

Dulse Electronics

**Appnote 4**  
**SEB3 Production Test Code**

Document Number: 8600012  
Version: 1.0  
Date: 14 May 2005

**Copyright 2005 Dulse Electronics**  
*All rights reserved.*

**This page intentionally left blank.**

---

## Table of Contents

<b>1. INTRODUCTION .....</b>	<b>1</b>
<b>2. DESIGN GOALS AND BACKGROUND.....</b>	<b>2</b>
<b>3. DETAILED DESCRIPTION OF TESTS .....</b>	<b>6</b>
3.1 SRAM .....	6
3.2 LEDs .....	6
3.3 7 Segment LEDs.....	7
3.4 LCD interface.....	8
3.5 2 <sup>nd</sup> Oscillator Socket.....	8
3.6 User IO test on JS3 .....	20
3.7 RS232 port 1.....	21
3.8 RS232 port 2.....	21
3.9 SRAM IO connector – IO pins .....	22
3.10 Parallel 3 Circuitry.....	22
3.11 Pushbutton Switches.....	22
3.12 DIPswitch .....	23
<b>4. PICOBLAZE BACKGROUND.....</b>	<b>24</b>
4.1 PicoBlaze Software .....	24
<b>5. CONCLUSION .....</b>	<b>26</b>
<b>6. REFERENCES .....</b>	<b>27</b>
<b>7. REVISION HISTORY.....</b>	<b>28</b>

### ***List of Figures***

**Figure 1 PicoBlaze Circuitry Block Diagram..... 4**  
**Figure 2 Discrete LED Test Circuitry..... 7**  
**Figure 3 7 segment LED Test Circuitry..... 8**  
**Figure 4 SRAM IO connector and 2<sup>nd</sup> Serial Port Circuitry ..... 22**

### ***List of Tables***

**Table 1 SEB3 Elements to be Tested..... 2**  
**Table 2 Production Test SW Commands ..... 25**

## **1. Introduction**

This application note describes the FPGA load that is used for testing the production SEB3s. This FPGA load is installed on the SEB3s when they are shipped to the customer. This appnote, by detailing the tests that are run, will show why some tests are shown to fail when the board is setup with the Quick Start instructions [1] and how to make them pass these tests.

This appnote assumes that the user has read the SEB3 User Manual [1] and has done a previous design with Xilinx FPGAs, such as the 1<sup>st</sup> Simple Application Appnote [2]. Note that this design is considerably more advanced than the 1<sup>st</sup> Simple Application. For more background on using the PicoBlaze in a design, try going through Appnote 2 SEB3 Circuit Cellar Project [3].

This appnote is broken down into two main parts. Part 1 (Chapter 2) will describe the design goals for the project and background information that is useful when doing the design. Part 2 (Chapter 3) describes the elements of the design in more detail. This appnote will NOT describe how to build the application from the ground up (except for the DCM clocking), like the 1<sup>st</sup> Simple Application Appnote. Instead, it will describe important parts of the completed application. Chapter 4 gives a bit of background information on the PicoBlaze processor.

## 2. Design Goals and Background

This intent of this design is to test that the SEB3 has been assembled properly. This means that all the circuitry on the card is exercised, including the 2<sup>nd</sup> oscillator socket and the IO to the breadboard. As with many of the other appnotes from Dulse Electronics, this design uses the PicoBlaze 8 bit Microcontroller and a UART from Xilinx as the basis for the design. Note that not all of the circuitry is tested through the PicoBlaze as will be seen shortly.

Table 1 is a listing showing each of the items on the SEB3 and how it is tested.

*Table 1 SEB3 Elements to be Tested*

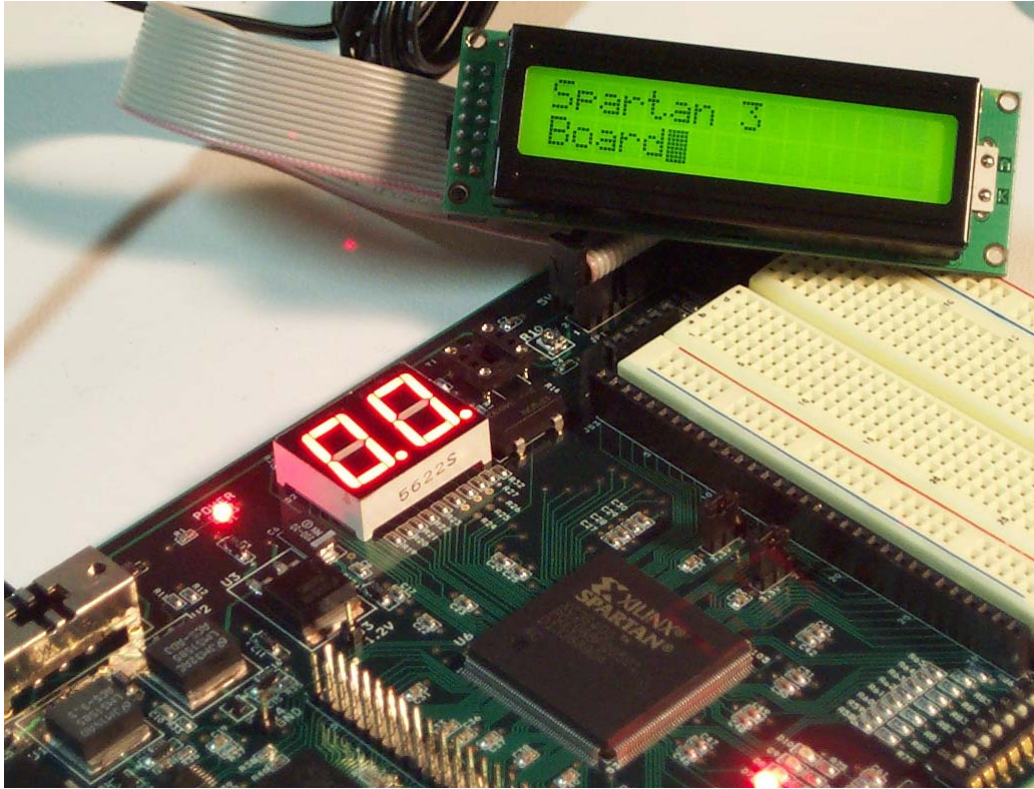
Item	Design Element	How is it tested?
1	Voltage Regulators	Digital Multimeter reading
2	SRAM	Tested via PicoBlaze
3	LEDs	Tested via an 8 bit shift register that shifts a walking 1 pattern left or right depending on the state of Switch SW2
4	7 Segment LEDs	Tested by showing either (i) the setting of the DIP switch (SW4) or (ii) the output of an internal 4 bit counter. The selection is accomplished by Switch SW3.
5	LCD interface	Tested via PicoBlaze. A simple message is displayed on the LCD. If the message appears, then the interface is working.
6	2 <sup>nd</sup> Oscillator Socket	Tested by putting a 60 MHz oscillator in the 2 <sup>nd</sup> oscillator socket and switching the PicoBlaze clock from the 50MHz soldered in oscillator to the 60MHz oscillator. The PicoBlaze does this through program control.  Note: the 60 MHz clock goes through a DCM, which brings the 60MHz clock down to 50MHz. This will be explained

		in more detail later.
7	User IO test on JS3	A loopback connector is placed on JS3 and then test patterns are written to the IO connected to JS3 and then read back on the shorted pin.
8	SRAM IO connector – IO pins	A loopback connector is placed on some of the pins on JP4. Test patterns are written to the IO connected to JP4 and then read back.
9	RS232 – port 1	This is tested just by being used as the main communications from the PC to the PicoBlaze.
10	RS232 - port 2	Tested by shorting TX2 to RX2 and sending a test pattern through the level shifter and back in.
11	Parallel 3 Circuitry	Tested by the actual programming of the SEB3.
12	Pushbutton Switches	Tested by the tests for the LEDs mentioned above.
13	DIP switch	Tested by the tests for the 7 segment LED mentioned above.

Figure 1 shows the circuitry in and around the FPGA used to test most of the elements of the SEB3. Figure 2 shows the circuitry to test the discrete LEDs. Figure 3 shows the circuitry used to test the 7 segment LEDs.



Photo 1 shows the production test circuitry running on the SEB3, with an output on the LCD display.



***Photo 1 SEB3 running with PicoBlaze driving LCD and LEDs.***

Note that the backlight on the production LCD is always on and just requires the standard 14-pin LCD cable.

## **3. Detailed Description of Tests**

This section will go over the production tests in more detail.

### **3.1 SRAM**

The SRAM is tested by SW code in the PicoBlaze. The HW control for the SRAM consists of 4 output registers: Top\_Addr\_port, Hi\_Addr\_port, Lo\_Addr\_port and Data. As well, the PicoBlaze needs to be able to read the content of the SRAM, hence the SRAM data lines go into the input mux for the PicoBlaze.

Essentially, what the test does is loop through all the SRAM address range (by writing to the Addr\_port registers) and outputting an incrementing data value. Note that the address range (128KB or 17 bits) is much larger than the data range (8 bits), therefore there will be repetition in the patterns put in the SRAM. The test starts by writing to all of SRAM with a pattern of data. It starts at data = 00h and counts up to FFh and then goes to 00h, then 00h again and then counts up again. This exercises the address and the data lines. See the SEB3\_ROM.psm file for details of the algorithm.

This pattern is then read back. If the values do not match, then there is an error.

Note that the code only indicates that there is an error, it doesn't tell you what the error is. This, as some professors would say, is an exercise for the reader!

### **3.2 LEDs**

The discrete LEDs are tested via an 8-bit shift register that shifts a walking 1 pattern left or right depending on the state of Switch SW2. This is shown in Figure 2. When SW2 is not pressed, a walking one pattern goes to the left and wraps back to the right. When SW2 is pressed, the pattern goes to the right.

The clock used to drive the 8-bit bi-directional shift register is labeled "slow\_clock". This clock is derived from a 32-bit counter running at 50MHz. The 23rd bit is used for slow\_clock. This works out to an approximately 3 Hz clock. This is slow enough for the human eye to see, but fast enough to ensure the test doesn't take too long.



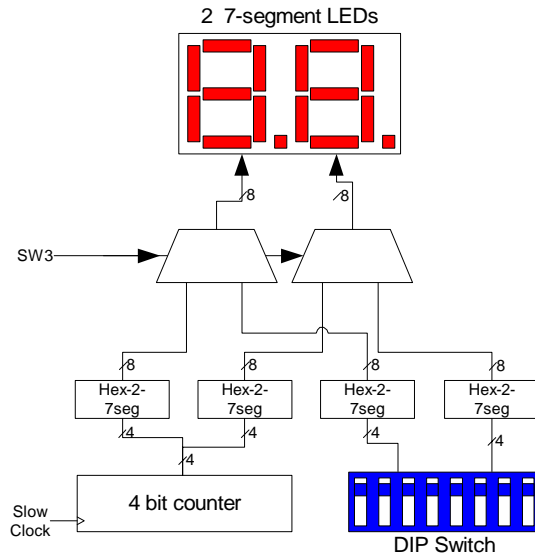


Figure 3 7 segment LED Test Circuitry

### 3.4 LCD interface

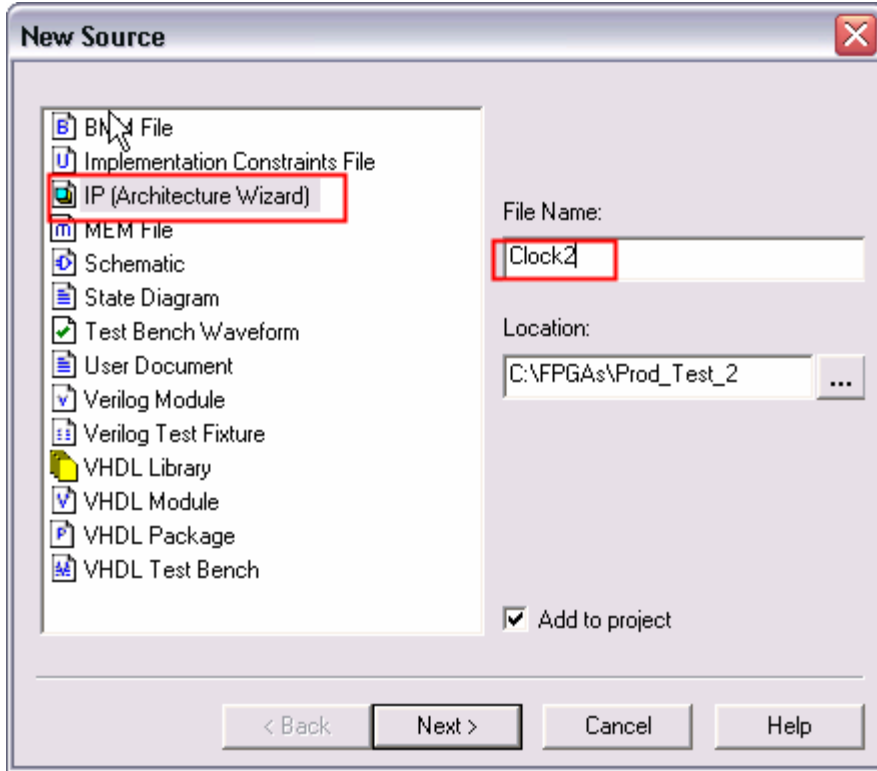
The LCD interface is tested by sending an output to the LCD. If the display shows the correct output, then the interface is deemed to be working. (Note that the LCD contrast potentiometer, R10, may need to be adjusted to get any output at all!)

The LCD interface test requires 2 output registers: LCD control and LCD data. These registers directly drive the LCD control pins and data pins respectively. The LCD data lines are also tri-stateable as the data lines also need to be readable. The VHDL code for the output registers are found in lines 542 to 595 of the SEB3\_top.vhd file. The code for the LCD data input (which is just fed through a big multiplexer is seen in lines 380 to 381.

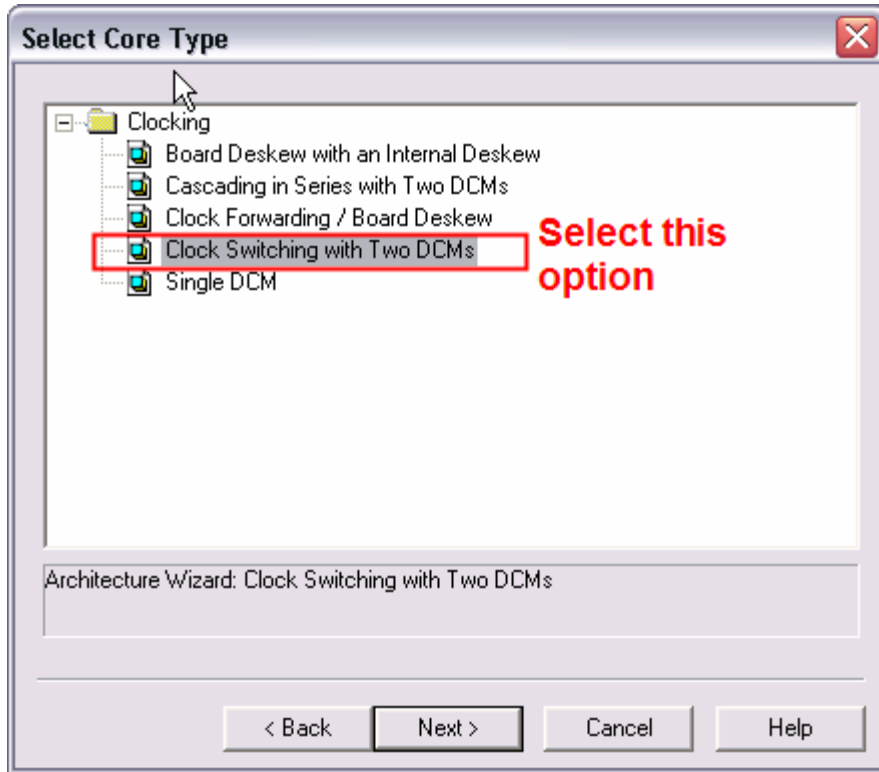
### 3.5 2<sup>nd</sup> Oscillator Socket

To test the 2<sup>nd</sup> oscillator socket, we put a 60 MHz oscillator in the 2<sup>nd</sup> oscillator socket and switch the PicoBlaze clock from the 50MHz soldered in oscillator to the 60MHz oscillator. The PicoBlaze does this through program control. The user is first asked if there is a 60MHz oscillator installed. Then the program switches the output of the multiplexer.

Since the on-board oscillator is 50MHz and the 2nd oscillator is 60MHz and all timing, (including that of the serial port) is driven off of the clock, it is necessary to convert the 60MHz clock down to 50MHz. This is done by using the clocking architecture wizard (through the **Project -> New Source...** menu seen below.)

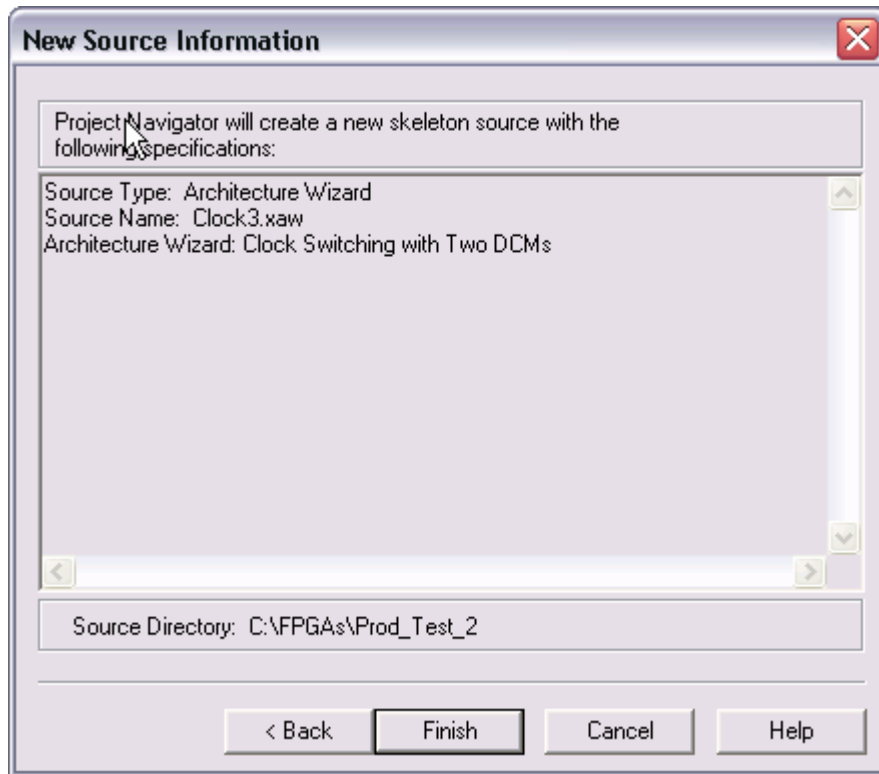


Click on **Next>**



Click on **Clock Switching with Two DCMs**

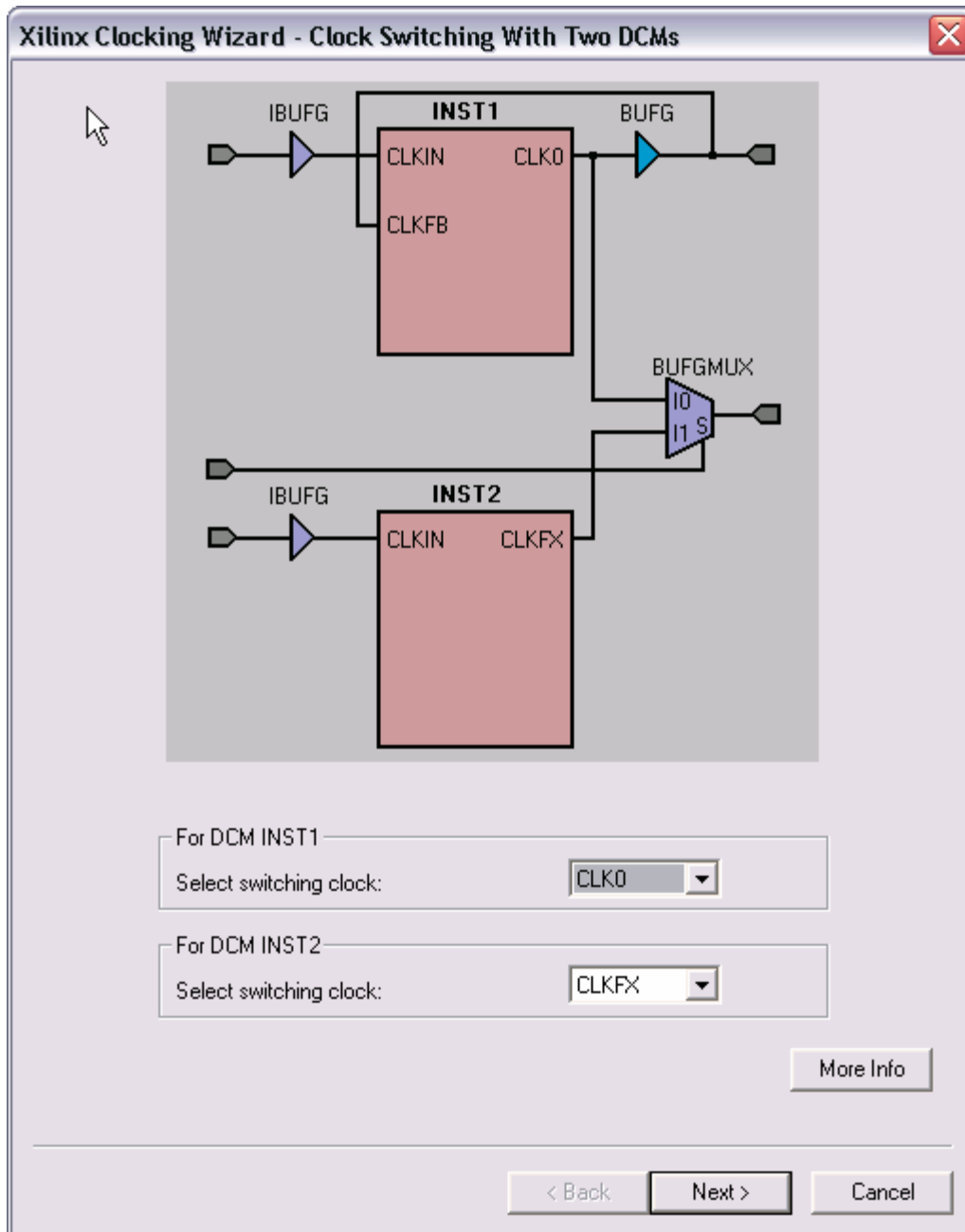
Click on **Next>**



This gives a summary of the information (note that it should be Clock2.xaw!).

Click on **Finish**.

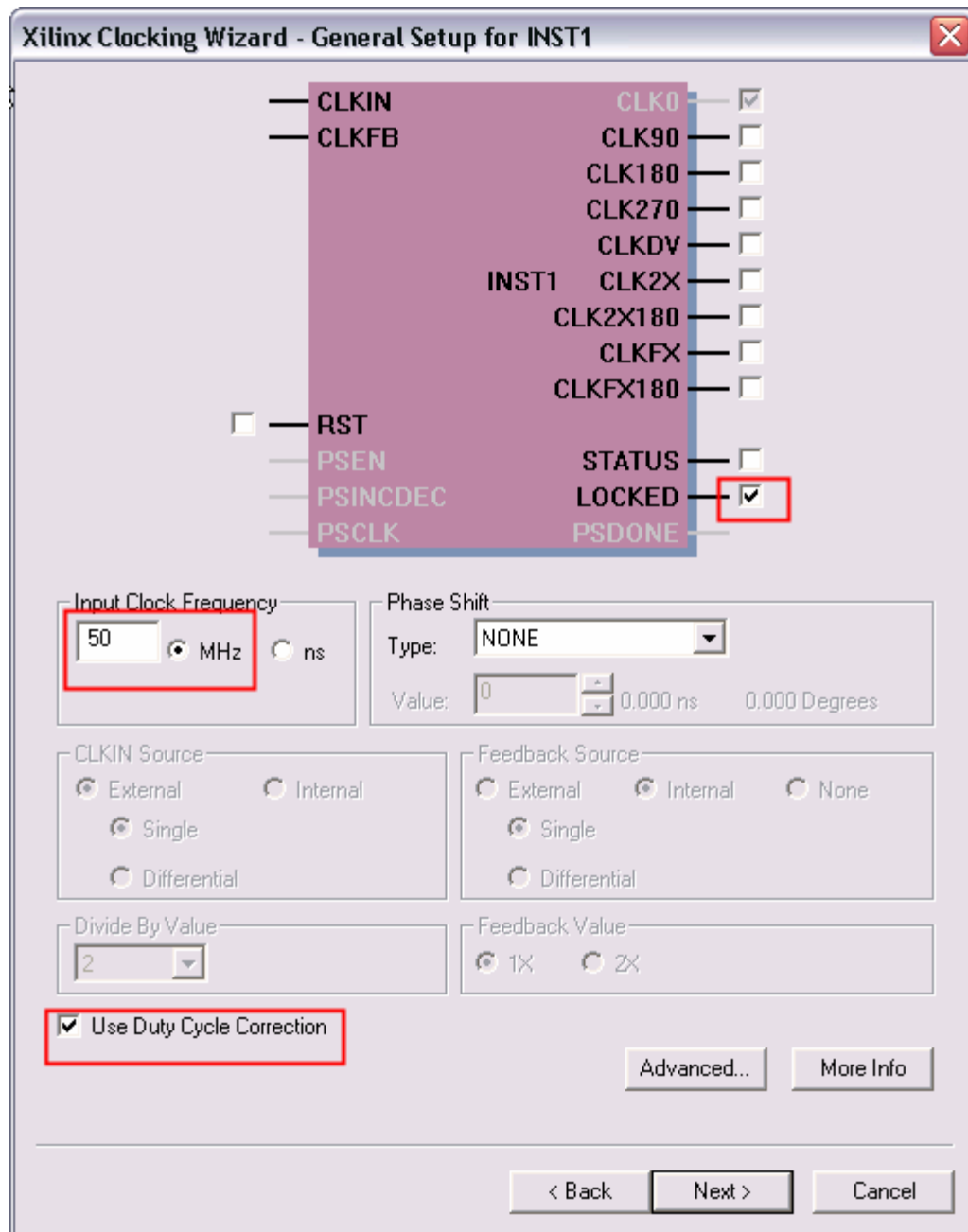
This brings up the following dialogs which customize the “macro”. These dialogs can also be accessed, in the finished design, by double-clicking on the **Clock2.xaw** source in the Sources window.



So, DCM INST1 we will use the 50MHz clock, so the output of INST1 is just the clock output (i.e. **CLK0**).

For DCM INST2, we need to use the DCM to change the 60MHz input clock to 50MHz. This is done by selecting the **CLKFX** output, which will be explained a bit more later.

Click on **Next>**



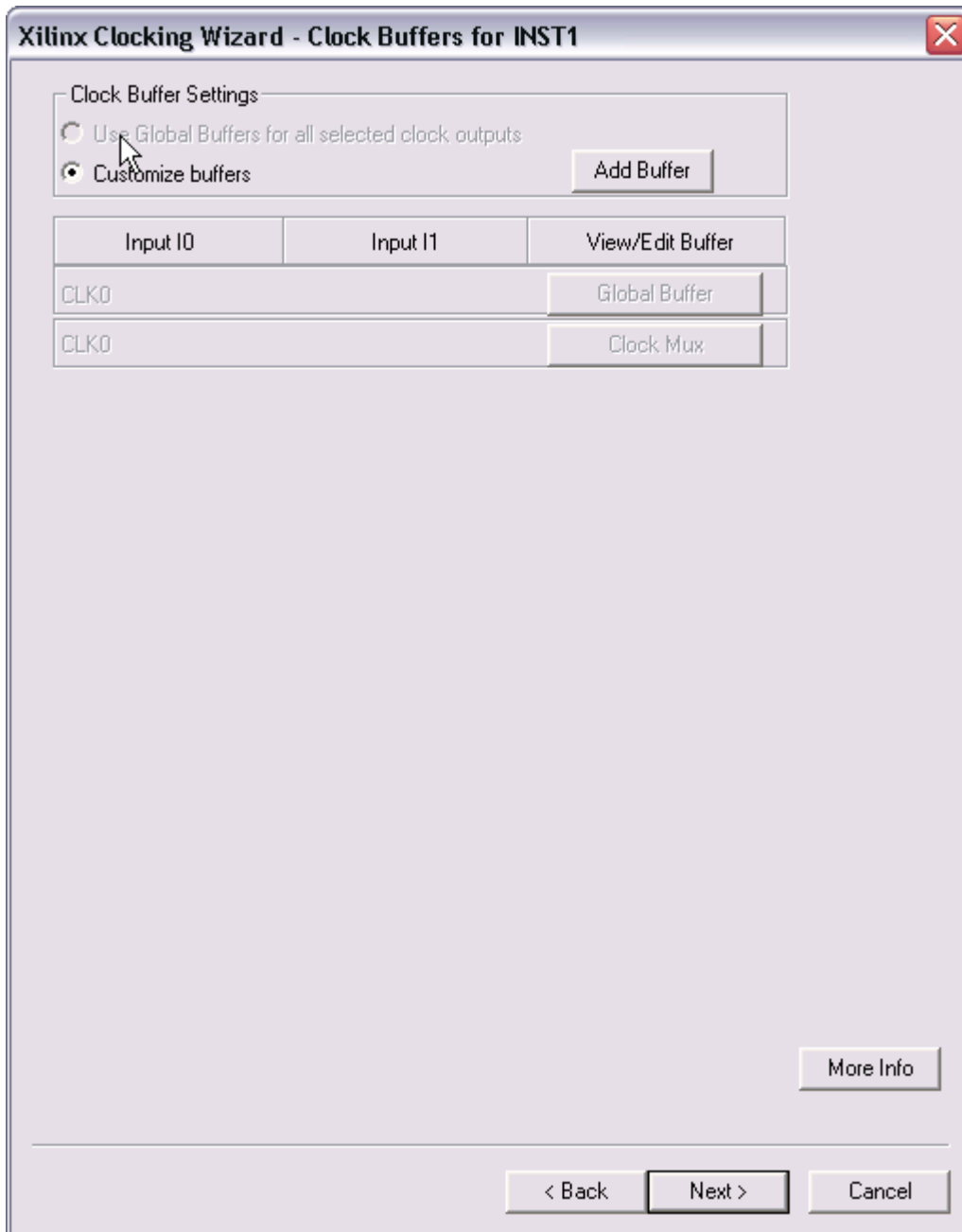
Note that this dialog is for DCM INST1.

Set the Input Clock Frequency to **50 MHz**.

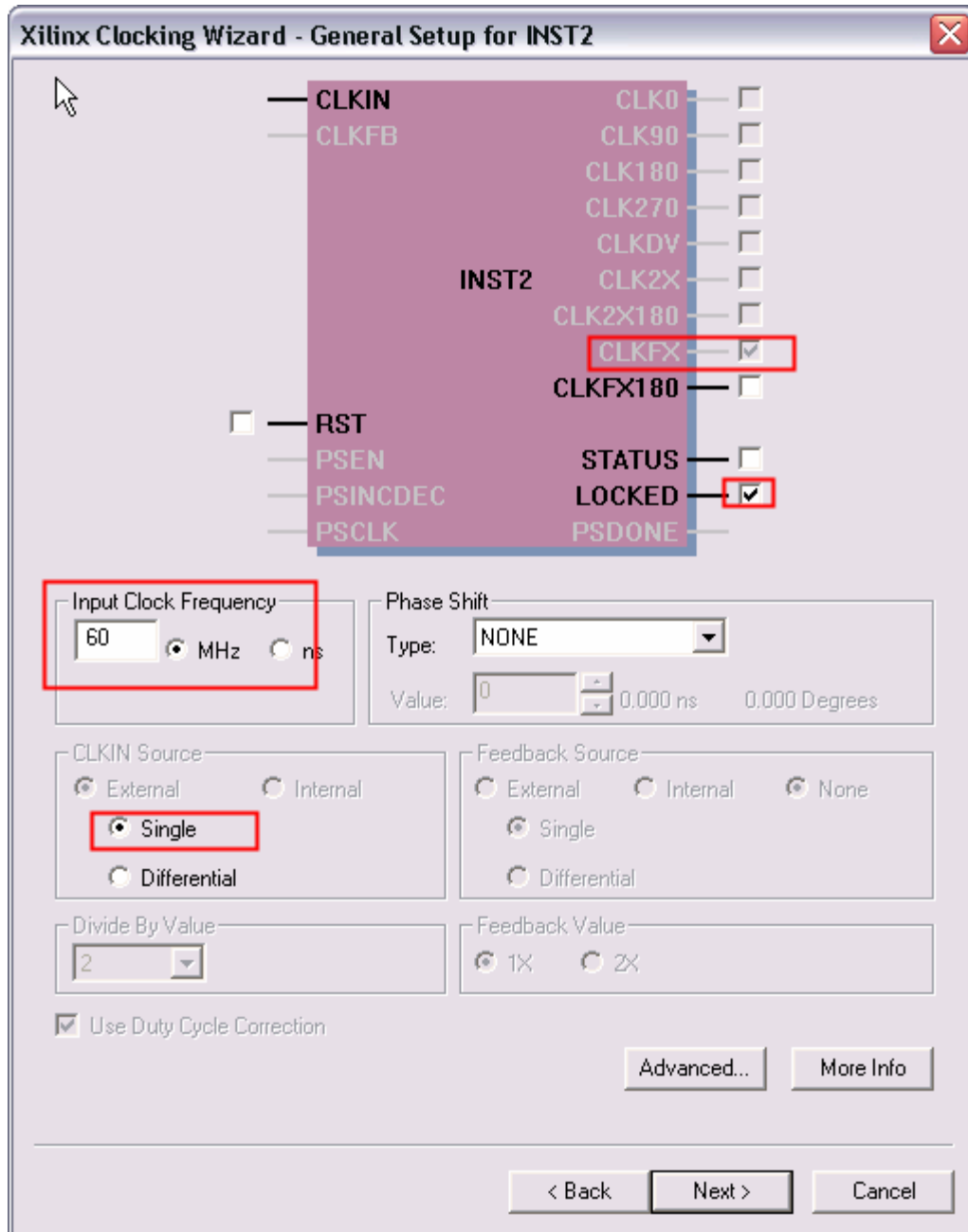
Enable the **Use Duty Cycle Correction**.

Enable the **Locked** pin.

Click on **Next>**



Click on **Next>**

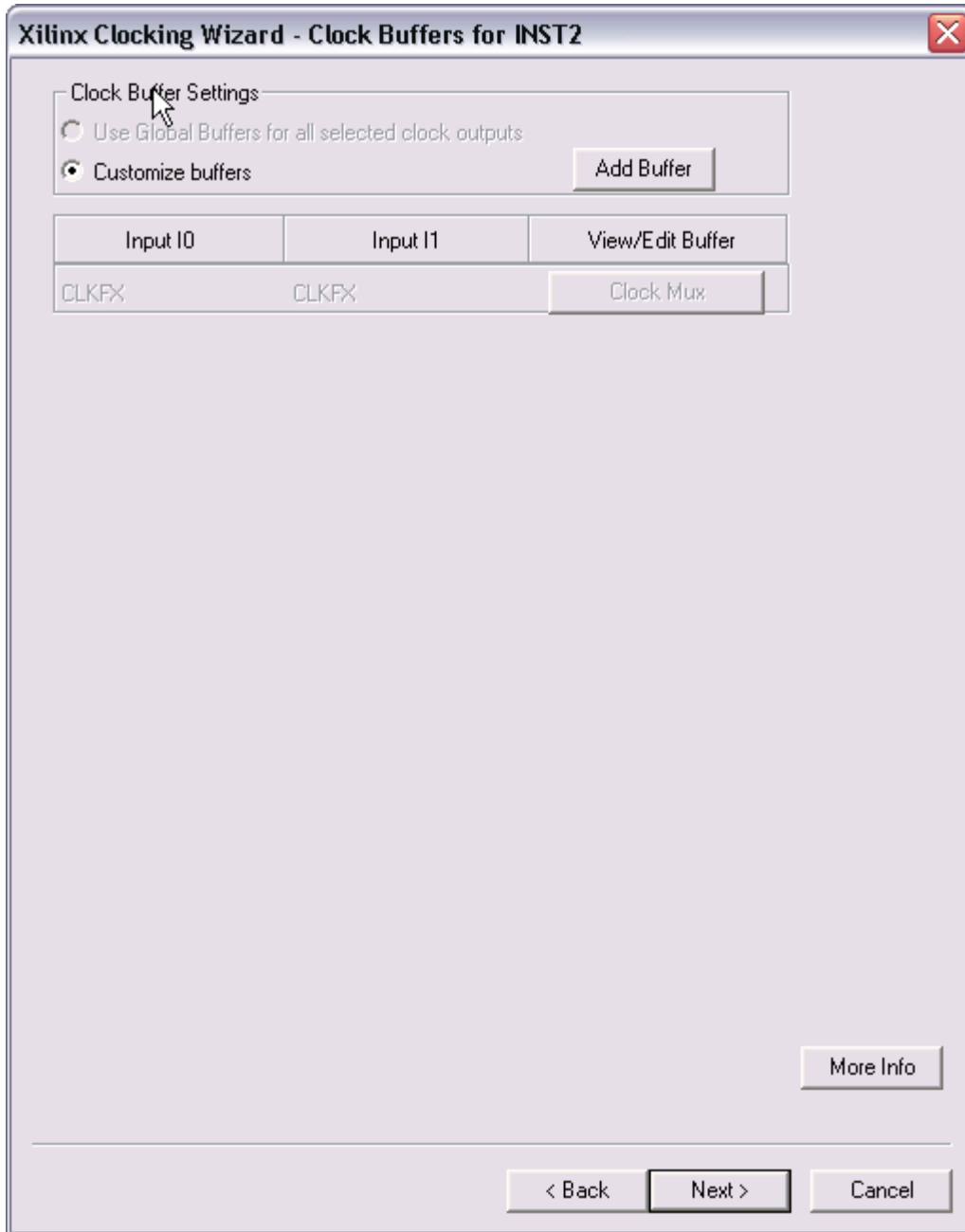


Note that the CLKFX output is already set.

Set the Input Clock Frequency to **60 MHz**.

Enable the **Locked** pin.

Click on **Next>**



Click on **Next**>

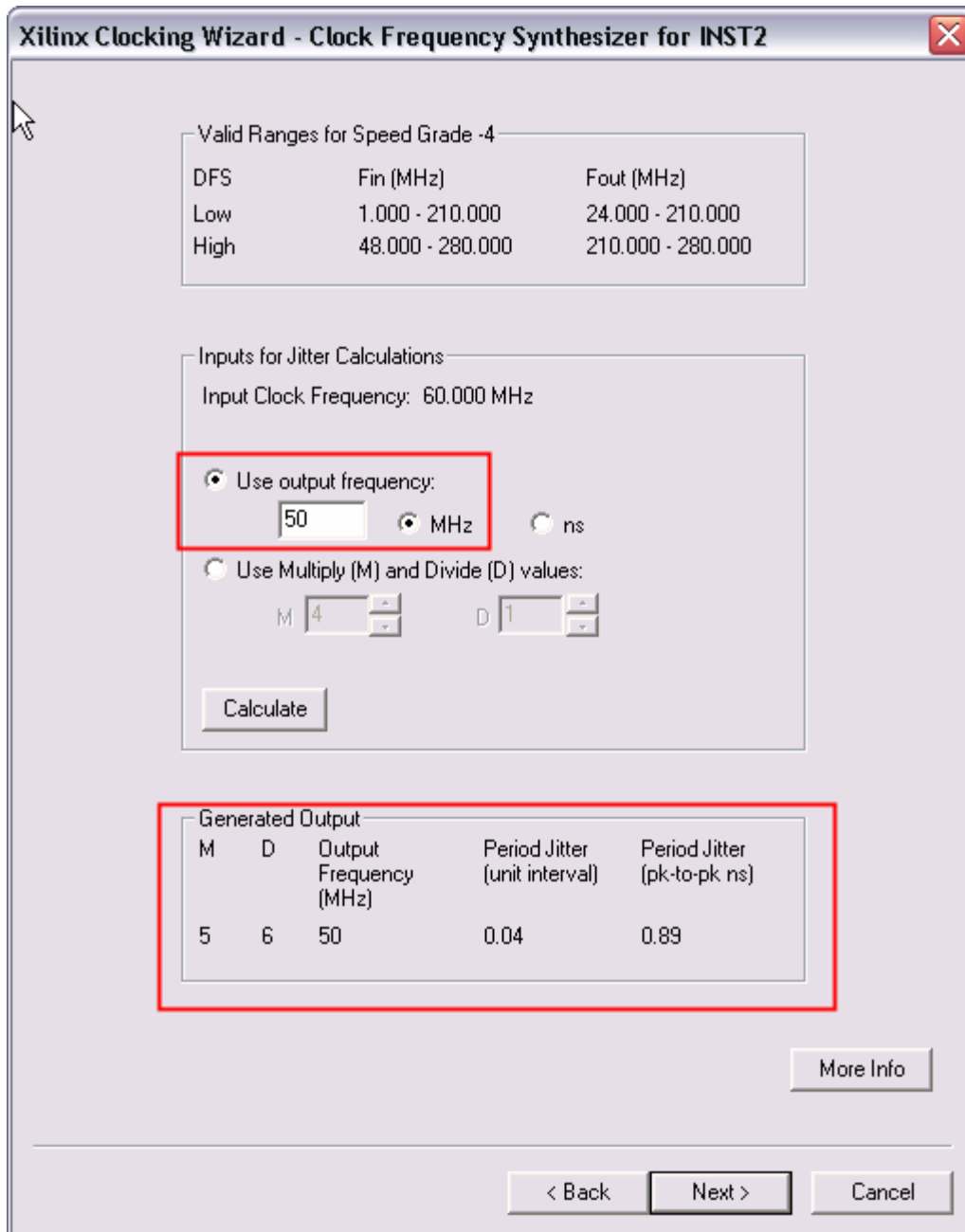
**CLKFX output**

CLKFX is the output of the Digital Clock Manager's Digital Frequency Synthesizer. It is determined by the input clock and two parameters (M and D). The CLKFX output frequency is given by the following formula.

$$\text{CLKFX} = \text{clock in} * M / D$$

where M is an integer from 2 to 32 and D is an integer from 1 to 32.

For this application, clock\_in is 60 MHz and we want CLKFX to be 50 MHz. M = 5 and D = 6 will give the desired CLKFX. These settings are shown in the next dialog. Note that the user can enter the M and D values directly or merely specify the final CLKFX frequency and let the tools calculate M and D.

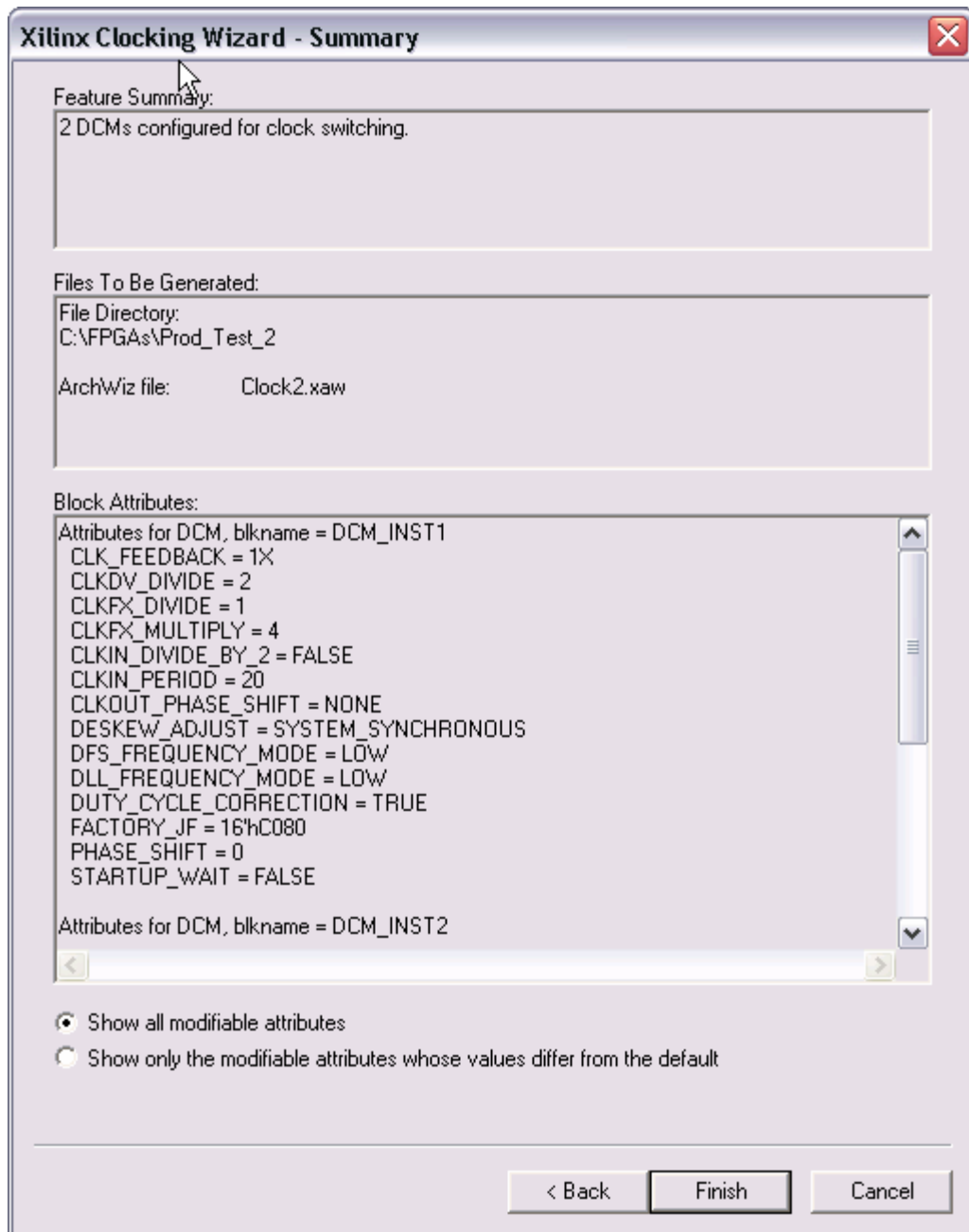


This dialog allows the user to indicate the output frequency of DCM INST2.

Set the output frequency to **50 MHz**.

When this is done, you will see that it automatically calculates the divider numbers for the DCM CLKFX output and gives you the jitter numbers.

Click on **Next>**



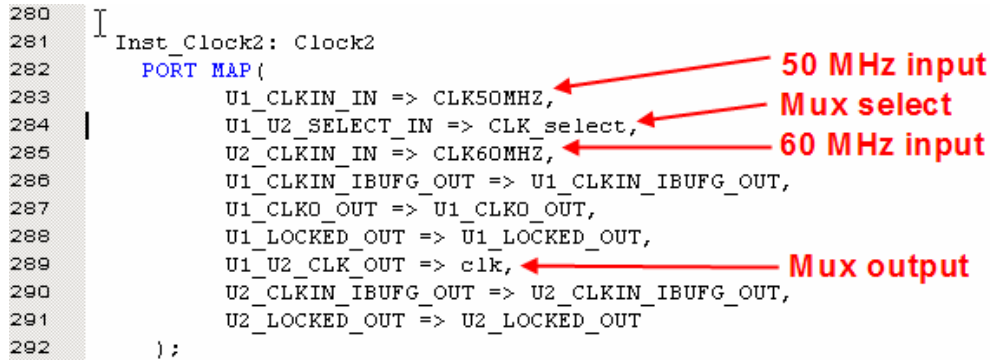
This shows all the attributes that have been setup for the 2 DCMs and the switch.

Click on **Finish**.

The two clocks are now setup.

This “macro” is instantiated in the top level seb3\_top VHDL file as shown next.

```
280
281 Inst_Clock2: Clock2
282   PORT MAP(
283     U1_CLKIN_IN => CLK50MHZ,
284     U1_U2_SELECT_IN => CLK_select,
285     U2_CLKIN_IN => CLK60MHZ,
286     U1_CLKIN_IBUFG_OUT => U1_CLKIN_IBUFG_OUT,
287     U1_CLKO_OUT => U1_CLKO_OUT,
288     U1_LOCKED_OUT => U1_LOCKED_OUT,
289     U1_U2_CLK_OUT => clk,
290     U2_CLKIN_IBUFG_OUT => U2_CLKIN_IBUFG_OUT,
291     U2_LOCKED_OUT => U2_LOCKED_OUT
292   );
```

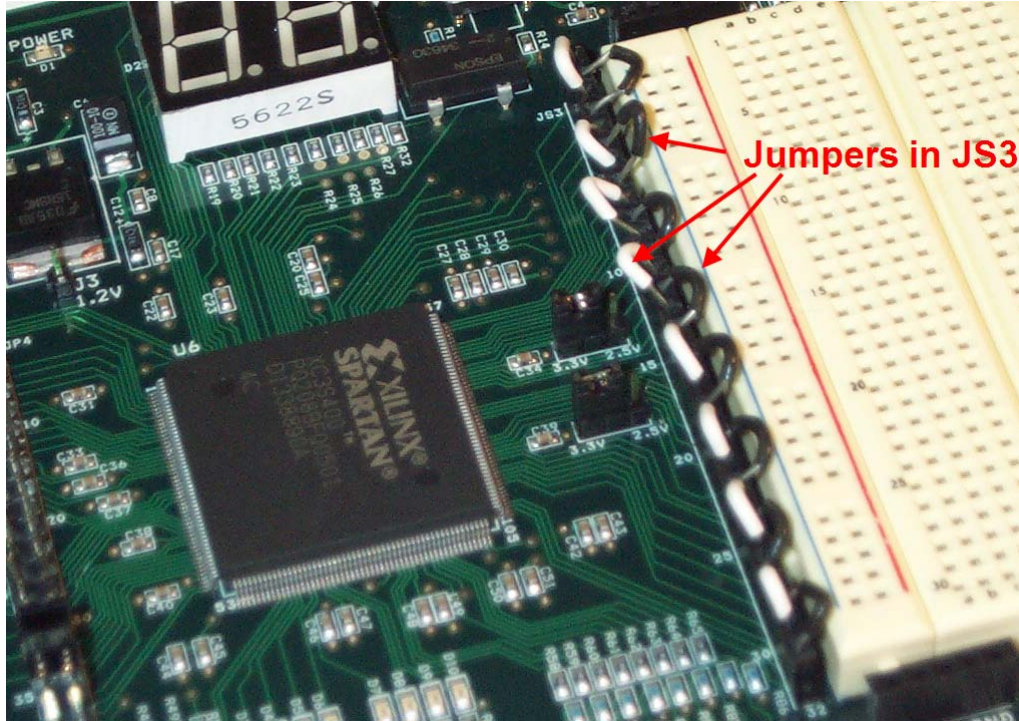


The CLK\_select signal comes from a PicoBlaze output register specifically for switching the clocks (see lines 662 to 682 in the SEB3\_top.vhd file).

### 3.6 User IO test on JS3

To test that the FPGA is connected properly to the User IO connector JS3, a loopback connector is placed on JS3 and then “walking 1” test patterns, followed by “walking 0” test patterns are written to the IO connected to JS3 and then read back on the corresponding shorted pin. Photo 2 shows some jumpers inserted in JS3 to emulate the loopback connector used in production.

The pattern to the loopback is as follows: pin 1 is shorted to pin 3, pin 2 is shorted to pin 4.... This pattern is repeated for the entire connector. In Photo 2, white jumper wires are used for the odd pins and black jumper wires are used for the even pins.



*Photo 2 Jumpers on User IO connector JS3*

### **3.7 RS232 port 1**

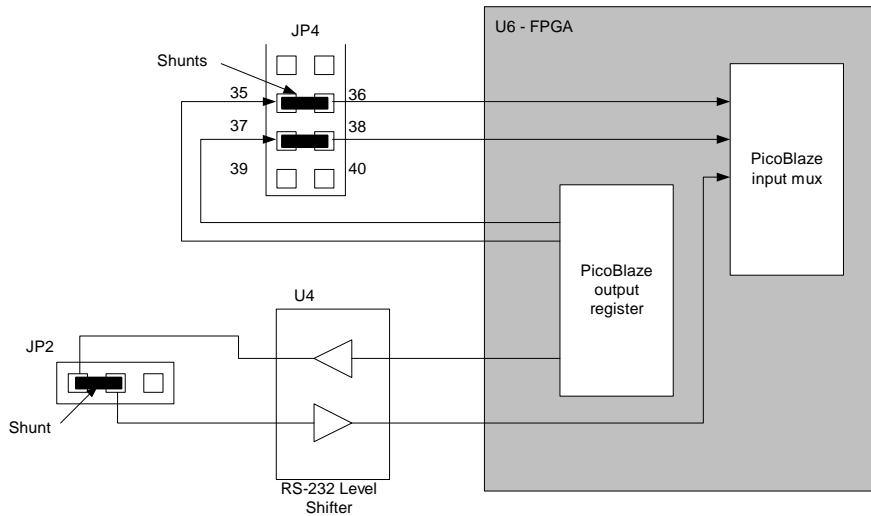
The first RS232 port is used to communicate from the PC to the PicoBlaze. Hence, this is tested just by running the tests. If there is an output on Hyperterminal (set to 8N1, no flow control), then port 1 works.

### **3.8 RS232 port 2**

RS232 Port 2 is actually tested along with the SRAM IO pins and will be detailed in the next section. Essentially, the 2<sup>nd</sup> RS232 port is tested by shorting TX2 to RX2 and sending a test pattern through the level shifter and back in.

### 3.9 SRAM IO connector – IO pins

The SRAM connector (JP4) on the SEB3 has 4 extra IO lines (pins 35 to 38) available from the FPGA. These are tested by having 2 of them, pins 35 and 37, connected to the FPGA as outputs and having the other 2 pins, 36 and 38 connected as inputs. These pins are then shorted together with shunts and the PicoBlaze writes a pattern to the output pins and reads them on the inputs. If there are no errors, then these pins work properly. The same thing is done, at the same time, with the 2<sup>nd</sup> serial port as can be seen in Figure 4. Note that there is a timing delay added in the PicoBlaze SW code to account for the delay through U4, the RS-232 level shifter.



**Figure 4 SRAM IO connector and 2<sup>nd</sup> Serial Port Circuitry**

Now you can see why the SEB3 production test shows errors when the Quick Start instructions are followed. To pass the SRAM IO tests, the three shunts shown in Figure 4 need to be added. To pass the IO tests, the loopback connections as shown in Photo 2 (or equivalent) needs to be installed.

### 3.10 Parallel 3 Circuitry

The Parallel 3 Circuitry is tested by programming the SEB3 PROM during the production test. If programming passes, then the circuitry is ok.

### 3.11 Pushbutton Switches

The pushbutton switches (SW2 and SW3) are tested by the tests for the LEDs and 7 segment LED mentioned in sections 3.2 and 3.3

### **3.12 DIPswitch**

The DIPswitch (SW4) is tested by the tests for the 7 segment LED mentioned in section 3.3.

## **4. PicoBlaze Background**

The PicoBlaze is a simple and very small 8-bit processor for the Xilinx FPGAs. KCPSM3 is the version for the Spartan 3 FPGAs. The program code space for the PicoBlaze is a single Block RAM (BRAM) organized as 1K instructions by 18 bits. The PicoBlaze was developed to be as small as possible and consumes only 96 slices (or 2.7% of a XC3S400) and 1 BRAM. The PicoBlaze executes an instruction in 2 clock cycles and runs at ~87MHz in a -4 speed grade part. This works out to ~43 MIPS. Note that this is faster than many 8 bit controllers on the market today and you can have multiple PicoBlazes in a single FPGA! The main application for a PicoBlaze is for relatively low speed control. It can be used instead of a state machine where the state machine design may be complicated or the high speed is not needed. In this application, the PicoBlaze runs at 50MHz (25 MIPS.)

The PicoBlaze has 16 general purpose registers, a 64 byte scratchpad memory, can handle interrupts and can call subroutines up to 31 levels deep. It has the capability of interfacing to 256 input and 256 output ports.

Refer to the PicoBlaze User Guide [4] for more information.

Designing with the PicoBlaze means instantiating it in the design using VHDL in this case. An external assembler is then used to generate the code that is downloaded to the BRAM. One of the outputs of the assembler is a VHDL file that defines the BRAM and it's contents. This is added as a source to the project and that's all there is to it.

The PicoBlaze comes with an optional UART, which handles 8 bits, no parity, 1 stop bit and has a 16 byte FIFO. This UART occupies 40 slices (~1% of XC3S400) and can be used at regular RS-232 speeds such as 38,400 baud or it can be used at rates exceeding 10 Mbaud for inter-FPGA communications.

### **4.1 PicoBlaze Software**

Programming for the PicoBlaze is done in assembler only. Remember that there are only 1K of instructions. This appnote uses the standard assembler that comes with the PicoBlaze core. However, the code could be converted to the Mediatronix's PicoBlaze IDE [5] if one wishes.

Table 2 shows the commands that have been implemented in the PicoBlaze code for the production test code. They are similar to what you might find in a simple general-purpose monitor. The design started with the UART real-time clock reference code that Xilinx provides with the PicoBlaze. It contains routines for handling the UART. This code was used as a starting point, it was stripped of the real-time clock routines and then the code to handle the SRAM, IO, 2<sup>nd</sup> serial port and the LCD was added.

**Table 2 Production Test SW Commands**

<b>Command</b>	<b>Description</b>	<b>Syntax</b>
RD	Read byte from SRAM	RD hhhh <sup>1</sup> where hhhh is the address in the SRAM
WR	Write byte to SRAM	WR hhhh,xx writes xx(in hex) to address hhhh
BD	Block dump from SRAM, returns 256 byte block	BD hh where hh is the upper byte of the address of the 256 byte block in the SRAM
FILL	Fill 256 bytes of SRAM with a value.	FILL hh,xx, where hh is the upper byte of the of the 256 byte block in the SRAM and xx is the data
LCD	Outputs string (16 chars max) on line of LCD display	LCD In,string, where In is either line 1 or 2 and the string is up to a 16 character string ending with a carriage return.
TEST	This command tests the User IO and the SRAM IO (including the 2 <sup>nd</sup> serial port). Note that the tests will only pass if the appropriate loopbacks and shunts are installed.	TEST

## **5. Conclusion**

This appnote has shown the design goals for the SEB3 production tests and has explained how each of the components on the SEB3 has been tested.

The production test code for the SEB3-400 exercises all the components on the SEB3 as well as the external LCD. However, unless the loopbacks and shunts are installed in JS3, JP4 and JP2, the IO tests and SRAM IO tests will show as FAILing when the SEB3 is turned on.

The code and circuitry is available on the Dulse Electronics website and the reader is encouraged to look over the design and use it for the basis for their own design.

Have fun!

## 6. References

- [1] SEB3 User Manual, Dulse Electronics, [www.dulseelectronics.com](http://www.dulseelectronics.com)
- [2] SEB3 Simple First Application, Dulse Electronics, [www.dulseelectronics.com](http://www.dulseelectronics.com)
- [3] Appnote 2 SEB3 Circuit Cellar Project, Dulse Electronics, [www.dulseelectronics.com](http://www.dulseelectronics.com)
- [4] PicoBlaze 8-bit Embedded Microcontroller User Guide for Spartan-3, Virtex-II, and Virtex-II Pro FPGAs (ug129), Xilinx, [www.xilinx.com](http://www.xilinx.com)
- [5] Mediatronix's PicoBlaze IDE, Mediatronix BV, [www.mediatronix.com](http://www.mediatronix.com)

## **7. Revision History**

<b>Date</b>	<b>Version Number</b>	<b>Description</b>
2005/05/14	1.0	Initial Release