Unit / Module Description:	SMT111-SMT372T-SMT946
Unit / Module Number:	
Document Issue Number:	1.0
Issue Date:	November 2012
Original Author:	Graeme Parker

Application Note for EVP6472-946

Sundance Multiprocessor Technology Ltd, Chiltern House, Waterside, Chesham, Bucks. HP5 1PS.

Revision History

Issue	Changes Made	Date	Initials
1.0	Initial release	Nov'12	GKP

Please read fully for a thorough understanding of this application note.

Table of Contents

1 Introduction

This application note describes the processes involved in sampling data from the SMT946 into the DSP's memory.

It also describes a method to "replay" the samples through the SMT946's DACs.

The SMT946 consists of 4 ADCs and 4 DACs as separate devices. A local 25MHz clock is also present. All of these devices are connected to the Sundance Local Bus (SLB) via logic/voltage level translators where appropriate.

The local clock is simply fed via the SLB to the SMT372T's FPGA as a sampling reference clock. Two MMCX connectors are also present on the SMT946, each of which is also routed to the FPGA via the SLB connector.

It is the responsibility of the FPGA to generate the ADC sample start command, to receive the ADC sampled data, and to send DAC data.

The described firmware is based on a Xilinx Microblaze (MB) processor. This is the core component of the SMT372T firmware and is responsible for allowing communication to and from the FPGA's attached peripherals (Ethernet, DSPs' Host Ports, SLB, etc.).

Several IP cores are provided within this FPGA. All sources for these cores is provided and is compatible with Xilinx ISE/EDK/SDK/BSB. Version 13.4 has been used in this instance.

2 Prerequisites

It is not essential in order to run this demonstration application but it is highly desirable for an in-depth knowledge of the following tools:

Xilinx ISE, EDK/SDK/XPS. Version 13.4 has been used throughout the development of this application.

ISE has primarily been used to create new IP cores and MB peripherals. Xilinx ChipScope has been used to show the logic functionality.

XPS/EDK has been used to create the MB system which is the heart of the firmware of the SMT372T.

SDK has been used to create the MB program which controls the whole system.

Code Composer Studio (CCS, Texas Instruments). Version 5 has been used in this instance. Its features show captured samples in a graphical way.

3 Firmware (system) Description

Whilst the major processing elements of the SMT372T are obviously the two TMS320C6472 DSPs, in some way they can be considered a peripheral of the FPGA.

When the SMT372T boots from power-on, the first action taken is that the FPGA is configured (automatically) from a bitstream held in flash memory. This bitstream can easily be changed using the Sundance SMT6002 flash programming utility (for use on a Windows host PC).

After the FPGA is configured, typically the firmware will include a Microblaze (MB) 32-bit processor. In the application described here, this processor is set to run at 125MHz. The MB processor clock is actually derived from the DSP's SYSCLKOUT pin. The SYSCLKOUT pin is set to 25MHz using a Phase Locked Loop (PLL). Note that the internal speed of the DSP is much higher than this.

In this application it is simpler to let the FPGA execute the default firmware, and then to load and execute the demonstration firmware using Xilinx's SDK tools via a Xilinx JTAG pod.

When the application firmware is loaded, cores are present to:

1) Read the serial data from the ADCs and make this available as 32-bit registers for the MB.
2) Write from MB accessible registers to the DACs.
3) DMA from either the ADC or DAC to the DSP's Host Port Interface (HPI).
4) Allow the MB to access the DSP's HPI.

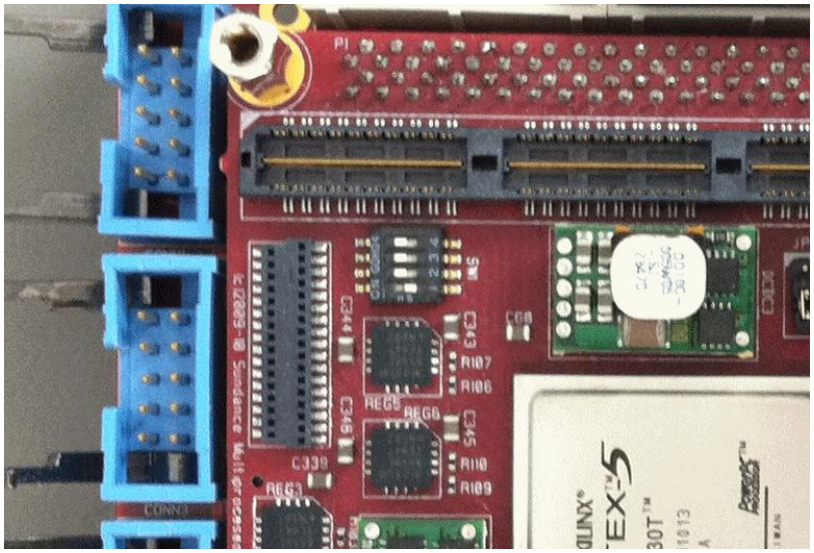
With the above cores in the firmware, the software that the MB runs performs the following:

1) Sets the DSP's PLL to allow the MB to run at 125MHz.
2) Sets the DSP's HPI destination address to point to DDR memory.
3) Initialises an ADC sample rate.
4) Sets up a DMA to copy ADC samples to the HPI.

The samples will be stored directly into the DSP's DDR memory. This can then be viewed in the time or frequency domains using the CCS embedded graph functions.

4 Hardware Setup

Set the SMT372T's DIP switches as shown here:



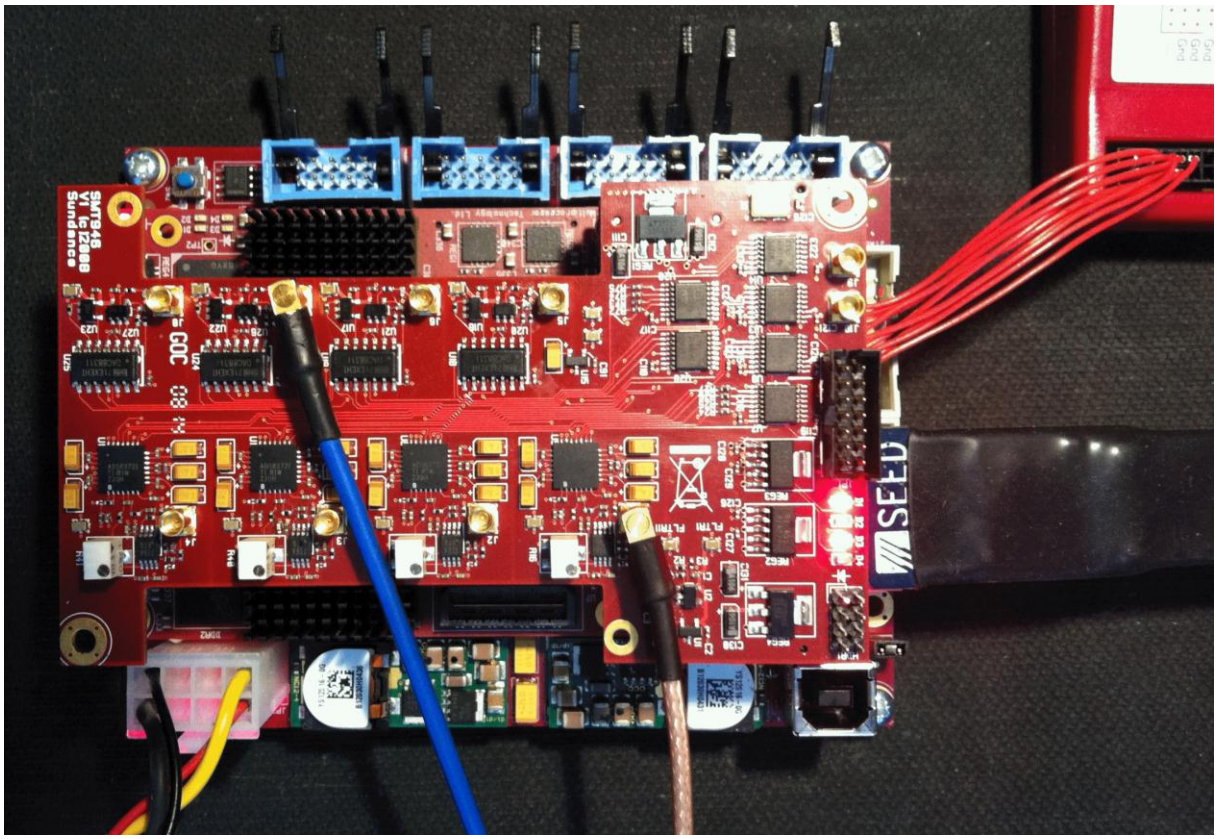
Build the system with the SMT372T screwed onto the SMT111 (this creates an EVP6472). Using two SLB extenders (with JTAG breakout), attach the SMT946 to the EVP6472.

Connect a signal generator and oscilloscope to the SMT946. The picture below shows the signal generator cable in brown and the oscilloscope cable in blue.

An EVP6472 USB connection to a host PC will allow the SMT372T's flash to be reprogrammed (not shown connected here).

The SLB extender's JTAG breakout cable will plug directly into a Xilinx JTAG programmer (pod).

CCS connects to the SMT111's JTAG1 header.

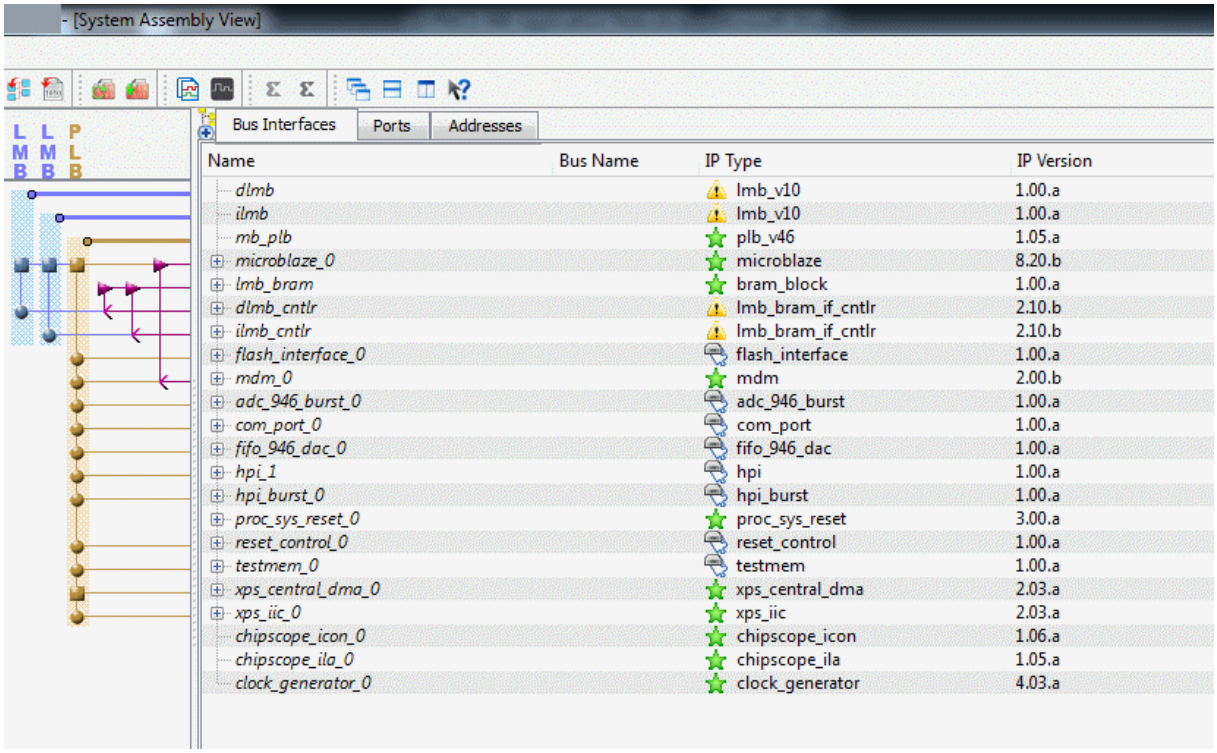


5 Xilinx Tools Setup

5.1 XPS/EDK Setup

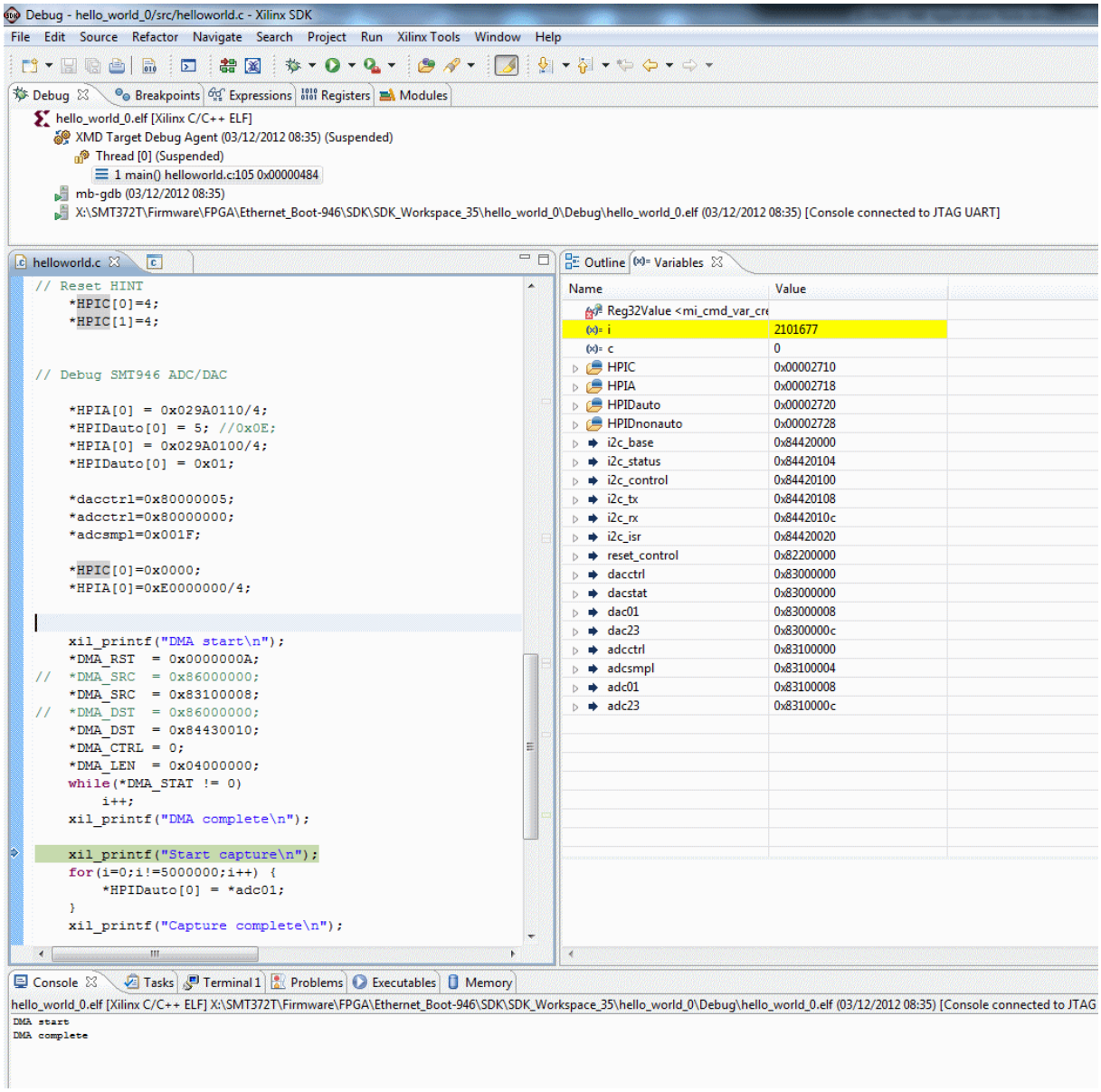
The following screenshot shows the XPS/EDK system assembly view open at the bus interfaces tab. This shows the MB and its components, and the peripherals for the ADC, DAC, DMA, and HPI.

The project is called system_372t.xmp and is provided as an archive.



5.2 SDK Setup

The following screenshot shows the SDK project during a debug session of the DMA process:



6 XPS/EDK Cores

All of the custom cores were created using XPS. The “create template” check box was selected to enable quick and easy development. The folder structures are the defaults that XPS creates.

6.1 ADC IP Core

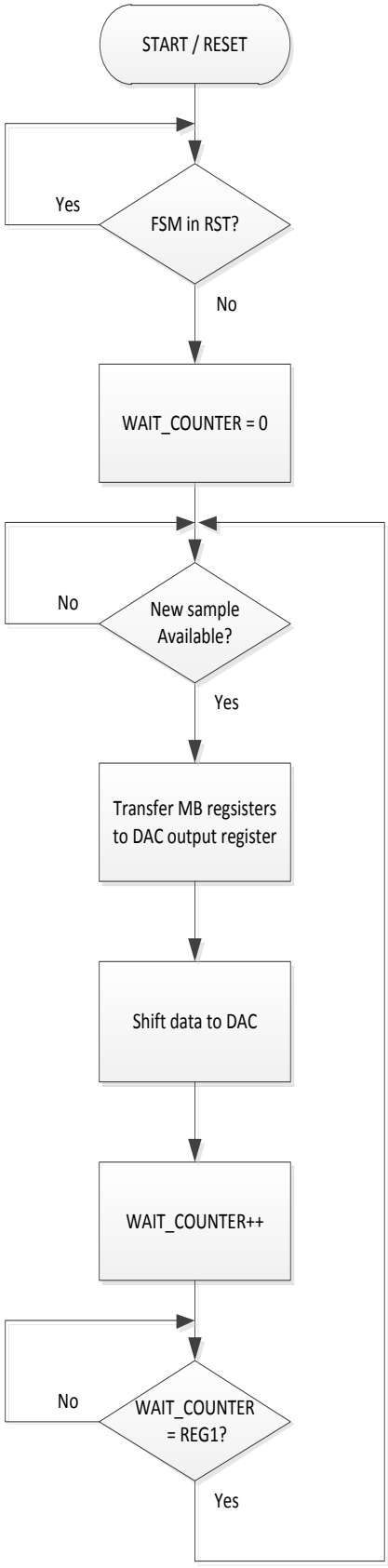
This core is located here:
.\Firmware\FPGA\Ethernet_Boot-946\pcores\adc_946_burst_v1_00_a

This core is presented with the following features:
It simultaneously samples all 4 channels at a rate determined by the setting in the sample wait counter. This counter is a 16 bit counter that is programmed by the MB (see address map). The ADC sampling is performed using a finite state machine (FSM) within the FPGA. When enabled, by writing to the control register, the FSM continues to run and control the physical ADC. The ADC sampling (see datasheet) is initiated by the assertion of the CONVST pin in conjunction with the CS pin. The ADC will then return a BUSY signal as is performs the requested conversion. During this time the inputs to the ADC should remain “quiet” (see datasheet). This improves the performance.
When BUSY is de-asserted the FSM retrieves the samples serially and stores them in 4 16-bit registers, one for each ADC channel. The MB can read these registers at a maximum speed equivalent to the sample rate. The MB has two memory addresses to read, each one returning the samples from two channels.
When attempting to read these sample registers, the MB will be put into a wait state until a valid sample is present.
The sample rate is equivalent to the speed at which the ADC can perform the conversion and the speed at which the FSM can generate the necessary ADC control signals. In addition to the basic FSM operation, a 16-bit counter is provided (see memory map) that delays the FSM operation. This is used to control the sample rate.

Address Offset	Register	Function
0	Control / status	Bit:0 1=enable FSM
4	Wait counter	16-bit FSM delay counter
8	ADC 0 & 1	Channels 0 & 1
C	ADC 2 & 3	Channels 2 & 3

Notes:
MB address is big endian.
This core has been created with burst support enabled.

The ADC core's FSM operation is shown here in flow diagram format:



6.2 DAC IP Core

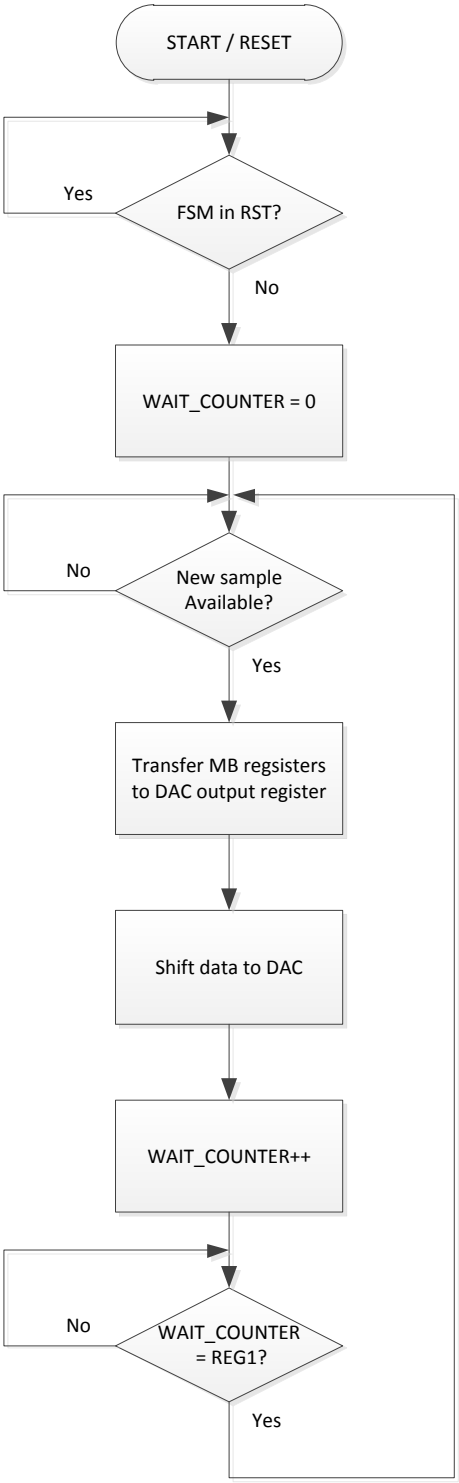
This core is located here:
.\Firmware\FPGA\Ethernet_Boot-946\pcores\fifo_946_dac_v1_00_a
Contrary to the core name, no FIFO is employed.

In a similar way to how the ADC core operates, when the MB writes to the DAC registers, the write transaction will only complete if the previous DAC register values are being shifter serially out to the DAC devices. If a sample remains in the DAC output registers, then the MB access cycle will stall until there is an empty transfer register.
A FIFO buffered interface is not needed for this demonstration as the MB is fast enough (coupled with DMA where necessary) to perform at rates well in excess of 100kHz.

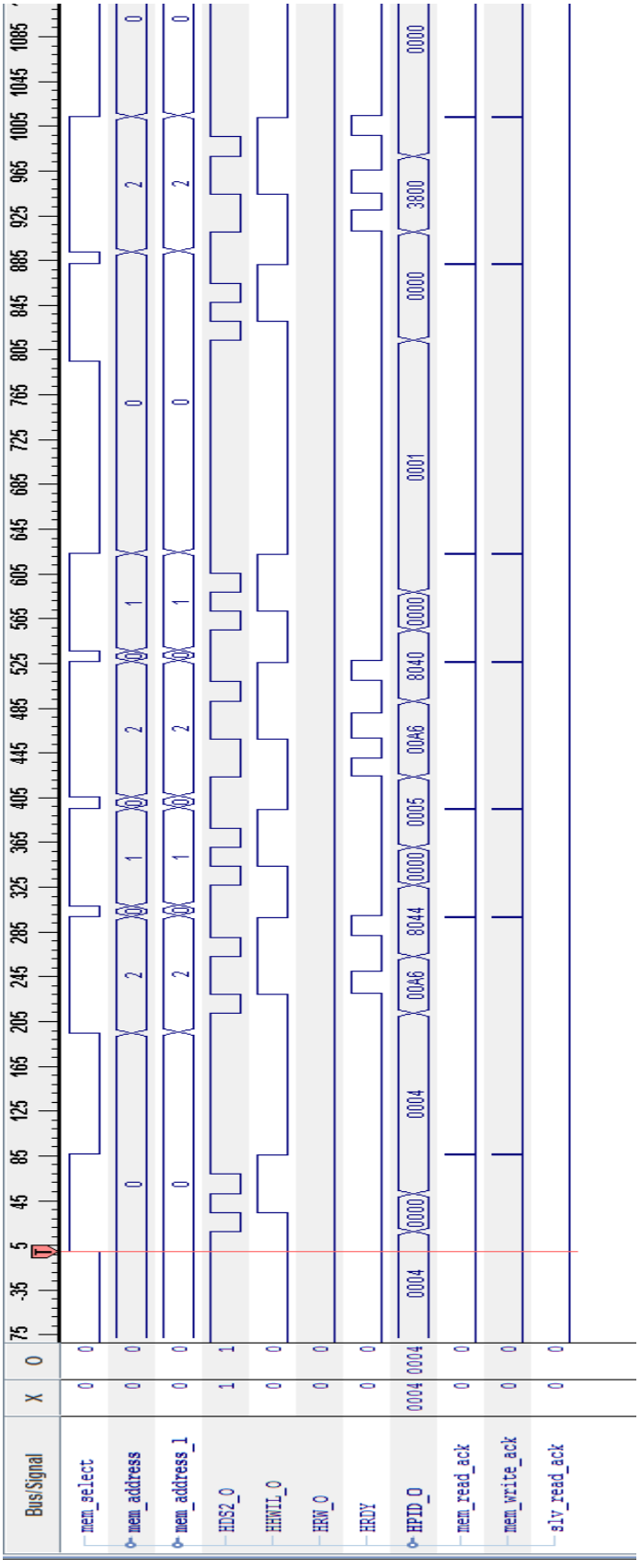
The DAC output word rate is timed in a similar fashion to the ADC by the use of a wait counter which can be programmed by the MB. In both cases (ADC and DAC) the wait counter (and associated FSM) operate at the LOCALCLK rate. This is based on a 25MHz fitted oscillator. Other clock source can be used such as the two MMCX connectors provided.

Address Offset	Register	Function
0	Control / status	Bit:0 1=enable FSM
4	Wait counter	16-bit FSM delay counter
8	DAC 0 & 1	Channels 0 & 1
C	DAC 2 & 3	Channels 2 & 3

The DAC core's FSM operation is shown here in flow diagram format:



6.3 HPI IP Core

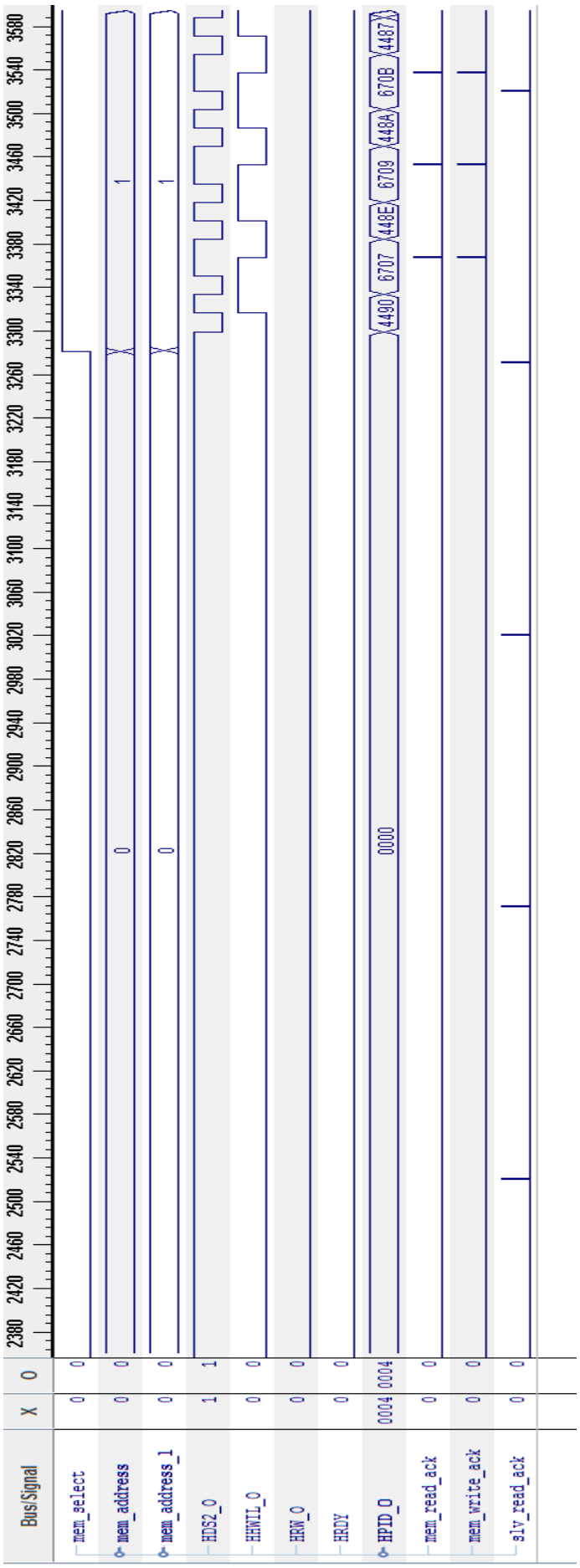


The previous timing diagram (captured from Xilinx ChipScope) shows the first HPI transfers that are made from the MB demonstration code.
The corresponding code excerpts are shown here:

```
*HPIC[0]=4;
*HPIA[0] = 0x029A0110/4;
*HPIDauto[0] = 5;
*HPIA[0] = 0x029A0100/4;
*HPIDauto[0] = 0x01;
*HPIC[0]=0x0000;
*HPIA[0]=0xE0000000/4;
```

In each HPI transfer you can clearly see the two 16-bit parts of the transaction; two HDS2_O active low strobes with alternate states for HHWIL_O.

The following diagram shows the regularly spaced slv_read_ack signal which corresponds to the ADC sampling rate. After a sufficient amount of samples have been made, the DMA controller transfers these samples over the HPI using a burst type transfer. You can see the two 16-bit parts forming samples for ADC channels 0 & 1.


Bus/Signal
X
O
mem_select
mem_address
mem_address_1
HDS2_O
HHWIL_O
HRW_O
HRDY
HPID_O
mem_read_ack
mem_write_ack
slv_read_ack
0004
0000
4490
6707
448E
6709
448A
670B
4487

6.4 Miscellaneous Cores

Other cores that are part of this application can be found in the Xilinx archived sources.

The DMA IP core is provided by Xilinx.

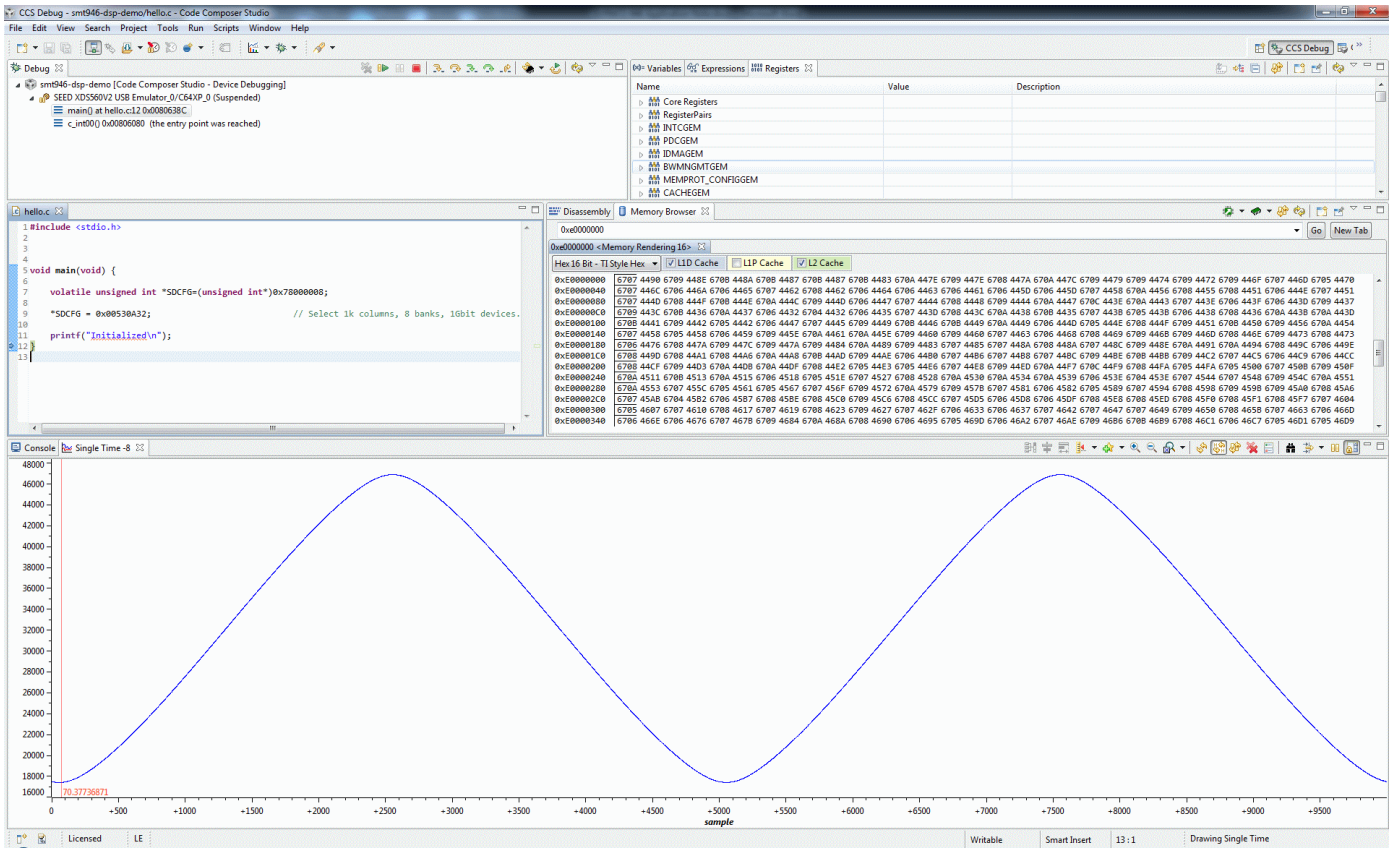
7 Code Composer

The following screenshot shows CCS connected to the first core of the first DSP in the JTAG scan path.

The memory browser is displaying the contents of the DDR memory which holds the ADC samples for channels 0 & 1. These are displayed as 16-bit unsigned integers.

The small demonstration program merely sets up the DDR controller. The DSPs' PLLs have already been setup using HPI transfers from the MB.

At the bottom of the screenshot can be seen a graphical display of the captured samples.



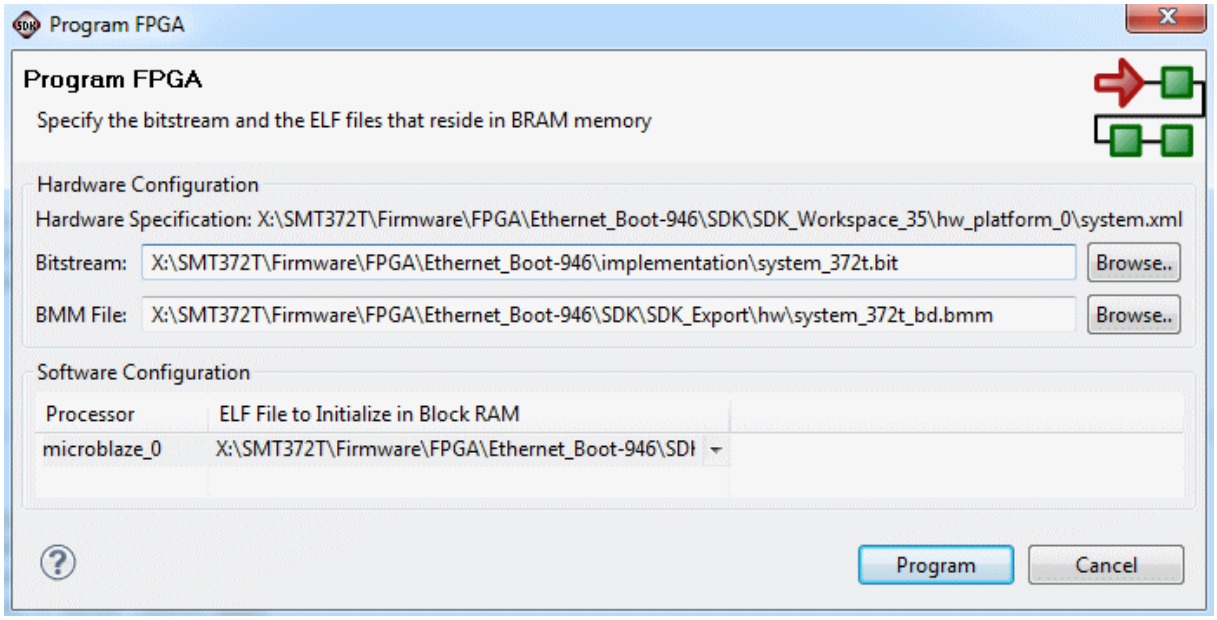
8 Running the Demonstration

Connect the Xilinx download pod to the SLB extender JTAG cable.
Connect the XDS560 (or compatible) to the EVP6472 connector labelled JTAG1.
Connect a suitable signal source as shown, and the DAC output to an oscilloscope.

Power on the system.

Run Xilinx SDK and load the workspace.
Run CCS but do not connect to the target yet. The target will be the first core of the first DSP in the JTAG path. As the first part of the MB program is to reset the DSPs, CCS and the XDS560 may be unsettled by this reset process if the core is connected.

Configure the FPGA using SDK option Xilinx Tools > Program FPGA.
The relevant options should already be set as shown here:



Run the MB until after the DMA Complete line. This will take about 34 seconds as it is capturing 1M samples at 500ks/s.
The next part of the program (commented out):

```
for(i=0;i!=5000000;i++) {
      *HPIDauto[0] = *adc01;
}
```

Shows capturing data without using DMA.

Debug (or Connect to the DSP core) in CCS. Examine the memory contents at address 0xE0000000 and display this as a single time graph. Start address should be 0xE0000002, 16-bit unsigned, index increment of 2.

The final program part shows “replaying” the samples using the MB in a simple read/write method.

The DAC output can be observed on an oscilloscope.

9 Related Documents

SMT372T User Guide
SMT111 User Guide
EVP6472 User Guide
SLB Specification
SMT946 User Guide

TMS320C6472 product page: http://www.ti.com/product/tms320c6472

ADC product page: http://www.ti.com/product/ads8372

DAC product page: http://www.ti.com/product/dac8831

10 Acronyms, Abbreviations and Definitions

CCS	Code Composer Studio
DDR	Dual Data Rate. Typically refers to memory.
DSP	Digital Signal Processor
EDK	Embedded Development Kit
FSM	Finite State Machine
MB	MicroBlaze (processor)
SDK	Software Development Kit
XPS	Xilinx Platform Studio