

Platform Developer's Kit

A thick yellow horizontal line with a right-pointing arrowhead at its end.

DSM FIR Filter Tutorial Manual

Celoxica, the Celoxica logo and Handel-C are trademarks of Celoxica Limited.

All other products or services mentioned herein may be trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous development and improvement. All particulars of the product and its use contained in this document are given by Celoxica Limited in good faith. However, all warranties implied or express, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. Celoxica Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any incorrect use of the product.

The information contained herein is subject to change without notice and is for general guidance only.

Copyright © 2003 Celoxica Limited. All rights reserved.

Authors: DS

Document number: UM-2340-1.2

Table of contents

Table of contents

TABLE OF CONTENTS.....	1
ASSUMPTIONS	2
OMISSIONS	2
1. INTRODUCTION	3
1.1 PURPOSE OF THIS TUTORIAL	3
1.2 THE TUTORIAL APPLICATION.....	3
1.3 REQUIREMENTS	3
1.4 THE TUTORIAL WORKSPACE	4
1.5 RUNNING THE TUTORIAL IN SIMULATION	5
2. TUTORIAL IMPLEMENTATION	8
2.1 DSM FIR SYSTEM DESIGN.....	8
2.1.1 Hardware side	8
2.1.2 Software side	11
2.2 RUNNING THE TUTORIAL IN HARDWARE.....	13
2.2.1 Targeting RC200/RC200E	13
2.2.2 Targeting the Memec Virtex-II Pro (MV2P)	14
2.3 BUILDING A HIGHPASS FILTER	15

Assumptions

Assumptions

This manual assumes that you:

- Have used Handel-C or have the Handel-C Language Reference Manual
- Are familiar with common programming terms (e.g. functions)

Omissions

This manual does not include:

- Instruction in Handel-C optimization
- Instruction in setting up DK for DSM (please refer to the *DSM User Guide*)
- Instruction in the use of place and route tools

1. Introduction

1. Introduction

1.1 Purpose of this tutorial

The purpose of this document and associated source code is to show a Handel-C programmer how to implement platform independent hardware-software co-designs between a processor and an FPGA using DK, the Handel-C language and the Data Stream Manager (DSM) API.

The tutorial runs on:

- the DSM Virtual Simulation Platform (Sim).
- the MicroBlaze platform running on Celoxica RC200, RC200E boards (MB_RC200, MB_RC200E).
- the Virtex-II Pro/PPC405 platform running on a Memec Design DS-BD-2VP7-FG456 REV2 board (MV2P).

For details of how to target hardware platforms with DSM applications, refer to the DSM User Manual.

1.2 The tutorial application

The example used in this tutorial guide is a FIR filter connected to a processor using DSM. The application sends a set of input samples stored in RAM to the FIR filter and reads the filtered data back. The input and output waveform is then displayed on the screen for DSM Virtual Simulation and MicroBlaze platforms. The connection to the video display is also based on a DSM layer.

In the case of the Virtex-II Pro/PPC405 platform, the FIR filter output is sent through the serial port to a console running on a PC. Optionally, if you have MATLAB 6.5 (Release 13) you can plot the output data on a PC screen.

The PSL and PAL API are used to provide platform abstraction for peripheral access and implementation of a simple FIR filter.

All interfacing between the software side and the hardware side is done using the DSM API.

1.3 Requirements

To compile the tutorial, you will need :

From Celoxica:

- DK Design Suite version 2.0 or greater
- Platform Developer's Kit (PDK) version 2.1 or greater

1. Introduction

- RC200 Development Board or RC200E Development Board or the Memec Design DS-BD-2VP7-FG456 REV2 Virtex-II Pro board (MV2P)

From Xilinx:

- Xilinx EDK 6.1i
- Xilinx ISE 6.1i
- Xilinx parallel or serial JTAG cable (for the Memec platform)

From Microsoft:

- Microsoft Visual C++ 6.0 or 7.0 to compile this tutorial for DSM Virtual Simulation Platform. (Other compilers may work but have not been tested).
- MS HyperTerminal (to display results from the Memec platform). MS HyperTerminal is provided with all MS Windows operating systems. Alternatively, you can use any other terminal program that supports the serial ports of your PC.

Other:

- RS-232 Serial cable

Optionally from MathWorks for MV2P Target:

- MATLAB 6.5 (Release 13), other versions might work, but have not been tested.



Please make sure that you have downloaded the latest MATLAB patch for serial communication problems. The patch is available at <http://www.mathworks.co.uk/support/solutions/data/34431.shtml>.

1.4 The tutorial workspace

The DK workspace for this tutorial can be accessed from the **Start** menu under **Programs>Celoxica>Platform Developer's Kit>DSM>DSM Examples Workspace [DK]**.

This workspace is already set up for use with DSM and is called **DsmFIR**.

1. Introduction

1.5 Running the tutorial in simulation

The 3 stages of the tutorial can be run in simulation using the DSM Sim Virtual Platform.

1. Open the tutorial **DK** workspace as described above.
2. Right-click the **DsmFIR** project in the left pane and select **Set Active Project**.
3. Choose to target the "Sim" platform (**Build>Set Active Configuration**).
4. Build the project by pressing F7.
5. Begin the simulation by pressing F5. A console window will open but will not generate any output.
6. Open the MSVC Examples workspace from the start menu: **Start>Programs>Celoxica>Platform Developer's Kit>DSM>DSM Examples Workspace [VC++]**.
7. Right click the **DsmFIR** project in the left pane and select **Set Active Project**.
8. Compile the project by pressing F7.
9. Execute the simulation by pressing F5.

When you run the simulation the PALSim application and the DSM Sim Monitor will appear.

The PALSim application allows you to simulate your PAL based designs providing a visual representation of the behaviour of devices such as a VGA screen, RAM and LEDs on a board.

When the simulation is finished the PALSim application appears as shown in Figure 1. The input data is represented by the green square wave signal. The samples of the input signal are stored in an array in the software source code. The output from the FIR filter is displayed in red.

1. Introduction

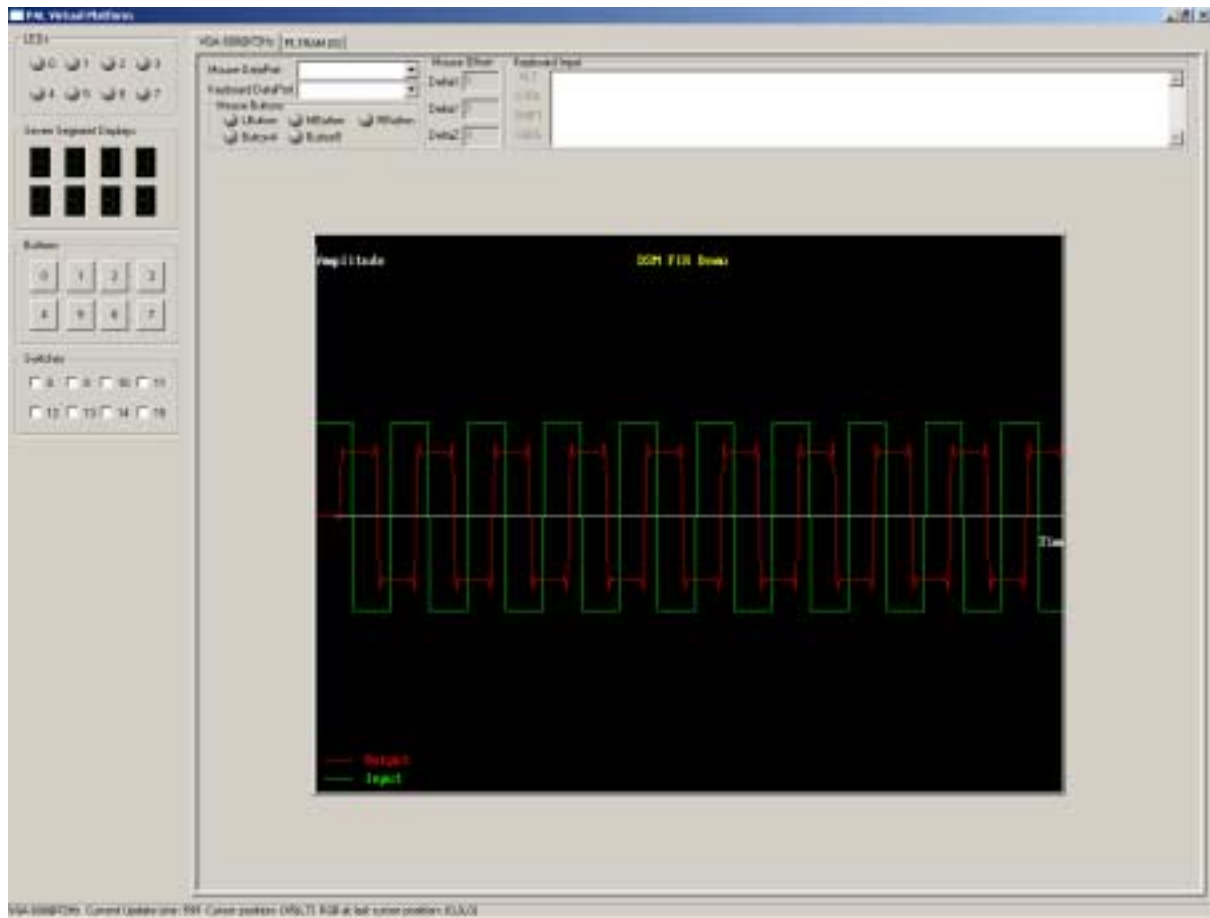
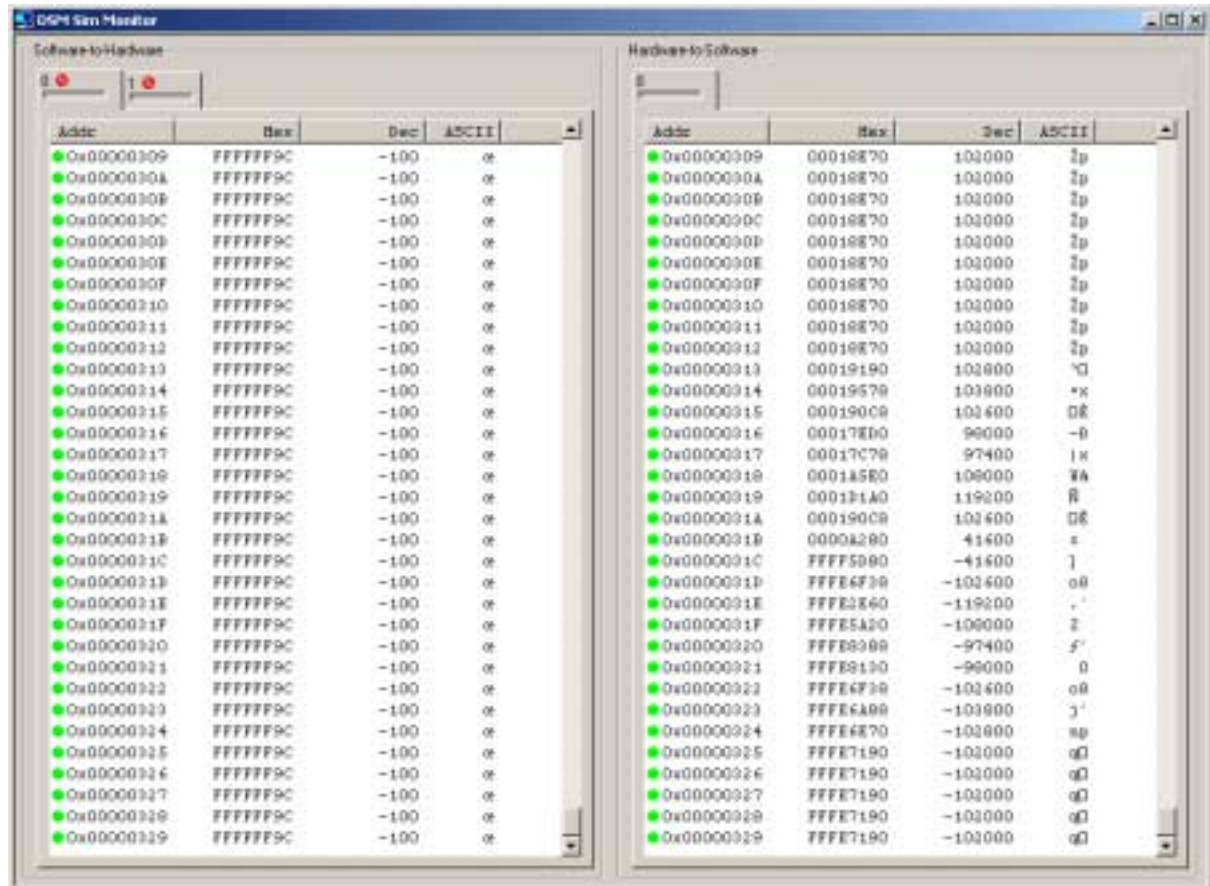


FIGURE 1: PALSIM APPLICATION IN FIR TUTORIAL EXAMPLE

The DSM Sim Monitor allows you to track transactions between the software and hardware sides. The number of DSM ports displayed depends on the number of hardware-to-software (H2S) and software-to-hardware (S2H) ports created in the Handel-C design.

The content of the DSM Sim Monitor for the example is shown in Figure 2. There are two software-to-hardware (S2H) ports, one to the FIR filter and another one to the framebuffer that serves as a display device. The hardware-to-software (H2S) port is from the FIR filter, and sends filtered data back to the software application.

1. Introduction



Addr	Hex	Dec	ASCII
0x0000309	FFFFFF9C	-100	œ
0x000030A	FFFFFF9C	-100	œ
0x000030B	FFFFFF9C	-100	œ
0x000030C	FFFFFF9C	-100	œ
0x000030D	FFFFFF9C	-100	œ
0x000030E	FFFFFF9C	-100	œ
0x000030F	FFFFFF9C	-100	œ
0x0000310	FFFFFF9C	-100	œ
0x0000311	FFFFFF9C	-100	œ
0x0000312	FFFFFF9C	-100	œ
0x0000313	FFFFFF9C	-100	œ
0x0000314	FFFFFF9C	-100	œ
0x0000315	FFFFFF9C	-100	œ
0x0000316	FFFFFF9C	-100	œ
0x0000317	FFFFFF9C	-100	œ
0x0000318	FFFFFF9C	-100	œ
0x0000319	FFFFFF9C	-100	œ
0x000031A	FFFFFF9C	-100	œ
0x000031B	FFFFFF9C	-100	œ
0x000031C	FFFFFF9C	-100	œ
0x000031D	FFFFFF9C	-100	œ
0x000031E	FFFFFF9C	-100	œ
0x000031F	FFFFFF9C	-100	œ
0x0000320	FFFFFF9C	-100	œ
0x0000321	FFFFFF9C	-100	œ
0x0000322	FFFFFF9C	-100	œ
0x0000323	FFFFFF9C	-100	œ
0x0000324	FFFFFF9C	-100	œ
0x0000325	FFFFFF9C	-100	œ
0x0000326	FFFFFF9C	-100	œ
0x0000327	FFFFFF9C	-100	œ
0x0000328	FFFFFF9C	-100	œ
0x0000329	FFFFFF9C	-100	œ

Addr	Hex	Dec	ASCII
0x0000309	00018870	102000	2p
0x000030A	00018870	102000	2p
0x000030B	00018870	102000	2p
0x000030C	00018870	102000	2p
0x000030D	00018870	102000	2p
0x000030E	00018870	102000	2p
0x000030F	00018870	102000	2p
0x0000310	00018870	102000	2p
0x0000311	00018870	102000	2p
0x0000312	00018870	102000	2p
0x0000313	00019190	102800	°D
0x0000314	00019578	103800	*x
0x0000315	000190C9	102400	DE
0x0000316	000178D0	98000	-0
0x0000317	00017C78	97400	ix
0x0000318	0001A5B0	106000	WA
0x0000319	000131A0	119200	R
0x000031A	000190C9	102400	DE
0x000031B	0000A280	41600	z
0x000031C	FFFF5080	-41600	1
0x000031D	FFFF6F38	-102400	o8
0x000031E	FFFF2E60	-119200	·
0x000031F	FFFF5A20	-106000	2
0x0000320	FFFF8088	-97400	5'
0x0000321	FFFF8130	-98000	0
0x0000322	FFFF6F38	-102400	o8
0x0000323	FFFF6A88	-103800	3'
0x0000324	FFFF6E70	-102800	wp
0x0000325	FFFF7190	-102000	qD
0x0000326	FFFF7190	-102000	qD
0x0000327	FFFF7190	-102000	qD
0x0000328	FFFF7190	-102000	qD
0x0000329	FFFF7190	-102000	qD

FIGURE 2 DSM SIM MONITOR CONTENTS

2. Tutorial implementation

2. Tutorial implementation

2.1 DSM FIR system design

The tutorial example describes hardware/software co-design. The first step is to create a system-level design. The hardware side of the design is then translated into Handel-C code, and finally, the software side of the system is translated into ANSI-C.

2.1.1 Hardware side

A block diagram of the system is shown in Figure 3.

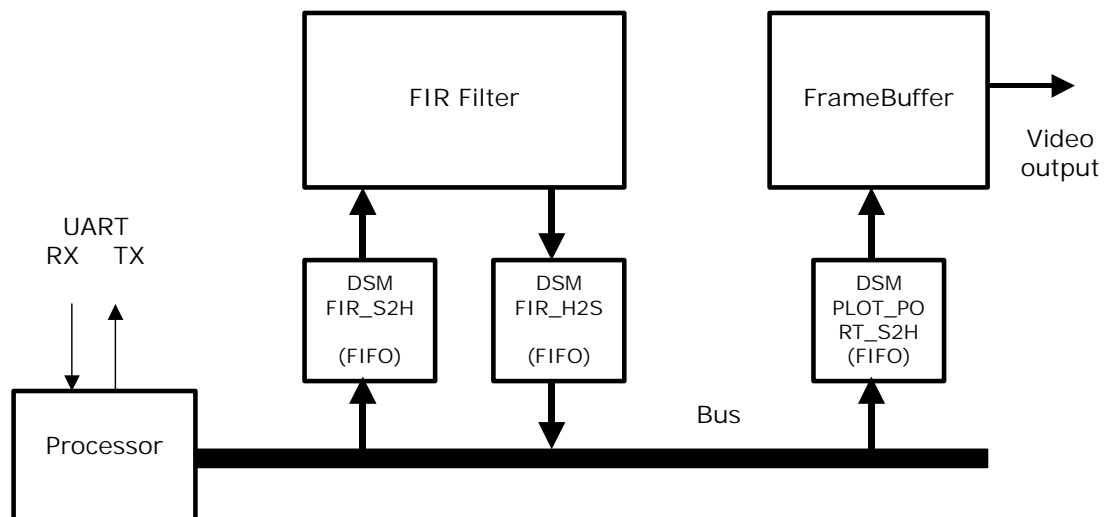


FIGURE 3: DSM FIR FILTER BLOCK DIAGRAM

The system consists of a FIR filter that is connected to the processor platform through DSM ports. One software-to hardware port is used to transfer data from software to FIR filter while the hardware-to-software (H2S) port is used to transfer data from the FIR filter back to the software side. To visualize the data processing a framebuffer is connected to the application. The framebuffer is connected to the software side through a DSM S2H port.

A Handel-C coded representation of the system in the block diagram for MicroBlaze running on RC200 is as follows:

2. Tutorial implementation

```
DsmVersionRequire (2, 0);
PalVersionRequire (1, 0);
PalVideoOutRequire (1);
PalPL1RAMRequire (1);

par
{
    DsmRun (DSM, DsmInterfaceDefault,
            DSM_INTERFACE_DATA,
            PORT_H2S_COUNT,
            PORT_S2H_COUNT);
    FIRFilter (FIRPortH2S, FIRPortS2H);
    FrameBuffer (PlotPortS2H);
}
```

FIR Filter implementation

The main task of the filter is to take input data and operate on it, and to provide results from operations on earlier input data.

The data value is the only thing sent from the software side to the FIR filter. The software side and the hardware side are connected together through a DSM S2H port.

The FIR filter is implemented in the `filterlib.hcl` library. It is a symmetrical type filter; the coefficients are symmetrical, so only half of them need be specified, rounded up if there are an odd number. e.g. {1,1,0,0,1,1} would be {1,1,0}, and {1,1,0,0,0,1,1} would be {1,1,0,0}.

To use the filter, you need to include `filterlib.hch`. This is done at the beginning of the source file `dsm_fir.hcc`. Then the `dsm_fir.h` header file is included. This is shared between hardware and software sides, and defines coefficients for the FIR filter and type of the filter. If you want to build a highpass filter you need to define the `HIGHPASS` preprocessor macro (`#define HIGHPASS`) in this header file. If the macro is not defined the default is a lowpass filter.

The `FIRFilter()` macro reads data from the DSM S2H port and stores it in the variable `input`. The value of `input` is then written to the FIR filter and processed. The new result available from the filter is then sent to the DSM H2S port.

The ports are read and written with the `DsmRead()` and `DsmWrite()` macros, e.g.:

```
macro proc FIRFilter (PortH2S, PortS2H)
{
    unsigned Input;
    signed (PFirSIResultWidth (DATAWIDTH, TAPS)) Output;

    par
    {
        /* Run the input and output FIR ports */
        DsmPortS2HRun (PortS2H);
```

2. Tutorial implementation

```

DsmPortH2SRun (PortH2S);
PFirSIRun (&myFIR, DATAWIDTH, TAPS, Coeffs, EXTRA_REGS);

while (1)
{
    seq
    {
        /* Read sample from DSM */
        DsmRead (PortS2H, &Input);
        /* Enable FIR Filter */
        PFirSIRenable (&myFIR);
        par
        {
            /* Write new sample to the filter */
            PFirSIWrite (&myFIR, (signed)Input<-DATAWIDTH);
            PFirSIRread (&myFIR, &Output);
            PFirSIRdisable (&myFIR);
        }
        DsmWrite (PortH2S,
                  (unsigned) adj_s (Output, width(DsmWord)));
        DsmFlush (PortH2S);
    }
}
}
}

```

Framebuffer implementation

The framebuffer is used to display processed data in visual form on a monitor screen.

The only things sent from software side of the framebuffer are the coordinates and colour of the pixel to be displayed on the screen. The software and hardware sides are connected together using a DSM S2H port. The index number of the port is PLOT_PORT_S2H. Again, reading is done using DsmRead(). DsmRead() and Pal FrameBuffer16Write() are running in sequence in a never-ending loop.

```

macro proc FrameBuffer (PortS2H)
{
    macro expr ClockRate = PAL_ACTUAL_CLOCK_RATE;
    macro expr PL1RAM     = Pal PL1RAMCT (0);
    macro expr VideoOut   = Pal VideoOutOptimal CT (ClockRate);
    unsigned Coordinate, Colour;
    Pal FrameBuffer16 *FBPtr;

    par
    {
        /* Run the plot port */
        DsmPortS2HRun (PortS2H);
        /* Run the frame buffer */
        Pal FrameBuffer16Run (&FBPtr, PL1RAM, VideoOut, ClockRate);
        /* Pass received data to the console */
    }
}

```

2. Tutorial implementation

```

seq
{
    PalFrameBuffer16Enable (FBPtr);
    while (1)
    {
        DsmRead (PortS2H, &Coordinate);
        DsmRead (PortS2H, &Colour);
        PalFrameBuffer16Write (FBPtr, Coordinate[9 : 0],
Coordinate[25: 16], Colour <- 24);
    }
}
}
}

```

2.1.2 Software side

The software side is implemented in `dsm_fir.c`.

The requirement for S2H and H2S ports is the same as in the hardware side, making the initialization and use of DSM in software as follows:

```

int FirFilter (DsmInterface Interface, void *InterfaceData)
{
    DsmInstance *Instance;
    DsmPortS2H *FirPortS2H;
    DsmPortS2H *PlotPortS2H;
    DsmPortH2S *FirPortH2S;

    /* Initialize Input buffer with samples */
    DsmWord Input[] = {
#include "samples.h"
    };
    ...

    DsmSetDefaultErrorHandler ();
    DsmInit (Interface, InterfaceData,
            PORT_H2S_COUNT, PORT_S2H_COUNT, &Instance);
    DsmPortS2HOpen (Instance, FIR_S2H, &FirPortS2H);
    DsmPortH2SOpen (Instance, FIR_S2H, &FirPortH2S);
    DsmPortS2HOpen (Instance, PLOT_PORT_S2H, &PlotPortS2H);

    for (i = 0; i < NSamples; i++)
    {
        printf ("Input = %d ", Input[i]);
        DsmWrite (FirPortS2H, &Input[i], 1, NULL);
        DsmFlush (FirPortS2H);
        DsmRead (FirPortH2S, &OutSample, 1, &Count);
        Output = (int)OutSample / ScaleFactor;
        printf ("Output = %d\n", Output);
    }
}

```

2. Tutorial implementation

```
#if defined WIN32 || defined __MICROBLAZE__
    if (i != 0)
    {
        SetColor (LIGHTGREEN); /* draw input by green */
        Line (i - 1,
            Input[i - 1] + HEIGHT/2,
            i,
            Input[i] + HEIGHT/2);
        SetColor (LIGHTRED); /* draw output by red */
        Line (i - 1,
            OldOutput + HEIGHT/2,
            i,
            Output + HEIGHT/2);
    }
    OldOutput = Output;
#endif
/* Flush remaining writes */
DsmFlush (PortS2H);
/* Shutdown */
DsmPortS2HClose (FirPortS2H);
DsmPortS2HClose (PortS2H);
DsmPortH2SClose (FirPortH2S);
DsmExit (Instance);
/* Exit cleanly */
return 0;
}
```

The samples of input waveform are defined in the `samples.h` header file. The reading from and writing to ports is done using the `DsmRead()` and `DsmWrite()` functions. The application must be linked with the libraries listed in the following table:

Library name	Description	Platform
<code>libdsmmicroblaze_dma_rc100.a</code>	DSM library for MicroBlaze running on RC200 and using DMA engine.	MB_RC200
<code>libdsmgraphics.a</code>	Simple graphic library for MicroBlaze using DSM interface.	MB_RC200
<code>librcx00microblaze.a</code>	Library for send protocol and standard input from keyboard connected to RC100 or RC200 board.	MB_RC200, MB_RC100
<code>libdsmv2pro.a</code>	DSM library for PPC running in Virtex-II Pro devices.	MV2P
<code>libmv2p.a</code>	Library for standard input using serial port (UART).	MV2P

2. Tutorial implementation

2.2 Running the tutorial in hardware

This section describes how to build the tutorial and run it on a hardware platform.

The example workspace is configured to automatically run Xilinx EDK and Place and Route tools in a custom build step when you target the MicroBlaze processor on the RC200, RC200E or Memec Design platform. You must have the Xilinx software installed for this to work.

2.2.1 Targeting RC200/RC200E

To target the RC200/RC200E you need a parallel cable to download BIT files onto the board and a serial cable to download the software. The BIT file is downloaded using Celoxica's FTU2 utility.

The description below assumes that you are targeting an RC200 board. If you are targeting the RC200E, substitute references to RC200 for RC200E.

Building the hardware side

1. Make sure that the board is connected to your PC with a parallel cable before you build the hardware.
2. Open the DSM Examples Workspace in DK by clicking on **Start>Programs>Celoxica>Platform Developer's Kit>DSM>DSM Examples Workspace [DK]**.
3. Choose the **DsmFIR** project and set it as the active project.
4. Choose the **MB_RC200** platform in **Active Build Configuration**.
5. Click on the build icon, or press F7 to start the compilation.

Building the software side

After the building of the hardware the software side might be already built for you. If not, you can build it as follows:

1. Open the DSM command prompt by clicking on **Start>Programs>Celoxica>Platform Developer's Kit>DSM>DSM Command Prompt**.
2. Change directory to the project, for example: `cd DsmFIR`.
3. Compile and link the source code by running a batch file in the command prompt; type:
`Bui l dSw MB_RC200`
This command will call the compiler that creates executabl e. el f in the DsmFI R\MB_RC200\code folder.

Running the application

1. Make sure that the board is connected to your PC with a serial cable and has a video monitor connected to it.

2. Tutorial implementation

2. To download the executable onto the board, type: **mbconnect MB_RC200**.
This command downloads executable elf onto the board and runs it.

After running the application you should see the same waveforms on the monitor screen as in Figure 1. The values of output samples are sent to XMD console through serial port.

2.2.2 Targeting the Memec Virtex-II Pro (MV2P)

To target the Memec Virtex-II Pro you need a Xilinx JTAG cable to download BIT files and a terminal program such as HyperTerminal for the standard I/O (see section 1.3).

Building the hardware side

1. Make sure that the board is connected to your PC with a parallel JTAG cable before you build the hardware.
2. Open the DSM Examples Workspace in DK by clicking on **Start>Programs>Celoxica>Platform Developer's Kit>DSM>DSM Examples Workspace [DK]**.
3. Choose **DsmFIR** project and set it as the active project.
4. Choose the **MV2P** platform in **Active Build Configuration**.
5. Click on the build icon, or press F7 to start the compilation.

Building the software side

The software is built before generation of the BIT file. You must run the terminal program before the BIT file is downloaded onto the board.

1. Select **Start>Programs>Celoxica>Platform Developer's Kit>PowerPC Hyperterminal**.
2. If you changed the program code and need to recompile it again, you can just hit the build button in DK. Program will be recompiled and downloaded with the bit file onto the board.

Running the application

1. Make sure that the board is connected to your PC with a serial cable.
2. To download the executable onto the board hit the build button or press F7 in DK. This command downloads the BIT file with the new executable elf onto the board and runs it.

After running the application you should see the output values from the FIR filter on the HyperTerminal console.

Running the application using MATLAB for video output

Due to the lack of video output on MV2P target you can use MATLAB to display input and output waveforms. The M-script to run this application is provided in `dsm_fir.m` M-file. This file is in `PDK\instal\Examples\DSM\DsmFIR` folder.

To display the waveforms in MATLAB:

1. Make sure that the board is connected to your PC with a serial cable.

2. Tutorial implementation

2. Run MATLAB.
3. Change the directory in the MATLAB shell to DsmFIR.
Type: `cd PDKInstal/Examples/DSM/DsmFIR`
4. Type: `dsm_fir` and hit enter.
5. Hit the build button or press F7 in DK to download the bit file onto the MV2P board. After downloading the BIT file you should see the waveforms as shown in Figure 4.

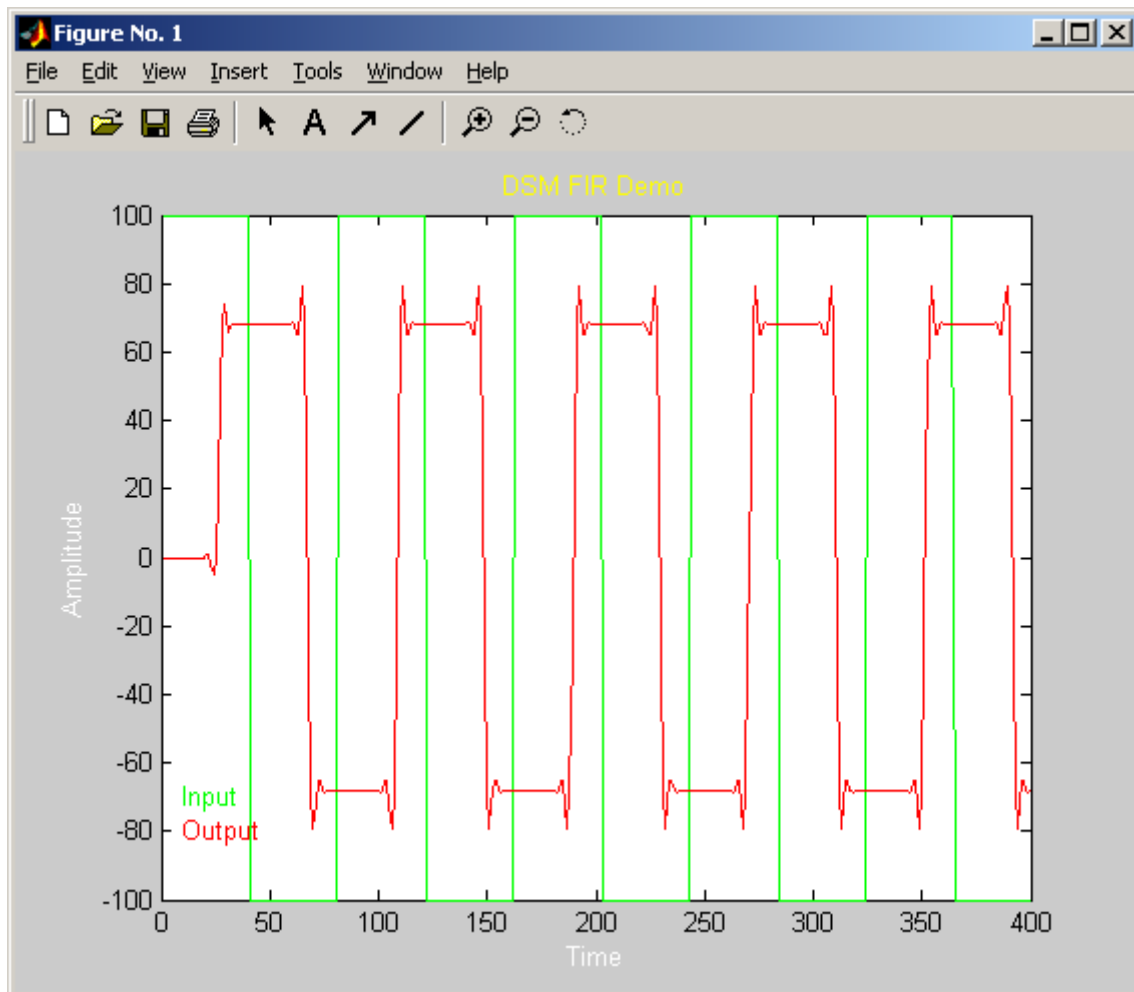


FIGURE 4: MATLAB OUTPUT FOR VIRTEX-II PRO (MV2P) TARGET

2.3 Building a highpass filter

To build a highpass filter you need to define HIGHPASS macro in the `dsm_fir.h` header file, because this file is shared between hardware and software side. To do this, open the header file and uncomment the line `//#define HIGHPASS` at the beginning of the file. Then you need to rebuild the hardware side and software side of the tutorial.

2. Tutorial implementation

To re-build the example, follow the steps in 2.2.

Customer Support at <http://www.celoxica.com/support/>

Celoxica in Europe

T: +44 (0) 1235 863 656

E: sales.emea@celoxica.com

Celoxica in Japan

T: +81 (0) 45 331 0218

E: sales.japan@celoxica.com

Celoxica in Asia Pacific

T: +65 6896 4838

E: sales.apac@celoxica.com

Celoxica in the Americas

T: +1 800 570 7004

E: sales.america@celoxica.com

Copyright © 2003 Celoxica Ltd. All rights reserved. Celoxica, the Celoxica logo and Handel-C are trademarks of Celoxica Limited. All other trademarks acknowledged. The information herein is subject to change and for guidance only

www.celoxica.com

