Unit / Module Description:	Capture Demo For Intech
Unit / Module Number:	EVP6472-SMT911
Document Issue Number	1.1
Issue Data:	6th October 2012
Original Author:	C Hong

EVP6472 Intech Demo

Abstract

WLAN 802.11g RF demonstration based on EVP6472 and SMT911

1. Features and Requirements

To run the application:

1. Sundance hardware:

   SMT372T, SMT111 and SMT911

2. Sundance Software:

   SMT6002 (flash programing) and SMT111 driver

3. Cables:

   SMA-SMA cable x1

   USB cable x1

   SMT111 power cable x1

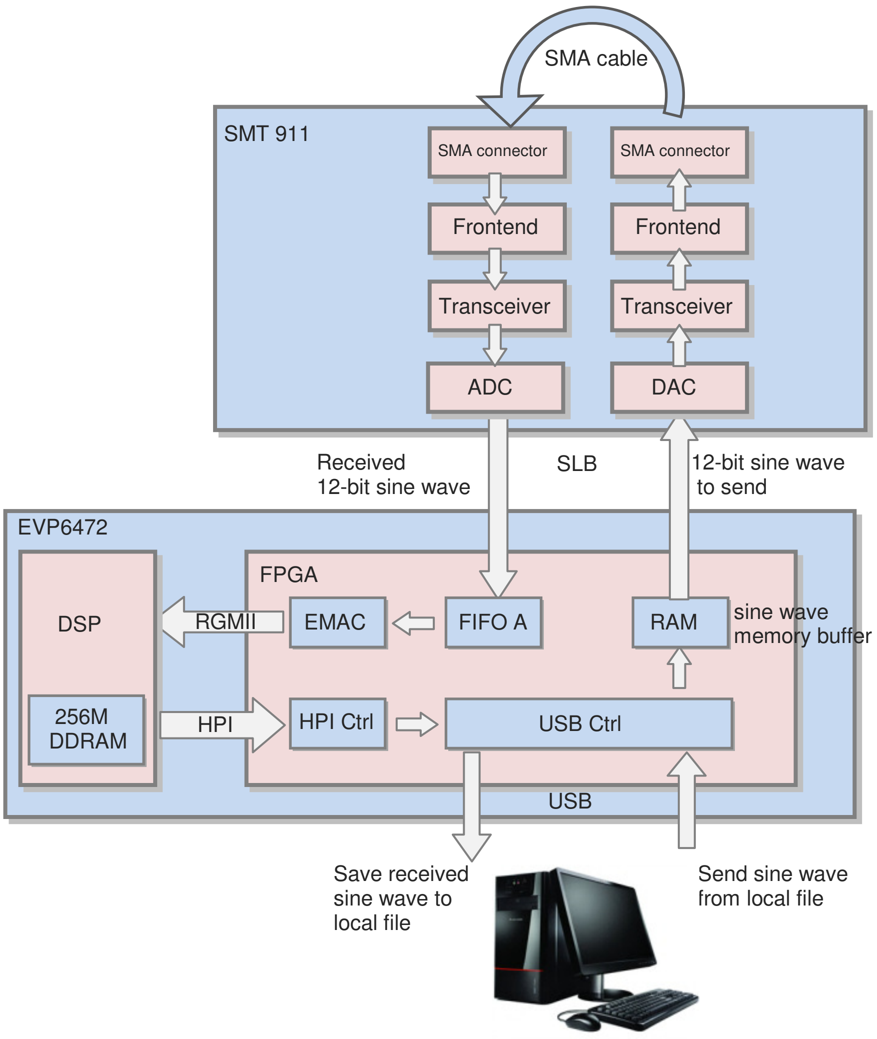
   Xilinx JTAG cable x1

To develop the application

1. Xilinx software tools: Xilinx EDK, SDK (12.3)
2. Taxes Instrument software tools: Code Composer Studio (CCS) (4.24)
3. Visual Studio 2008 (Compiled in WindowsXP 32-bit OS)

   *Source code is generated with the version in the bracket

2. Data Path



1. The data (one period sine wave) to be transmitted is pre-stored on the host PC
2. The host sends the data to the FPGA memory buffer
3. The data in the buffer is periodically sent to the DAC, to form a continuous sine wave
4. The 12-bit digital sine wave is converted to analogue (40MHz sampling rate) and used to modulate 2.4GHz RF carrier, then transmitted through SMA connector
5. The RF signal is loopback to the SMT911 receiver, demodulated, and converted (40MHz sampling rate) to digits
6. The received sine wave is buffered in FIFOs. Once the received data size meets the Ethernet package size, the data in FIFO is transmitted to DSP via the RGMII link.
7. The DSPs receive the pixels and stores it in their 256M DDR RAMs.
8. The received signal stored in the DDR RAM can be read through Host-Port Interface (HPI) by the FPGA, and sent to the host PC through the USB cable.

3. How the system works

After device configured:

1. After boot, the FPGA is automatically configured by the bitstream in the flash memory. The FPGA is configured as an embedded processor (MicroBlaze), and its peripherals used for accessing various interfaces.
2. MicroBlaze reads the DSP code from Flash memory.
3. MicroBlaze writes the DSP code to the DSP program memory through the HPI interface.
4. MicroBlaze sets up the DSP configuration through HPI interface.
5. MicroBlaze resets and starts the DSP.
6. MicroBlaze configures its capturer peripheral e.g. EMAC package size and package numbers.
7. The sine wave to be transmitted is read from host and stored in BRAMs on the FPGA.
8. MicroBlaze configures the transceiver on SMT911 through SPI bus, and sets control signals for ADC/DAC, transceivers, and frontends, then starts the transmission.
9. The transmitter continuously sends the signal to SMT911
10. At the receiver side, after one package size of data is sampled, the capturer starts one EMAC transmission to transmit the received data to the DSP.
11. After the all required frames are transmitted, DSPs can start to process the data.
12. Once requested by the host, the received signal stored in the DDR RAMs can be read through the HPI and transmitted to the host PC via USB.

4. How to run the demonstration

Every time the EVP6472 boots up, it is self configured by loading the application code (bitstream for the FPGA and binary code for the DSP) from the flash memory. Therefore we need to program the flash for the very first time.

After flash programmed, the EVP6472 automatically starts working every time it boots up.

4.1 First time flash programming/recovery

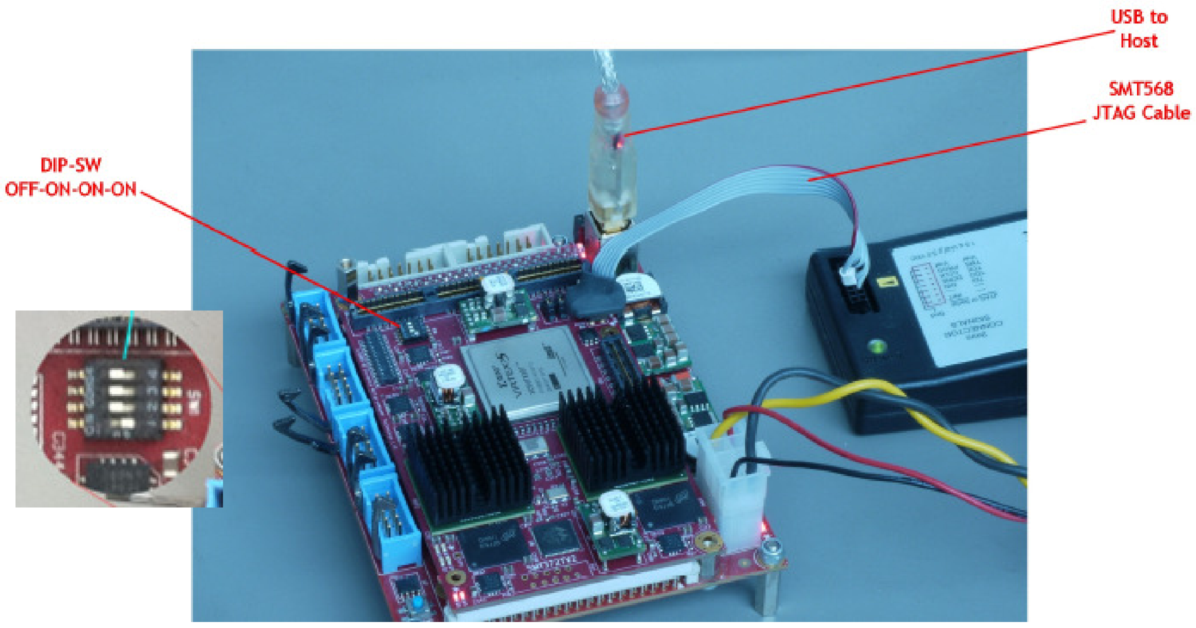
To program the flash, we use JTAG to load the bitstream to the FPGA, which communicates with the host through the USB.

The bitstream used for programming flash is the same one used for our normal application. It performs one of the actions depending on the first bit of the DIP switch (1 for programming flash and 0 for normal application). To programme the flash, steps are followed.

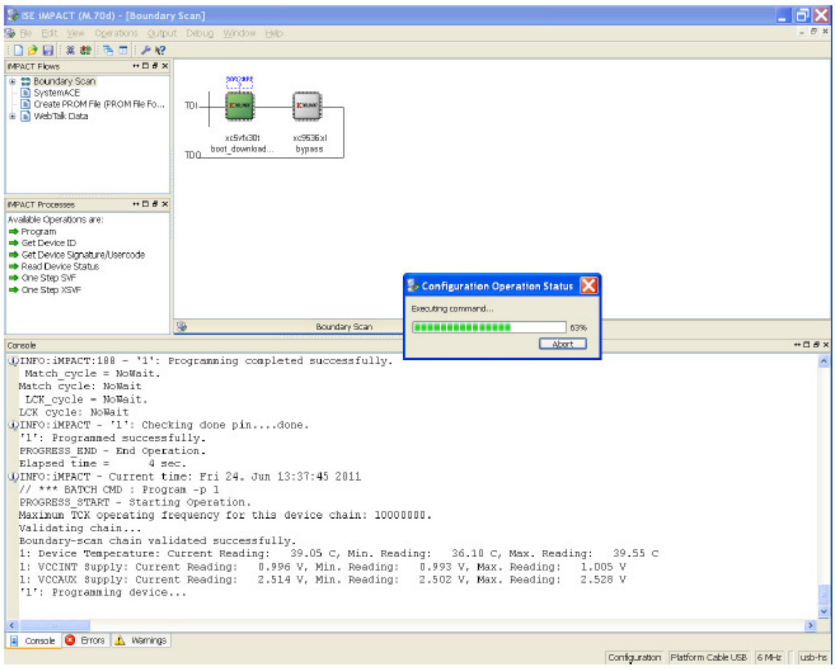
Ensure that the DIP-SW is set as shown above (position 1 OFF, the others ON).

Connect the USB cable from the SMT111 to the Host PC.

Connect the Xilinx programming pod to the SMT372T using an SMT568 JTAG cable.



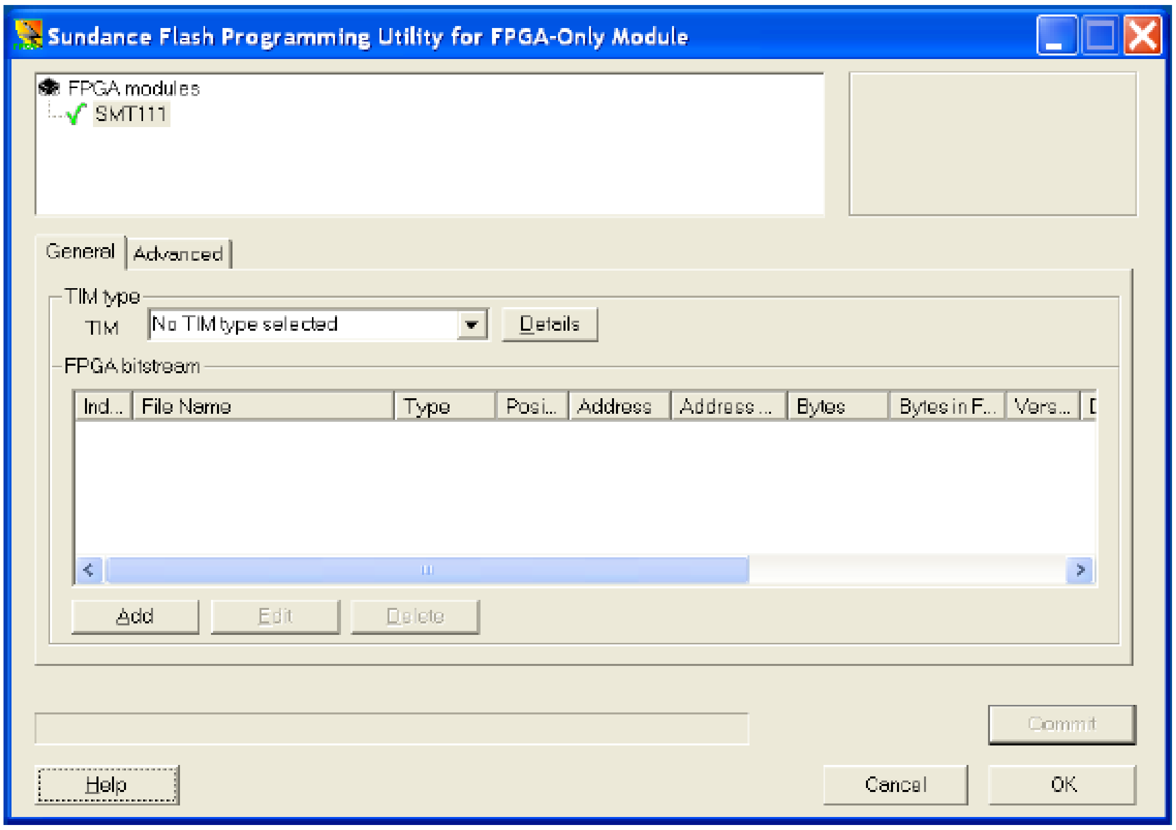
Run Xilinx impact programming tool and select boot_download.bit as the configuration file for the XC5VFX30T of the SMT372T.



```
INFO:iMPACT:188 - '1': Programming completed successfully.
 Match_cycle = NoWait.
Match cycle: NoWait
 LCK_cycle = NoWait.
LCK cycle: NoWait
INFO:iMPACT - '1': Checking done pin....done.
'1': Programmed successfully.
PROGRESS_END - End Operation.
Elapsed time =      4 sec.
INFO:iMPACT - Current time: Fri 24. Jun 13:37:45 2011
// *** BATCH CMD : Program -p 1
PROGRESS_START - Starting Operation.
Maximum TCK operating frequency for this device chain: 10000000.
Validating chain...
Boundary-scan chain validated successfully.
1: Device Temperature: Current Reading:   39.05 C, Min. Reading:   36.10 C, Max. Reading:   39.55 C
1: VCCINT Supply: Current Reading:   0.996 V, Min. Reading:   0.993 V, Max. Reading:   1.005 V
1: VCCAUX Supply: Current Reading:   2.514 V, Min. Reading:   2.502 V, Max. Reading:   2.528 V
'1': Programming device...
```

Configure the FPGA. When configuration is complete press the reset button on the SMT111. Do NOT power off the EVP system.

Run the SMT6002 application.

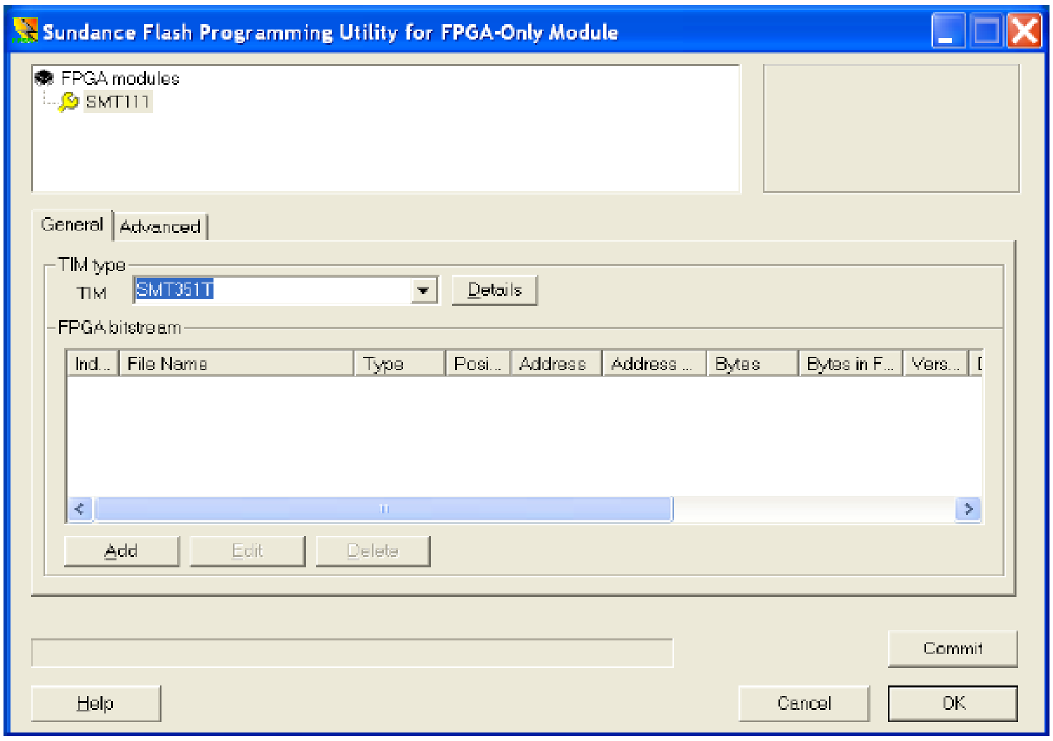


TIM type

TIM: No TIM type selected

FPGA bitstream

Select the TIM type as SMT351T.



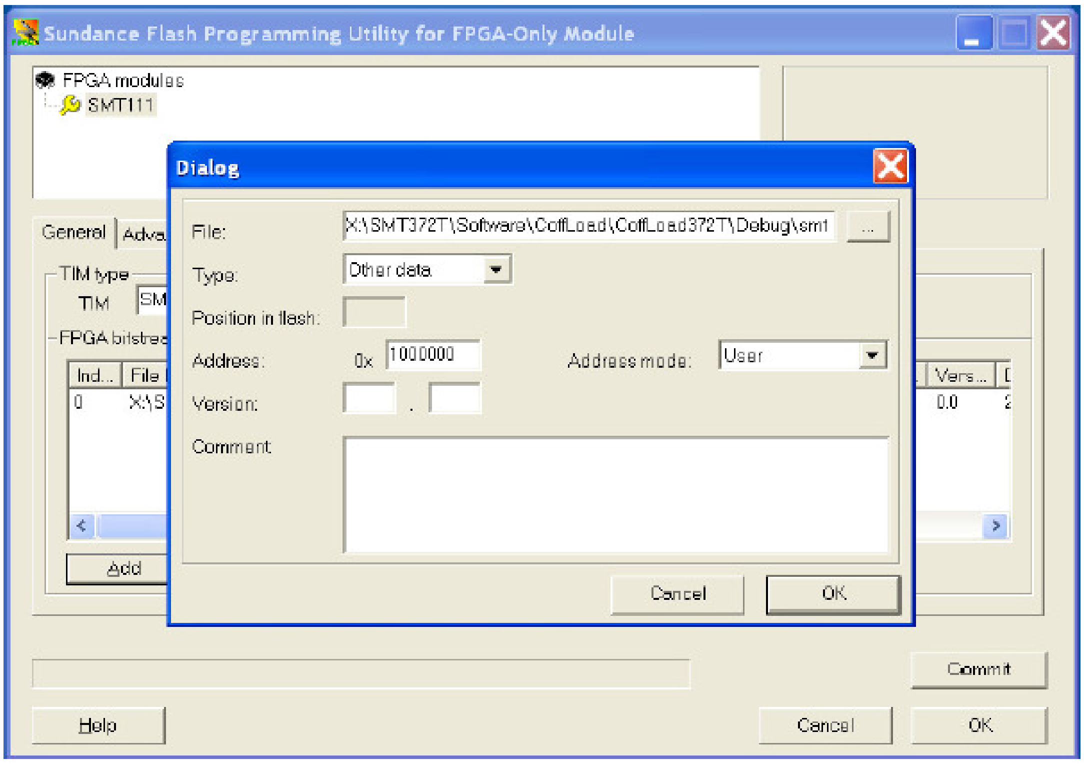
TIM: SMT351T

Add the “FPGA.bit” bitstream:

Type: Bitstream; address 0x0; address mode: basic.

Then add the “DSP.bin” DSP application:

Type: other data; address 0x1000000; address mode: user.



File: X:\SMT372T\Software\CoffLoad\CoffLoad372T\Debug\smt

Type: Other data

Position in flash:

Address: 0x 1000000 Address mode: User

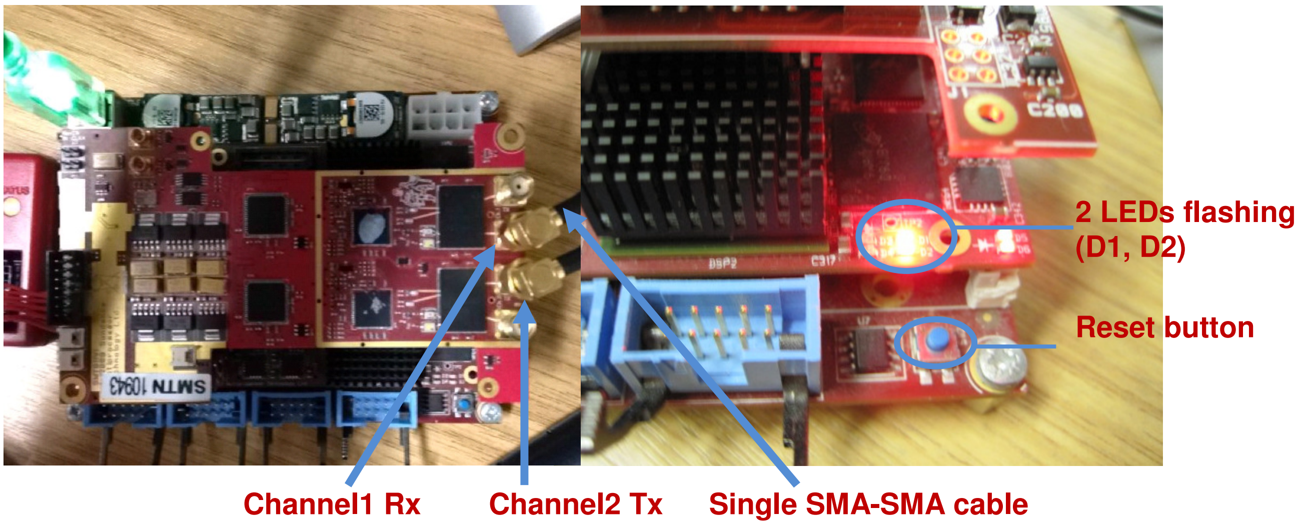
Version: .

Comment:

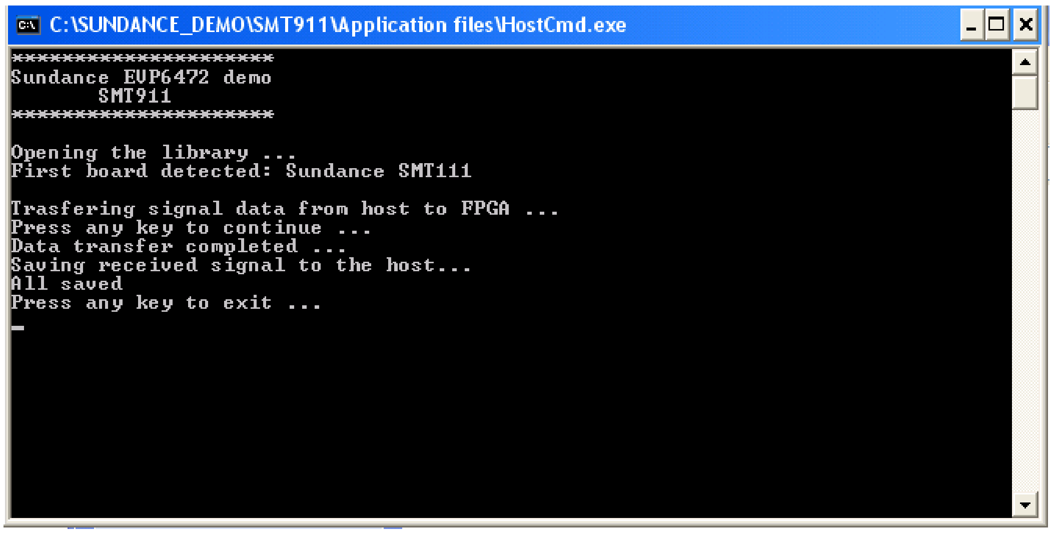
Click OK, then Commit. When programming has finished close SMT6002.

4.2 Run the application

1. Set the DIP-SW as “1111” for normal application.
2. Power off the board, plug in SMT911, and connect two SMA connectors (Channel1 Rx to Channel2 Tx) with a SMA-SMA cable, as shown in the picture.
3. Power on the board, press the reset button to reset the DSP. If success, 2 of the 4 LEDs (D1 and D2) should flash for 1 second.



4. Make sure the file “transmit_sine.data” is in the same directory with “HostCmd.exe”.
5. Run the host program (HostCmd.exe). On success, “received_sine.data” will be generated.



```
**********************
Sundance EVP6472 demo
       SMT911
**********************

Opening the library ...
First board detected: Sundance SMT111

Trasfering signal data from host to FPGA ...
Press any key to continue ...
Data transfer completed ...
Saving received signal to the host...
All saved
Press any key to exit ...
_
```

6. Plot the data to see the received sinewave. (below is an example plotted in Matlab)

Fig.a: the plotted "transmit_sine.data" (only one period needed)

Matlab command:

fid = fopen('transmit_sine.data');

m1= fread(fid, [1,256], 'uint8');

plot(m1,'DisplayName','m1','YDataSource','m1');figure(gcf)

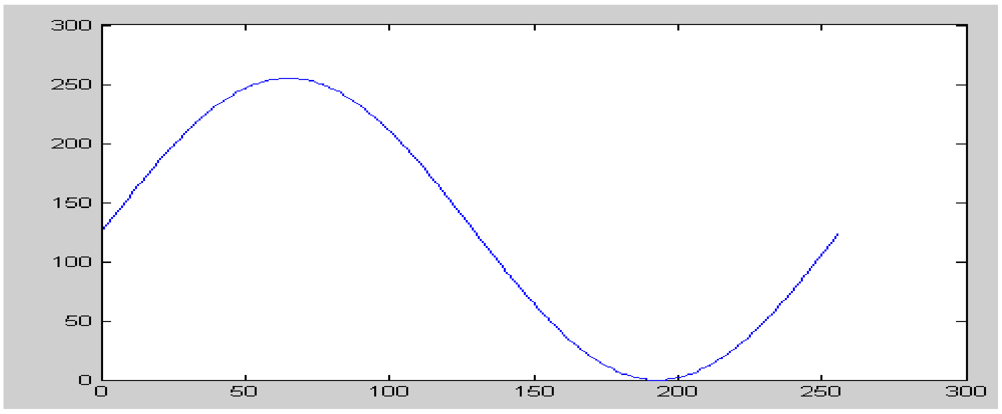


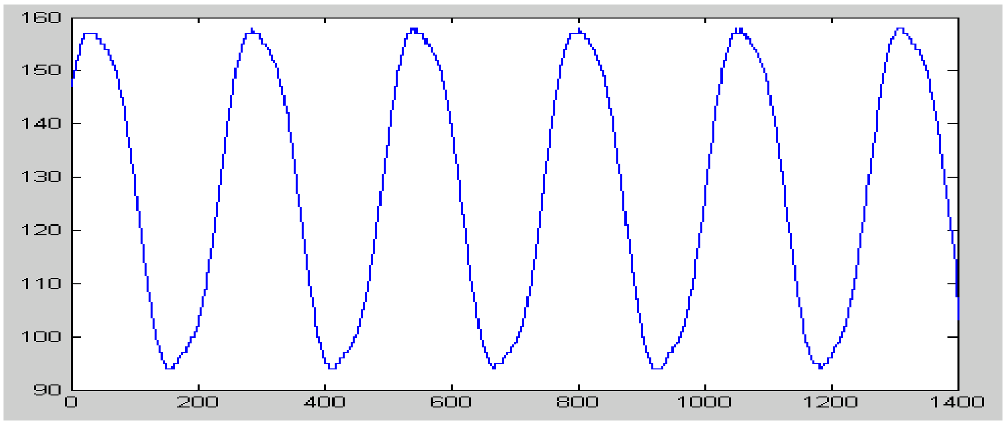
Fig.b: the plotted "received_sine.data" (1400 samples)

Matlab command:

fid = fopen('receive_sine.data');

m1= fread(fid, [1,1400], 'uint8');

plot(m1,'DisplayName','m1','YDataSource','m1');figure(gcf)


160
150
140
130
120
110
100
90
0
200
400
600
800
1000
1200
1400