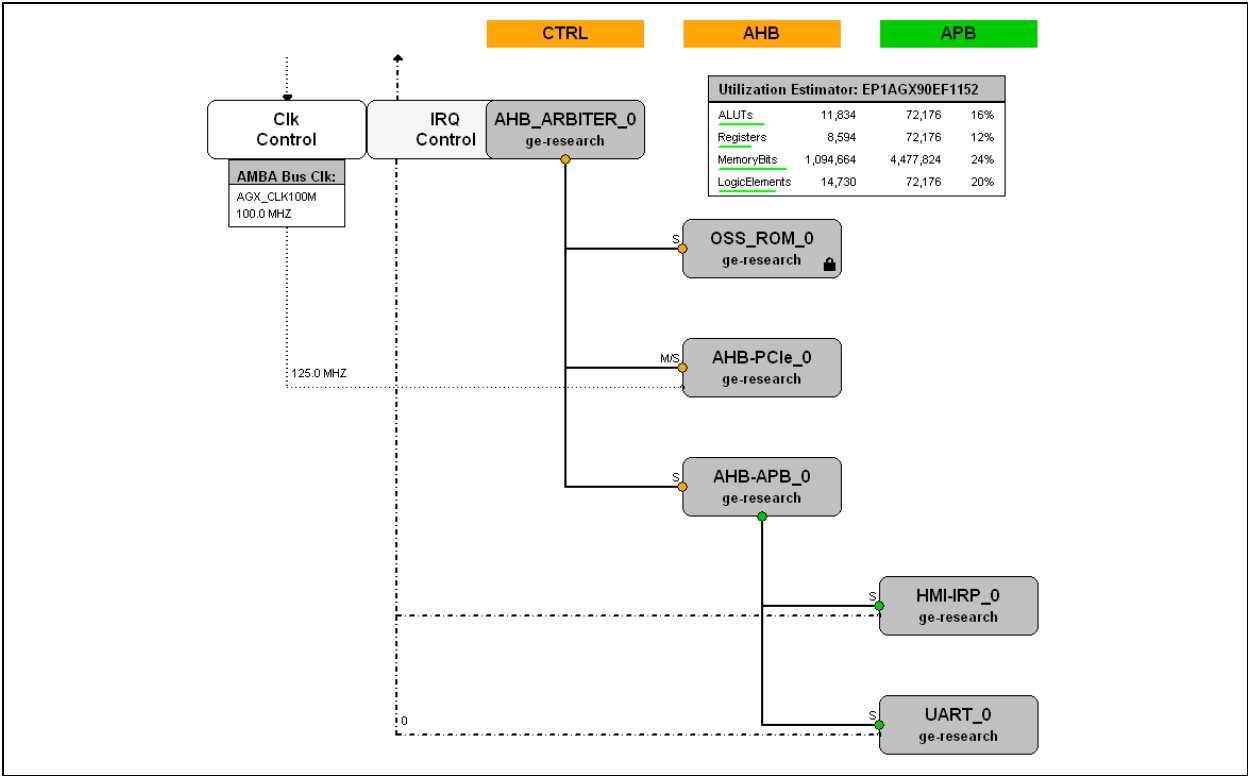
5 Summary

Finally we want to summarize what you learned in this demo.

1. You learned about the architecture of the Hpe[®] IRP.



2. You learned that a Gentoo Linux with an OSADL real time kernel is running on the Hpe[®] IRP.
3. You learned that the FPGA is connected to the Intel[®] Atom™ processor by PCIe*.
4. You learned that the Linux drivers automatically read information of the IP Cores from an OSS ROM in the FPGA.
5. You learned that the content of this OSS ROM is generated by Hpe[®] Desk before synthesis.
6. You learned about the Linux* environment.
7. You learned how to establish a connection between your development machine and the Hpe[®] IRP.
8. You learned how to access IP cores within the FPGA.
9. You learned how to create an FPGA design with the AMBA-IP-Manager.



10. You learned which IP Cores we need in every design for the Hpe[®] IRP.
11. You learned which IP cores are available.
12. You learned how to make the pin assignment.