XAPP1005 (v1.1) October 3, 2007

# Using Clocking Resources on XtremeDSP Development Kits

Author: Jacobus Naude

## Summary

This application note describes the steps for using the different clocking resources on the XtremeDSP™ Development Kits developed by Nallatech. In addition to a detailed description of the clock resources, the clocking schemes used for Hardware Cosimulation with System Generator for DSP are explained and clocking suggestions for using the board as a stand-alone board are given.

This application note includes five design examples with accompanying reference designs, (that is, System Generator for DSP models and HDL design files), demonstrating how to use different combinations of clocks, as well as different ways to generate the target designs (using System Generator, ISE™ software, or a combination of both).

After reading this document, users will know:

- How to use the different clocking resources on the different XtremeDSP Development Kits
- How System Generator for DSP configures the clocking on these kits during Hardware Cosimulation (HW-Cosimulation)
- How to use the kits' clocking resources outside the System Generator for DSP environment.

## Introduction

This application note can be applied to four XtremeDSP development kits listed:

- XtremeDSP Development Kit for Virtex™-II FPGAs (XC2V2000)
- XtremeDSP Development Kit-II for Virtex-II FPGAs (XC2V3000)
- XtremeDSP Development Kit-II Pro for Virtex-II Pro FPGAs (XC2VP30)
- XtremeDSP Development Kit-IV for Virtex-4 FPGAs (XC4VSX35)

For the purposes of this document, the following differences between the four mentioned kits regarding the clocking structure are important:

- The onboard oscillators do not run at the same frequencies.
- The main FPGA pinouts differ for each board.
- The provided `osc_clock_2v80.bit` and `ext_clock_2v80.bit` files route the output clock to a different pin on the XtremeDSP Development Kit for Virtex-4 FPGAs.

The pin locations used in this document are based on the XtremeDSP Development Kit-IV for Virtex-4 FPGAs, as found in the right-most column of Table 1. Using the numbers shown in Figure 1, it is possible to apply this application note to the four XtremeDSP Development Kits previously mentioned.

Looking at Table 1, the clock FPGA pinout stays the same for all four kits, while the main FPGAs pinout is different for each kit. The examples in the "Examples Using Board Clocking Resources" section of this document can be applied to all four of these kits using Table 1. However, the source files are only provided for the XtremeDSP Development Kit for Virtex-II FPGAs (XC2V3000) and the XtremeDSP Development Kit for Virtex-4 FPGAs.

© 2007 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.



www.BDTIC.com/XILINX

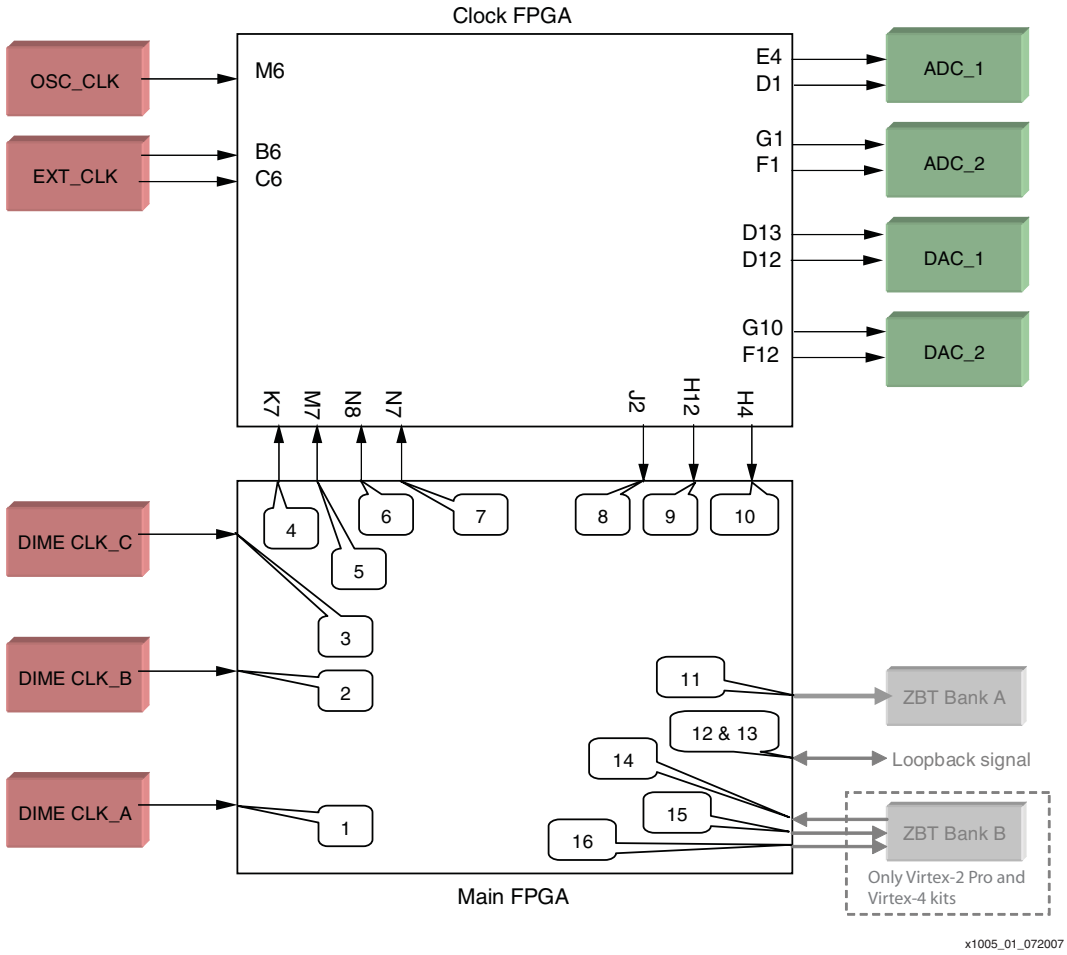


*Figure 1:* **Number Diagram Correlating to all Four XtremeDSP Development Kits**

*Table 1:* **Equivalent Pin Locations to Differentiate Between the Development Kits**

| Number from Figure 1 | Associated Net Name | XC2V2000 -4FG676 | XC2V3000 -4FG676 | XC2VP30 -5FF1152 | XC4VSX35 -10FF668 |
|---|---|---|---|---|---|
| 1 | CLKA | B11 | D13 | H17 | AF12 |
| 2 | CLKB | V11 | Y13 | AJ17 | A16 |
| 3 | CLKC | Y11 | AB13 | J18 | AF11 |
| 4 | GEN_CLKA | Y10 | AB12 | AF18 | B13 |
| 5 | GEN_CLKB | V10 | Y12 | AK19 | C15 |
| 6 | GEN_CLKC | AA10 | AC12 | AG18 | B12 |
| 7 | GEN_CLKD | W10 | AA12 | AL19 | C14 |
| 8 | CLK1_FB | Y12 | AB14 | AH18 | B14(1) |
| 9 | CLK2_FB | A11 | AC13 | AH17 | A15 |
| 10 | CLK3_FB | W12 | AA14 | AJ18 | B15 |
| 11 | ZBT_CLK | A13 | A13 | G18 | AB10 |
| 12(2) | ZBT_FB_OUT | A12 | A12 | F18 | AC10 |
| 13(2) | ZBT_FB_IN | C13 | C13 | H18 | AE12 |
| 14 | ZBTB_CLK | NA | NA | F17 | AE14 |


www.BDTIC.com/XILINX

*Table 1:* **Equivalent Pin Locations to Differentiate Between the Development Kits**

| Number from Figure 1 | Associated Net Name | XC2V2000 -4FG676 | XC2V3000 -4FG676 | XC2VP30 -5FF1152 | XC4VSX35 -10FF668 |
|---|---|---|---|---|---|
| 15 | ZBTB_CLK_FB_OUT | NA | NA | G17 | AB17 |
| 16 | ZBTB_CLK_FB_IN | NA | NA | D17 | AC17 |

***Note:***

1. Pin B14 (number 8 in Figure 1) is not an optimal clock IOB site on the XC4VSX35 -10 FF668 which is found on the XtremeDSP Development Kit for Virtex-4 FPGAs. To use this pin location, the user should set the XIL_PLACE_ALLOW_LOCAL_BUFG_ROUTING environment variable to 1 to avoid errors during implementation. It should be noted that this will cause a drop in performance.
2. In Figure 1, the loopback signal is shown as a line with arrows on both sides. This represents a loopback signal and does not represent a bidirectional signal, thus, the reason it has two pins associated with it.

## Development Kit Clocking Structure

The clocking structure of the development kits is shown in Figure 2. Clock sources are shown in pale red blocks and clock sinks are shown in pale green blocks.

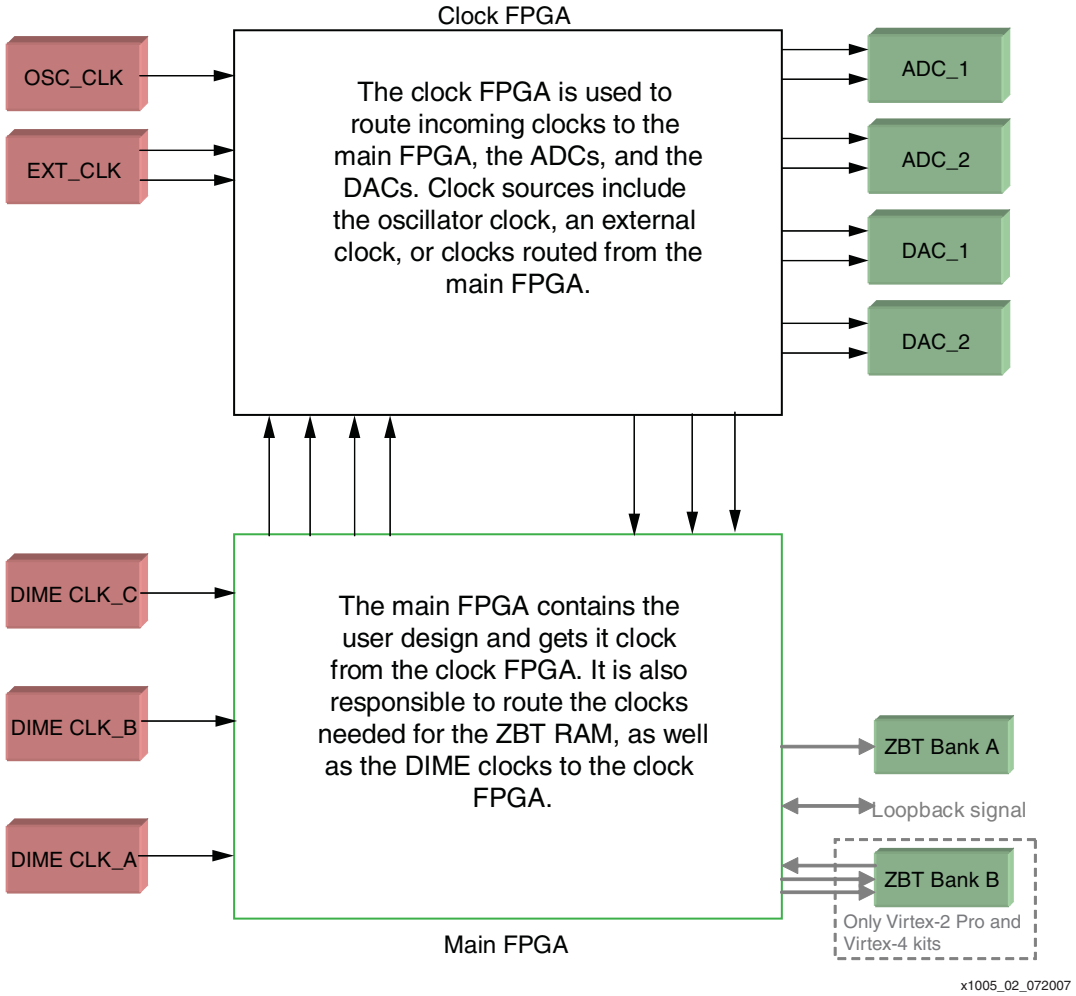


*Figure 2:* **Clocking Structure on XtremeDSP Development Kits (Clock sources are shown in pale red and clock sinks are shown in pale green)**


www.BDTIC.com/XILINX

The clock sources include the following:

- OSC_CLK: Onboard oscillator running at a frequency which is dependent on the kit used. See Table 2 for frequency details depending on the kit used.
- EXT_CLK: External oscillator
- DIME CLK_C: Clock input connected to a single fixed oscillator socket
- DIME CLK_B: Software programmable oscillator
- DIME CLK_A: Software programmable oscillator

*Table 2:* **Oscillator Frequency Differences on the Different Kits**

| Development Kit | Oscillator Frequency |
|---|---|
| XtremeDSP Development Kit for Virtex-II FPGAs (XC2V2000) | 50 MHz |
| XtremeDSP Development Kit for Virtex-II FPGAs (XC2V3000) | 65 MHz |
| XtremeDSP Development Kit for Virtex-II Pro FPGAs (XC2VP30) | 105 MHz |
| XtremeDSP Development Kit for Virtex-4 FPGAs (XC4VSX35) | 105 MHz |

Each of these clocks or combinations of clocks can be used to clock the different clock sinks (this includes the main FPGA) as shown in Figure 2. OSC_CLK and EXT_CLK are connected to the clock FPGA. When using OSC_CLK and EXT_CLK, the clock FPGA is responsible to route them as needed. The DIME clocks on the other hand are connected to the main FPGA, which is responsible for routing these clocks to the clock nets that connect the main FPGA and the clock FPGA. The clock FPGA receives the clocks on these nets and then routes them as needed.

The clock nets along with feedback nets, which feed clock signals back from the clock FPGA to the main FPGA, allow flexible routing of clock signals between the two FPGAs, as well as allow the user to select a wide range of different clocking combinations. The feedback signals between the clock FPGA and the user FPGA are necessary to let all data going to and from the user FPGA to be clocked on the same clock edge as the data in the DACs and ADCs, which are connected to the clock FPGA. More information can be found in the different kit user guides on the CDs that are shipped with the development kits.

## How to Use Each Clock Source

### OSC_CLK: Oscillator Clock

Using the onboard oscillator is the easiest way to use a clock on the development kits. The routing that needs to be done on the clock FPGA is to create a route on a global clock route from input pin K7 to the appropriate output pin. The user can either create a programming file with such a route or can use the `osc_clock_2v80.bit` file that is provided on the kit CD. The source files and example `.bit` file are provided on the kit CDs at the following location: `CD Drive\Examples\Clock_Designs`. The `osc_clock_2v80.bit` file also routes the oscillator clock to the ADCs and DACs as shown in Figure 3.


www.BDTIC.com/XILINX

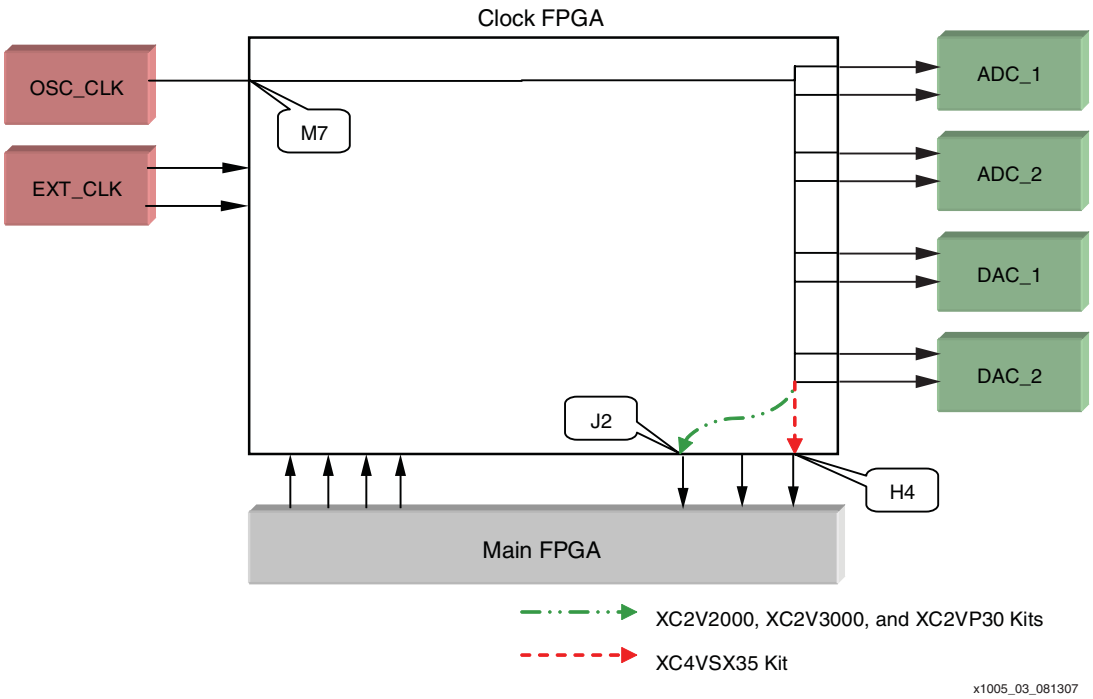


*Figure 3:* **How to Route the Clock FPGA When Using the Oscillator Clock**

When inspecting the source files used to create the `osc_clock_2v80.bit` file, it can be seen (by looking at the `BenADDA_ClockDriver_XC2V80.ucf` constraints file) that the LOC constraints on the input and output pins lock these pins down to the correct IOB sites as shown in Figure 3. The part of the constraints file which is responsible for this is shown here:

```
#####################################
# Clock Signals sent to ADCs/ DACs
#####################################
#ADC1 (Middle-right on PCB)
NET ADC0_CLKn LOC = D1; #Connected to ENCODE pin of ADC
NET ADC0_CLKp LOC = E4; #Connected to Complement of ENCODE pin of ADC
#ADC2 (Far right on PCB)
NET ADC1_CLKp LOC = G1; #Connected to ENCODE pin of ADC
NET ADC1_CLKn LOC = F1; #Connected to Complement of ENCODE pin of ADC
#DAC1 (Far left on PCB)
NET DAC0_CLKp LOC = D13; #Connected to Noninverting Input of Differential
CLK
NET DAC0_CLKn LOC = D12; #Connected to Inverting Input of Differential CLK
#DAC1 (Middle-left on PCB)
NET DAC1_CLKp LOC = G10; #Connected to Noninverting Input of Differential
CLK
NET DAC1_CLKn LOC = F12; #Connected to Inverting Input of Differential CLK
##############################################
# Various Clock Signals arriving in clock FPGA
##############################################
#Onboard Oscillator
NET OSC_IN LOC = M6; #Connected to a Primary Global CLK Pin
##############################################
# FEEDBACK Clock Signals to main FPGA
##############################################
# For the XC2V2000, XC2V3000 and XC2VP30 kits uncomment this constraint:
### NET CLK1_OUTp LOC = J2
# For the XC4VSX35 kit uncomment this constraint:
### NET CLK1_OUTp LOC = H4
TIMESPEC TS01 = FROM PADS TO PADS 7ns;
```


www.BDTIC.com/XILINX

When looking at the RTL schematic of the `osc_clock_2v80.vhd` file shown in Figure 4, it can be seen that it matches the routing structure in Figure 3.

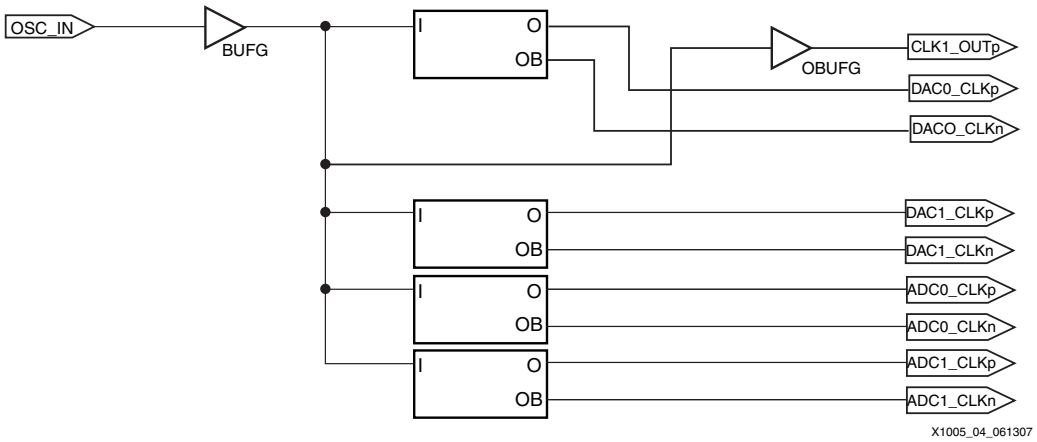


*Figure 4:* **RTL Schematic of the `osc_clock_2v80.bit` Source Files**

When using the HDL output from a System Generator for DSP design on the main FPGA and the `osc_clock_2v80.bit` file on the clock FPGA, the user needs to ensure that the *Clock pin location* in the System Generator for DSP token is set to the correct pin location depending on the kit used. Looking at Figure 1, it can either be numbers 8 or 10. As an example, the constraint is on H4 which is wired to number 8 on the main FPGA. Generation of this system locks the design's clock to the correct location, if the correct *Clock Pin location* is set in the design's System Generator for DSP token. You can check the System Generator for DSP pin lock by looking at the `.xcf` constraints file found in the netlist directory. You should expect to see something like this:

```
# Loc constraints
NET "clk" LOC = "B15";
```

Figure 5 shows how to set the clock pin location to B15. It should be noted that it is not possible to change the *Clock pin location* when targeting an XtremeDSP kit during a HW-Cosimulation, because in this case the clocking is automatically done by System Generator for DSP. This is described in more detail in the "Designs Used Inside System Generator for DSP During HW-Cosimulation" section.

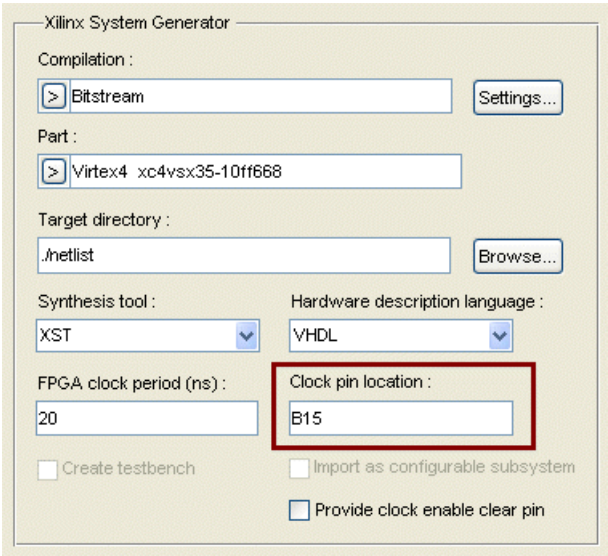


*Figure 5:* **System Generator Token Properties Showing How to Set the Clock Pin Location to B15**


www.BDTIC.com/XILINX

## EXT_CLK: External Clock

Using the external clock is almost the same as using the oscillator clock, with the main differences being that the external clock is a differential signal and the pin locations differ. There is a ready-to-use `.bit` file on each kit's CD that routes the clock to the main FPGA and the ADCs and DACs as needed. The routing structure needed to use the external clock is shown in Figure 6.

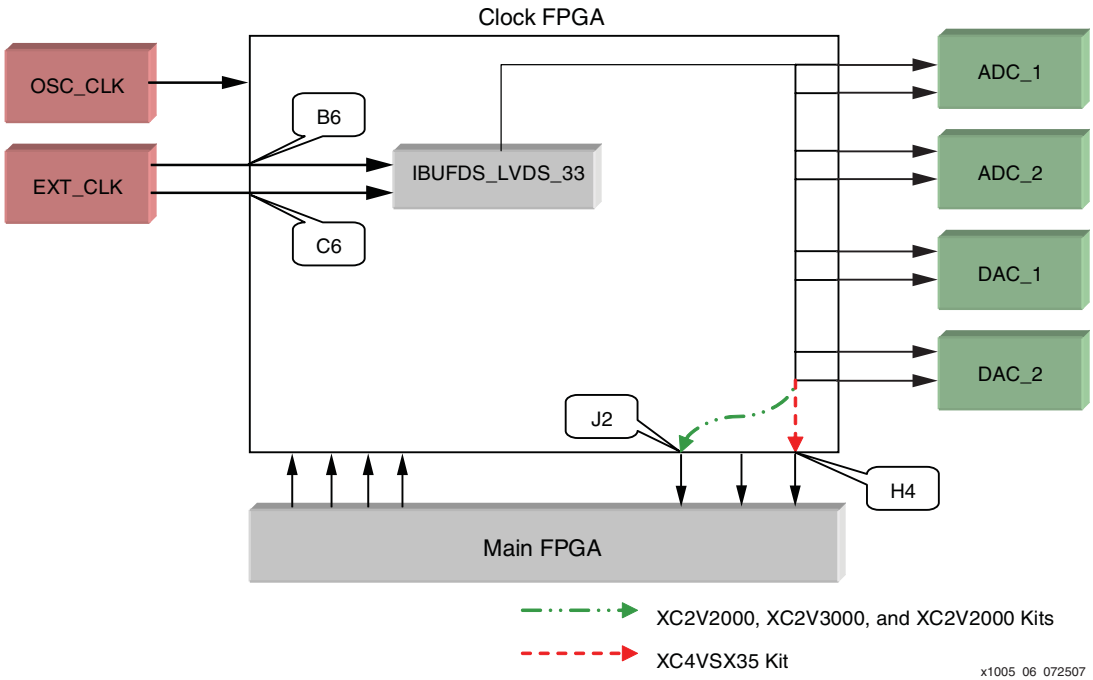


*Figure 6:* **How to Route the Clock FPGA When Using the External Clock**

Again under the `CD_Drive\Examples\Clock_Designs` folder, an `ext_clock_2v80.bit` file can be found which is compiled with the routing shown in Figure 6. The RTL of the `ext_clock_2v80.bit` looks almost identical to that shown in Figure 4 except for the different input buffer used.

Looking at the `BenADDA_ClockDriver_XC2V80_ExternalClock.ucf` constraints file, the input locations have changed and there is a P and N signal arriving at the clock FPGA. This part of the constraints file is extracted and shown below:

```
##############################################
# Various Clock Signals arriving in Virtex-II
##############################################
#USER CLK from Front Panel RF Connector via Op-Amp
NET CLK_INp LOC = B6; #Connected to a Secondary Global CLK Pin
NET CLK_INn LOC = C6; #Connected to a Primary Global CLK Pin
```

Again when using output from System Generator, the user must make sure to specify the correct *Clock pin location* in the System Generator for DSP token. For the `ext_clk_2v80.bit` file, the clock output is located on either pin J2 or H4 of the clock FPGA. The user should use Table 1 to find the pin location corresponding to number 8 or 10 depending on the kit used.


www.BDTIC.com/XILINX

## Programmable DIME Clocks

Using the programmable DIME clocks is not as easy as the oscillator clock or the external clock and no example source files are provided on the kit CDs. However, the main idea stays the same, i.e., programming the clock FPGA with the correct routing structure. After that, the user needs to take care of the clock routing on the main FPGA. The FUSE software user interface which allows the user to change these frequencies are shown in Figure 7. As seen in Figure 7, the user interface uses different names to describe the DIME clocks:

DIME CLK_A = SYSCLK, DIME CLK_B = DSPCLK, and DIME CLK_C = PIXCLK.

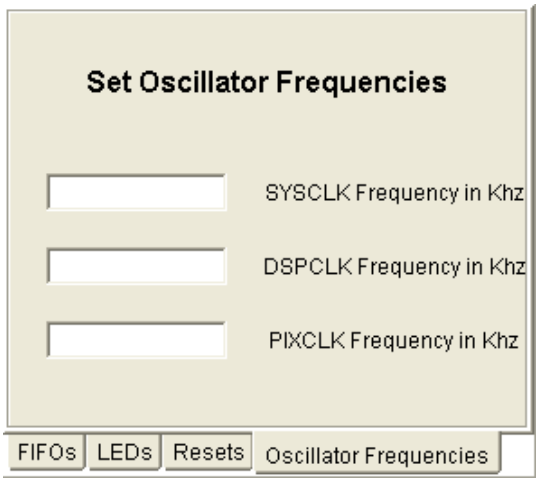


*Figure 7:* FUSE Software GUI to Change Frequencies of the DIME Clocks

The DIME clocks can be used in the following two scenarios: Either the user uses the ADCs or DACs along with the target design on the main FPGA or the user does not use the ADCs or DACs with their design. Each of these scenarios is described in the following sections. Some points that might not be clear when using these clocks at first are mentioned in the notes below.

***Notes:***

1. DIME CLK_C is connected to a socket for a crystal oscillator and will not work if an extra oscillator is not installed in the socket. Note that the XtremeDSP Development Kit does not ship with an oscillator in this socket.
2. DIME CLK_B and DIME CLK_A are programmable oscillators which can be set using the FUSE software, System Generator for DSP during HW-Cosimulation, or the DIME API.
3. A list of the possible operating frequencies of the DIME clocks can be found in each kit's user guide. When the user selects a frequency that does not match any of the supported frequencies, the frequency will be changed to the closest possible operating frequency of the DIME clock.

### When not Using ADCs or DACs

When the user does not use the ADCs or DACs, there is no need to use the clock FPGA to route clocks to them. This is shown in Figure 8. Thus, it is only necessary to specify the correct input pin of the DIME clock the user wants to use in the System Generator for DSP token. In this case, no clocks are routed to the ADCs or DACs except if the clock FPGA is programmed to route the oscillator clock or the external clock to the ADCs and DACs. When not using System Generator, it is only necessary to set the custom target design's clock input to the correct pin.


www.BDTIC.com/XILINX

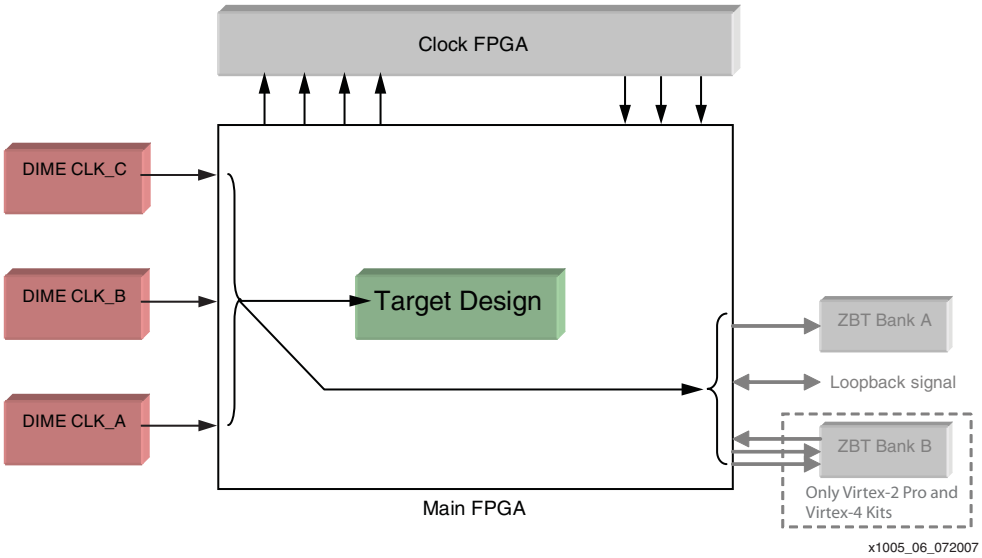


*Figure 8:* **Using the DIME Clocks Without ADCs or DACs**

### When Using ADCs or DACs

When using ADCs or DACs, it is necessary to route clocks to them. This is shown in Figure 9. The user decides which DIME clock to use, as well as which nets between the clock and main FPGA to use. There is no limitation and any DIME clock can be used with any nets between the two FPGAs as long as the net directions match the arrows in Figure 9.


www.BDTIC.com/XILINX

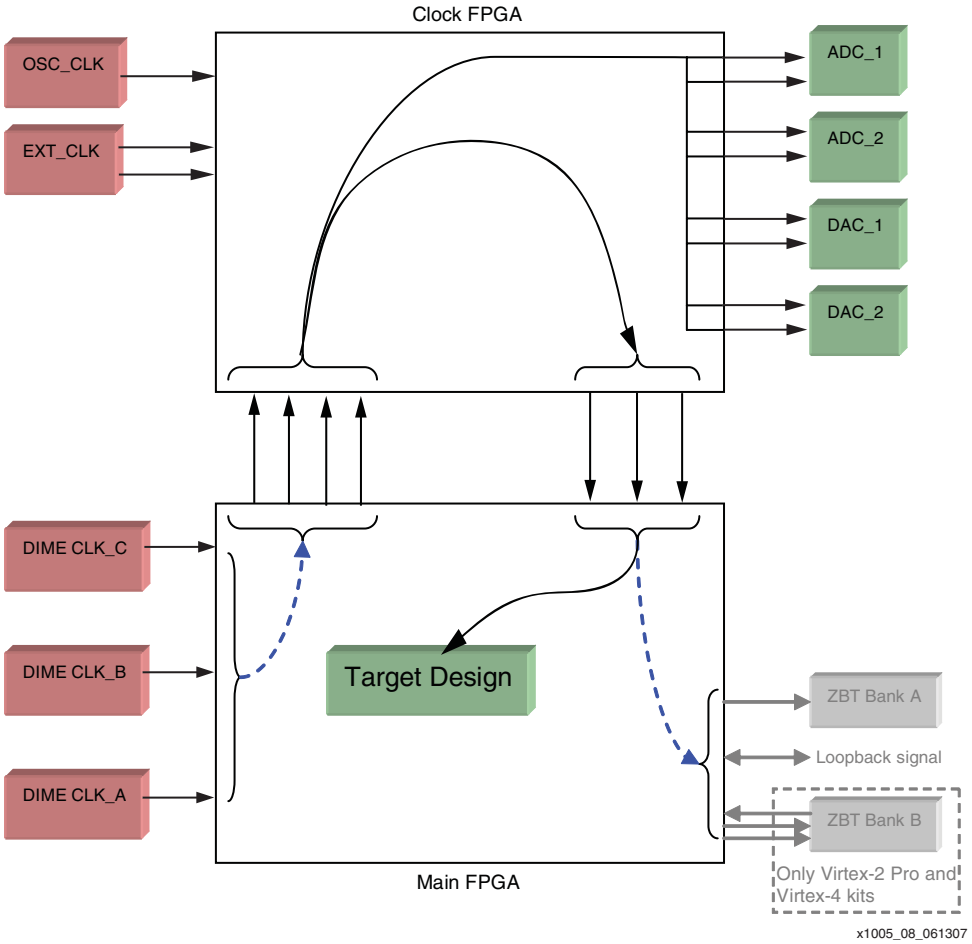


*Figure 9:* **How to Configure Both FPGAs When Using DIME Clocks to Drive the ADCs/DACs and the Main FPGA**

When using the DIME clocks, there are no ready-to use precompiled `.bit` programming files on the kit CDs. The user must create the needed programming file for the clock FPGA. To do this, it is only necessary to route the incoming clocks from the main FPGA on a global clock route to the needed ADCs and DACs, as well as to feedback nets between the two FPGAs. Example source files showing how to do this are described in detail in the "Examples Using Board Clocking Resources" section.

***Notes:***

1. During HW-Cosimulation, System Generator for DSP creates the routes shown with dotted blue lines in Figure 9. These routes route the DIME clock to the clock FPGA and the necessary clocks to the ZBT RAM. When using a design outside of the System Generator for DSP environment, it is necessary to create these routes manually. For more details, see the "Manually Creating Clock Routing Paths for the DIME Clocks" section.
2. When using a design outside of the System Generator for DSP environment, a clock routing scheme as shown in Figure 9 is recommended. This scheme minimizes the skew between the clocks feeding the ADCs/DACs and the user's target design.


www.BDTIC.com/XILINX

## System Generator for DSP Designs not Using Hardware Cosimulation

When using a System Generator for DSP generated design outside of the System Generator, there are a few things to be aware of. It is possible to use a System Generator for DSP generated design outside of System Generator for DSP by selecting any of the following compilation targets:

- HDL Netlist
- NGC Netlist
- Bitstream

System Generator for DSP takes care of the following for the user when doing a HW-Cosimulation that needs to be done manually for a design to work correctly outside of the System Generator for DSP environment.

- When using the ADCs or DACs, the DIME clocks are automatically routed to the clock FPGA, as described in the "Designs Used Inside System Generator for DSP During HW-Cosimulation" section, and the feedback clock is automatically routed to the ZBT RAM.
- The output pin location constraints is created for the user when using components out of the XtremeDSP kit library.

The following sections explain how to perform these steps manually.

### Manually Creating Clock Routing Paths for the DIME Clocks

The different clock routes that need to be created manually by the user are shown in dashed blue lines in Figure 10. *It is important to know that this is only needed in the following two scenarios:*

1. When using only the ADCs or DACs, and not the ZBT RAM:

   In this case, the route labeled number 1 must be created by the user to route the selected DIME clock to the clock FPGA to drive the ADCs and DACs.

2. When using the ZBT RAM *and* the ADCs or DACs:

   In this case, the route labeled number 1 must be created, as well as the route labeled number 2, which is the clock driving the ZBT RAM.

Examples 3, 4, and 5 in the "Examples Using Board Clocking Resources" section use the manually created clock path labeled with the number 1 in Figure 10.

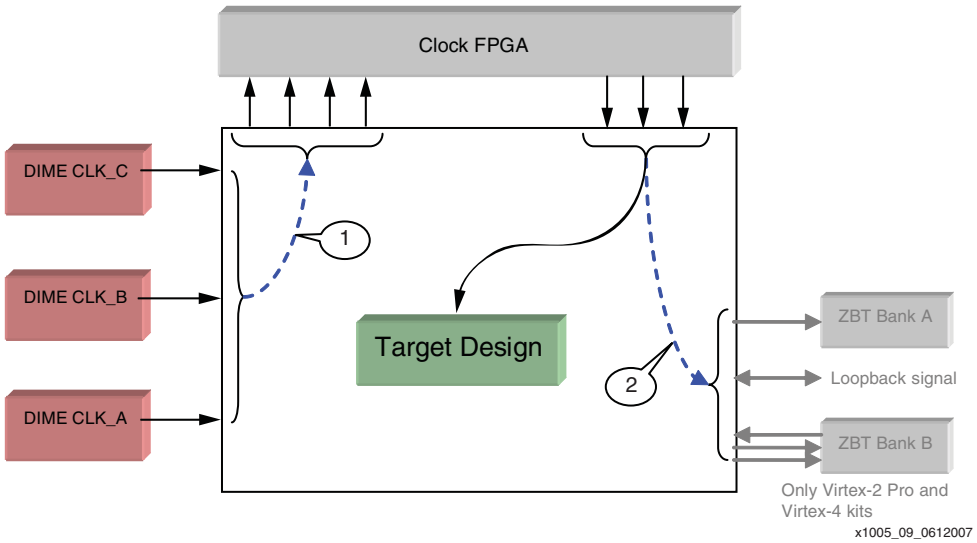


*Figure 10:* **Clock Routes Created Automatically by System Generator for DSP During HW-Cosimulation**


www.BDTIC.com/XILINX

## Specifying Output Location Constraints for XtremeDSP Library Components

To use the components in the XtremeDSP Kit Simulink® library, the user needs to ensure that the IOB locations are specified in the user design. The components of this library are shown in Figure 11.

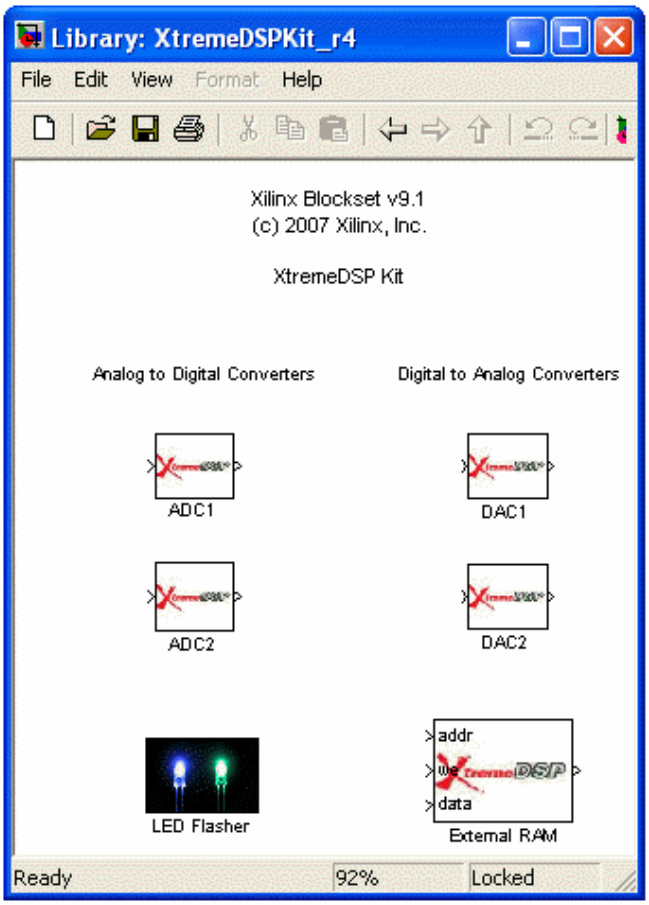


*Figure 11:* **Components Contained in the XtremeDSP Kit Simulink Library**

***Note:*** Pin names can be found in the user guide for the XtremeDSP Development kit, as well as in the UCFs.

Because these components can be used when any of the different XtremeDSP kits are targeted, the IOB locations are not fixed. When doing a HW-Cosimulation, the target kit is specified. During the generation of the HW-Cosimulation token, the input and output of the components are constrained using an `.ucf` file created in the HW-Cosimulation output directory. In the Virtex-IV kit, the output file (for example, called `benone_top_pci.ucf` in the PCI HW-Cosimulation case) contains the correct LOC constraints for these components on the XC4VSX35 as shown below. The files used by System Generator for DSP for this purpose can be found in the `SYSGEN_INSTALL\sysgen\hwcosim\xtremedspkit` directory:

```
################
## DAC Signals
################
# DAC Channel 1
net dac1_d(0) loc = A7;
net dac1_d(1) loc = C7;
net dac1_d(2) loc = B7;
net dac1_d(3) loc = C5;
net dac1_d(4) loc = D4;
```


www.BDTIC.com/XILINX

```
net dac1_d(5) loc = C4;
#....
#... etc.
# DAC Channel 2
net dac2_d(0) loc = D10;
net dac2_d(1) loc = F10;
net dac2_d(2) loc = C10;
net dac2_d(3) loc = E10;
net dac2_d(4) loc = D9;
net dac2_d(5) loc = F9;
#....
#... etc.
# LEDs
net led1_green loc = E26;
net led2_green loc = D3;
net led1_red loc = D26;
net led2_red loc = F3;
#....
#... etc.
net zbt_addr(0) loc = AD2;
net zbt_addr(1) loc = Y3;
net zbt_addr(2) loc = AD13;
#....
#... etc.
```

When not doing HW-Cosimulation, the user needs to specify that the IOB location constraints need to be specified. The following steps describe how to do this for a System Generator for DSP design. When doing a custom design, the user needs to ensure that the output to the library components are correctly LOCed down.

1. Right click on the component and select Look Under Mask (see Figure 12).

   This reveals the ADC/DAC data path and control signals along with the gateway out blocks that need to be constrained.

2. Edit the properties of each gateway out block.

Once under the mask, the user needs to go into the properties of all the Gateway Out blocks and tick the Specify IOB location constraints box as shown in Figure 13. If another kit is used, these pad locations should be changed to match the correct output pins for that kit.

By default, the IOB pad locations as shown in Figure 13 match the correct output pins for the XC2V3000-4FG676, which is found on the XtremeDSP Development Kit for Virtex-II (XC2V3000). If another kit is used, the IOB pad locations should be changed to match that kit. These details can be found in the respective kit's user guide.

***Notes:***

1. The figures explaining this process only show how to specify the IOB location constraints for a DAC component, but the process is the same for the ADCs, LED Flasher, and ZBT RAM.
2. The user needs to disable the component's link to the library to edit gateway outs under the mask. When asked the first time, the user can do this by selecting "disable link." Alternatively, the link can be disabled by using Edit -> Link Options -> Disable Link.


www.BDTIC.com/XILINX

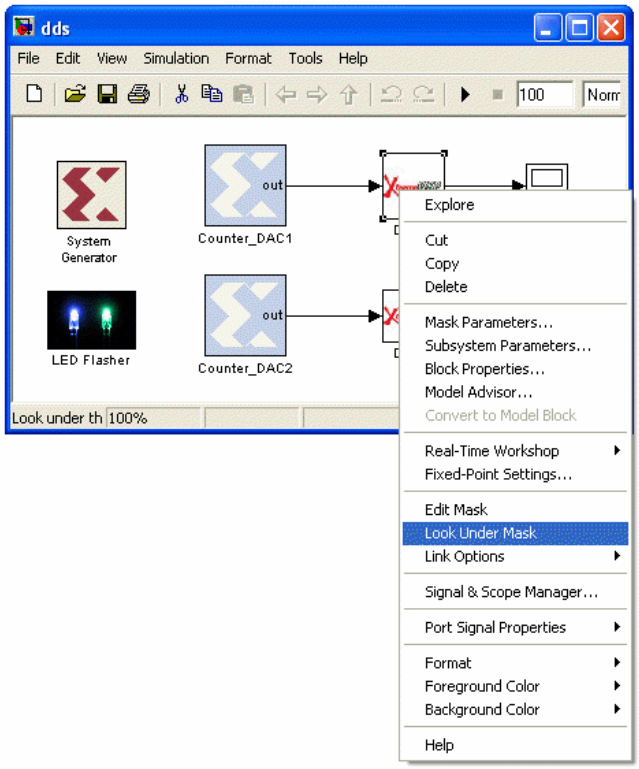


*Figure 12:* **Right-Clicking on the DAC Component Block Gives the User the Option to Look Under the Mask of That Block**

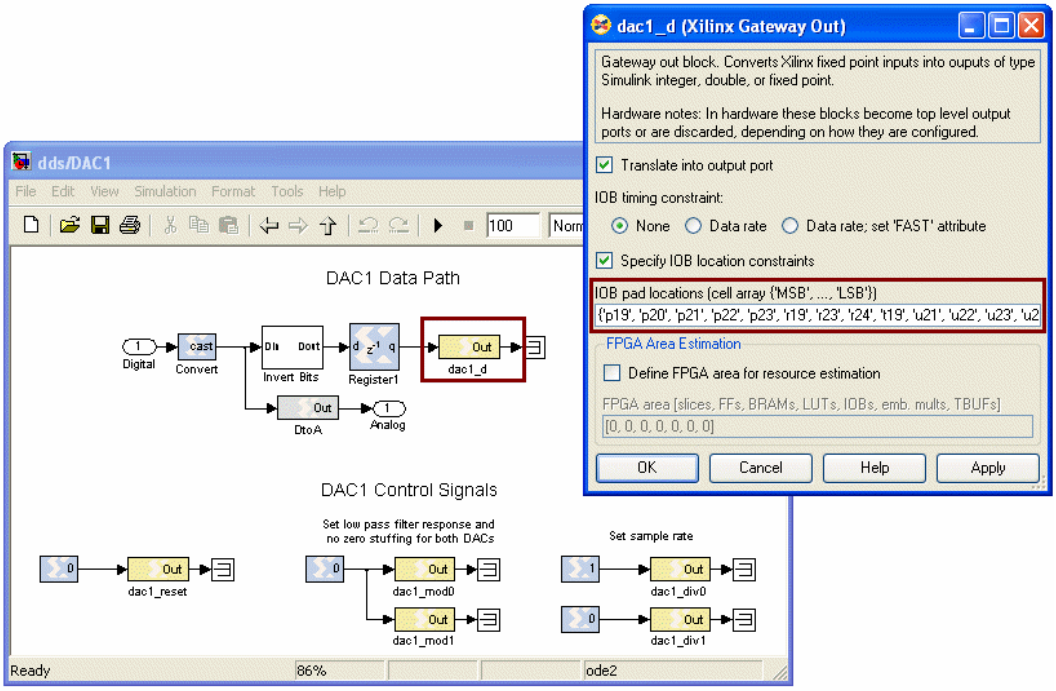


*Figure 13:* **Editing Each Output Gateway to Specify IOB Location Constraints Manually**


www.BDTIC.com/XILINX

# Designs Used Inside System Generator for DSP During HW-Cosimulation

When doing a HW-Cosimulation on one of the XtremeDSP development kits from within the System Generator for DSP environment, the clocking can be done in two ways: Free Running and Single Stepped. More details on these two methods can be found in the *Using Hardware Co-Simulation* section under the *Hardware Co-Simulation Clocking* subsection of the *System Generator for DSP User Guide*. This section describes how a free running clock is handled by System Generator. Since the clocking requirement for using the ADCs and DACs is to use a Free Running clock, this clocking method is described in detail in this section. This section ends with a short discussion of how the clocking works in the Single Stepped HW-Cosimulation logic.

## Example Used to Demonstrate HW-Cosimulation Clocking

The model used below is the same as the one used in most of the examples in the "Examples Using Board Clocking Resources" section and is described in more detail in that section. A screenshot of the model is shown in Figure 14. The model is taken and a JTAG Hardware Cosimulation target for the XtremeDSP Development Kit for Virtex-4 FPGAs is selected and generated. The files for the example are provided under the HW-Cosimulation Example Design directory of the reference design files.

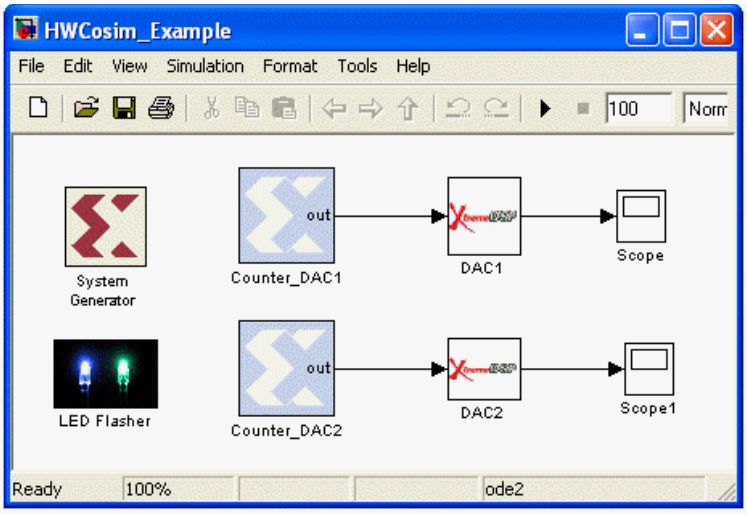


*Figure 14:* **Example Model Used to Explain How HW-Cosimulation Clocking is Done**

## Free Running HW-Cosimulation Clocking

When doing a HW-Cosimulation, the clocks are handled automatically. The user should not be concerned about this, except if the user wants to change the operating clock frequency while running the HW-Cosimulation. If not, the clock period is determined by the period entered in the design's System Generator for DSP token. Thus, this explanation is to provide a better understanding of how the clocking is actually done by System Generator for DSP when a free running clock is selected, and to help debug problems. The diagram in Figure 15 shows the routing setup of the clocks done during a Free Running HW-Cosimulation.

***Notes:***

1. The routing is automatically done by System Generator for DSP during HW-Cosimulation on both the main FPGA and the clock FPGA.
2. System Generator for DSP is out of sync with the hardware and the user cannot monitor the ADCs or DACs using the scopes in MATLAB. External hardware is necessary for this.


www.BDTIC.com/XILINX

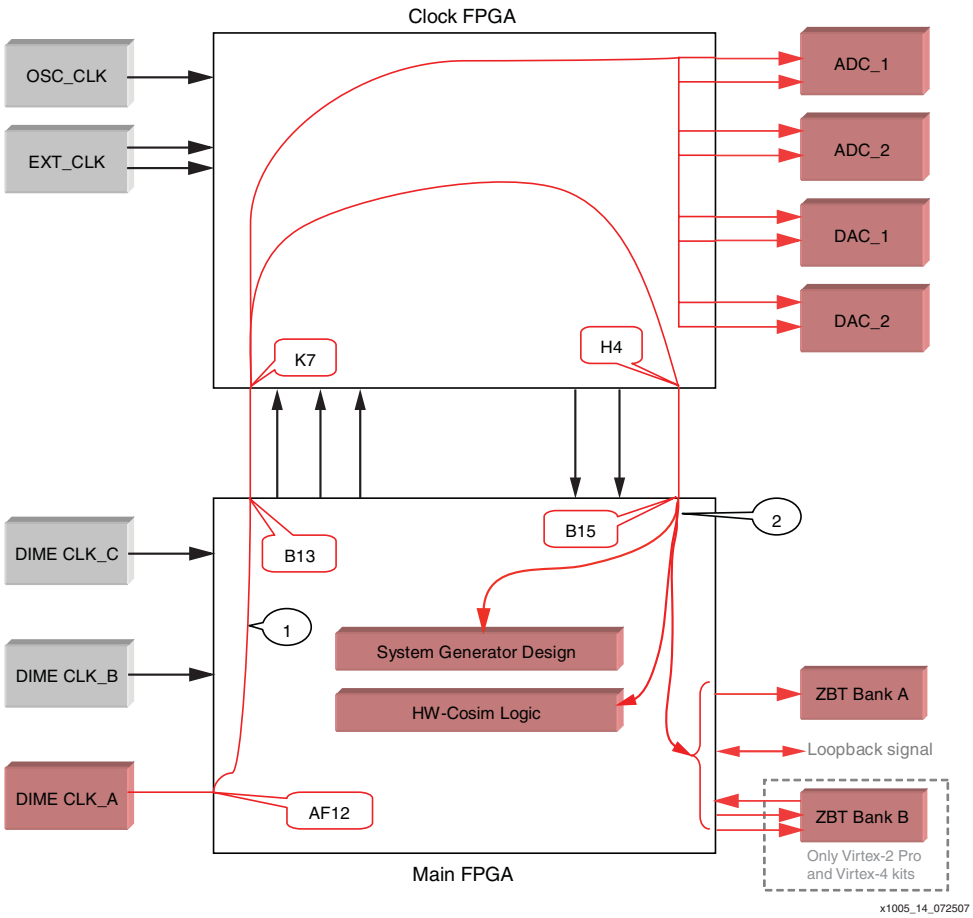


*Figure 15:* **How Clock Routing is Done by System Generator for DSP During a Free Running HW-Cosimulation**

The design logic in the diagram refers to the logic used to realize the actual design in your Simulink model and the HW-Cosimulation logic refers to the logic needed to do the HW Cosimulation. This is shown in Figure 16 where the left floorplan shows the design logic and the right floorplan shows the HW-Cosimulation logic. These two figures show that a HW-Cosimulation adds logic to the design which is used to feed data back and forth between Simulink and the board. If generating a Bistream, HDL Netlist, or NGC Netlist of the design, it would not contain the HW-Cosimulation.

As shown in the routing diagram in Figure 15, DIME CLK_A is routed to the clock FPGA (labeled as route number 1) from where it is fed to the ADCs and DACs. The clock is also routed back from the clock FPGA to the main FPGA (labeled as route number 2). This routing can be easily verified by looking at the routed design used to create the `.bit` programming file uploaded to the main FPGA during HW-Cosimulation.

To verify and view the implementation of the numbered routes in the routing diagram, follow the steps below for each route:

1. To view the implementation of the route labeled as number 1 in Figure 15, open the `benone_top_jtag.ncd` file in the Reference Design Files archive in the FPGA Editor and select the `gen_clka_OBUF` net in the List window.
2. To view the implementation of the three routes labeled as number 2 in Figure 15, open the `benone_top_jtag.ncd` file in FPGA Editor and select the following nets in the List window: `zbt_clk_OBUF` + `jtagcosim_top_u1/sys_clk_buf` + `cosim_clk` + `clk1_fb_ibufg`.


www.BDTIC.com/XILINX

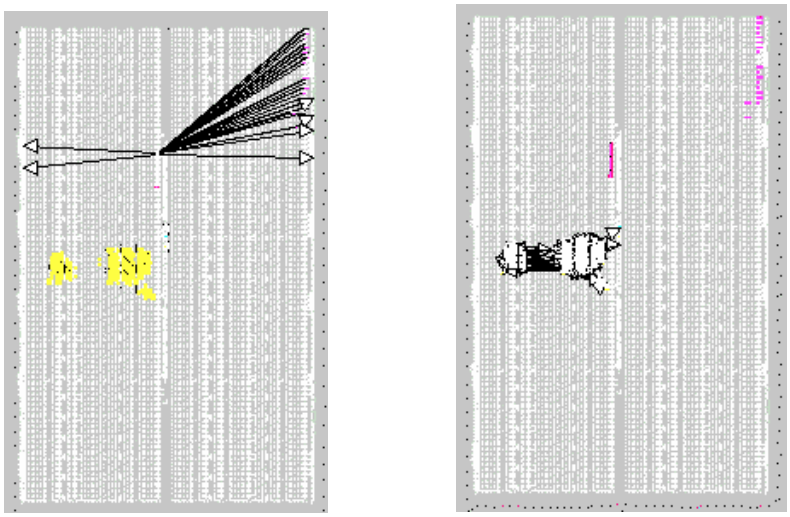


*Figure 16:* **Left Floorplan Shows the Design Logic Needed to Realize the Example Design in Hardware and Right Floorplan Shows the HW-Cosimulation Logic**

***Note:*** The floorplan in Figure 16 is for a JTAG HW-Cosimulation on the XtremeDSP Development Kit for Virtex-4 FPGAs generated from the example model in Figure 14. The floorplan shown in Figure 16 looks different for the other kits. The same holds true for the routes shown in Figure 17 and Figure 20.

## Single Stepped HW-Cosimulation Clocking

When doing a Single Stepped HW-Cosimulation, the user design and the ZBT RAM is clocked by a clock given to the HW-Cosimulation logic by Simulink. As shown in Figure 17, the HW-Cosimulation logic produces the Simulink clock and this clock is selected instead of the feedback clock from the clock FPGA by using a BUGFGCTRL. The Simulink clock route is shown with the dotted line in Figure 17. Note that this route and the BUFGCTRL are also included in the Free Running HW-Cosimulation, because the same programming file is used for both types of HW-Cosimulations. This route and the BUFGCTRL were not shown in Figure 15 to make the Free Running HW-Cosimulation easier to understand. In the Free Running case, the BUFGCTRL selects the feedback clock instead of the Simulink clock.

***Note:*** In this mode, ADCs or DACs cannot be used because they require a free-running clock.

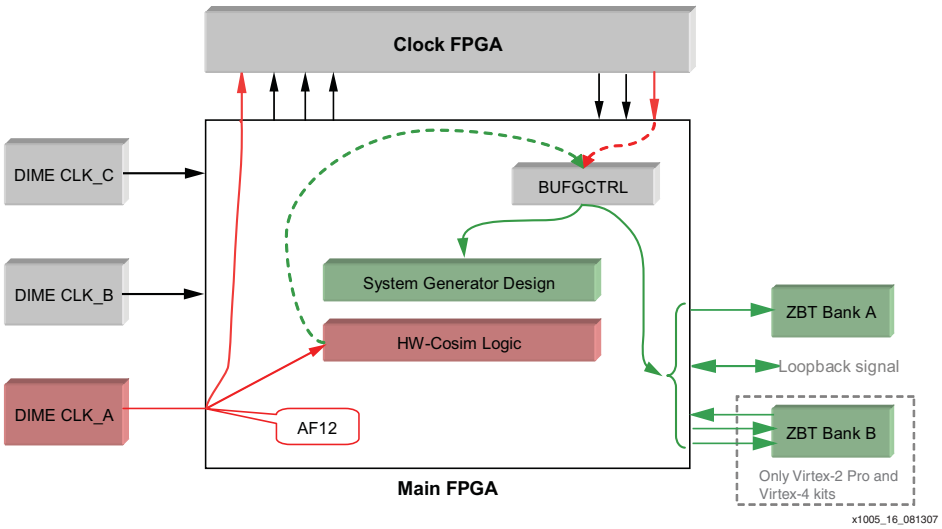


*Figure 17:* **Diagram Showing Clock Routing During Single-Stepped HW-Cosimulation**


www.BDTIC.com/XILINX

# Examples Using Board Clocking Resources

This section presents five different examples of using clocking resources on the board in addition to those already on the XtremeDSP Development Kit's CD. Deskewing the clocks using a DCM was not done in these examples, but should be done if excessive clock jitter or delay is encountered.

## Model Used in Examples

For the System Generator for DSP part of examples 1 to 4, the model shown in Figure 18 was used. The model flashes the LEDs using the LED Flasher and uses both DAC1 and DAC2. The DACs are fed by two counters that change between 0 and 1 continuously on every clock edge. This provides an easy way to verify the clock frequency of the clock that feeds the counter. Thus, the frequency observed by an oscilloscope on this output should be doubled to get the respective counter clock frequency.

The discussion of the following five examples are based on implementation on the XtremeDSP Development Kit for Virtex-4 FPGAs. Note that the examples can be applied to any of the four development kits. Source files, ready-to-use `.bit` files, and oscilloscope results are available in the reference design files for both the XtremeDSP Development Kit for Virtex-4 FPGAs and the XtremeDSP Development Kit for Virtex-II FPGAs (XC2V3000).

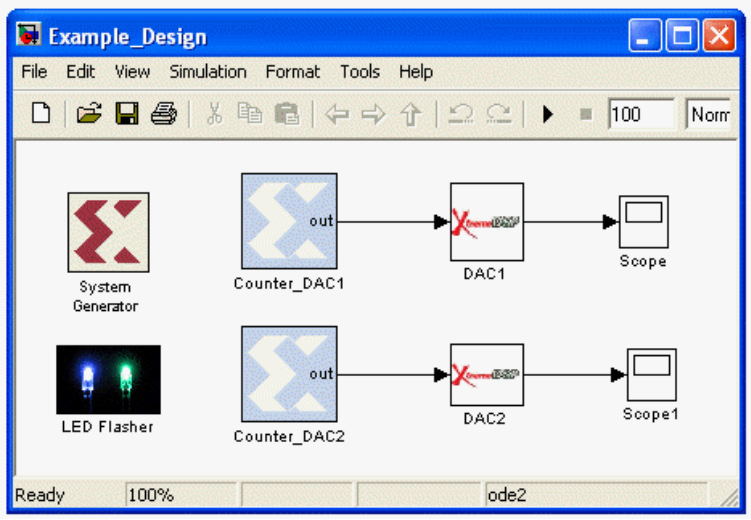


*Figure 18:* **Demonstration System Generator for DSP Design Used in Examples**


www.BDTIC.com/XILINX

## Example 1

**Description**: This example shows wrong output on the DACs where the DAC output pad locations under the DAC component masks were not specified correctly as described in the "Specifying Output Location Constraints for XtremeDSP Library Components" section. Similar results are obtained when something is done wrong while configuring the clocks.

**Clock used:** OSC_CLK (`osc_clock_2v80.bit` from kit CD)

**Model configuration:** Demo model with clock input on pin B15

**Files:** -> Reference Design Files -> Example 1

**Output on the DACs:** When examining the output on the DACs shown in Figure 19, no output is present and the DACs are not driven by any data. This is because the DAC output pad locations were not specified in System Generator.

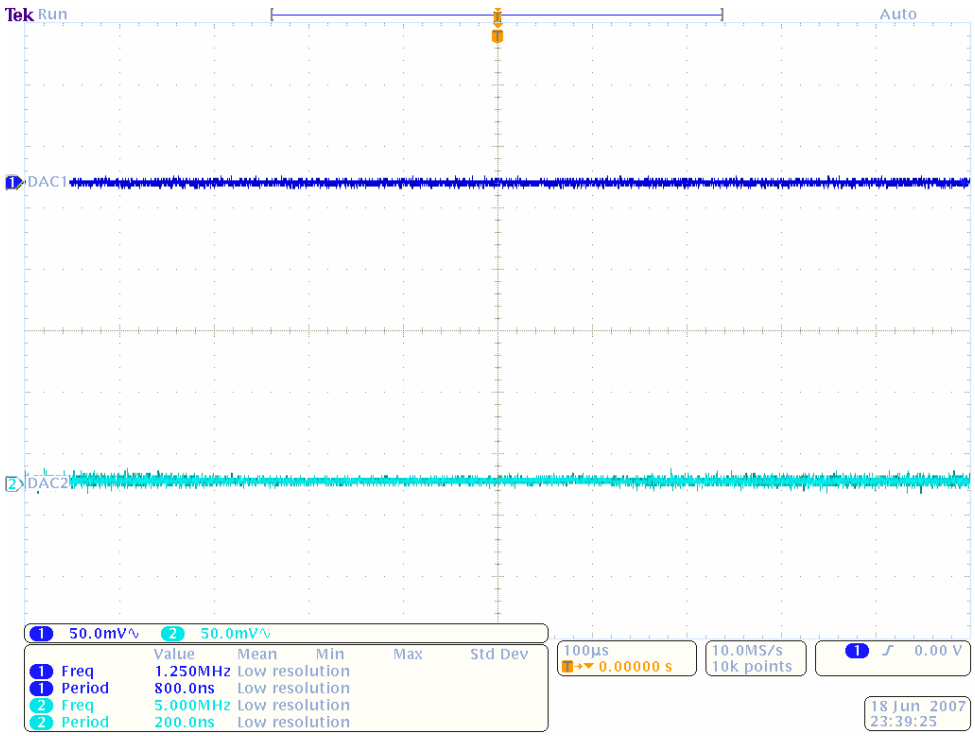


*Figure 19:* **Output on DAC_1 (top) and DAC_2 (bottom) for Example 1**


www.BDTIC.com/XILINX

## Example 2

**Description**: This example has the same setup as in example 1, but this time the DAC output pad locations were specified correctly. The `.bit` file for the demo model was generated in System Generator for DSP by choosing a Bitstream compilation target with the clock pin set to B15.

**Clock used**: `OSC_CLK` (`osc_clock_2v80.bit` from kit CD)

**Model configuration**: Demo model with clock input on pin B15

**Files**: -> Reference Design Files -> Example 2

**Clock routing**: See Figure 20.

**Output on the DACs**: The output on DAC1 and DAC2 are as expected when in looking at the routing diagram in Figure 20. The output signals (in Figure 21) on both DACs are exactly the same and the spectrum of these signals clearly shows that a signal with a frequency of 52.5 MHz is present. This corresponds to the 105 MHz oscillator clock present on the XtremeDSP Development Kit for Virtex-4 FPGAs that is used to drive both counters in the demonstration model.

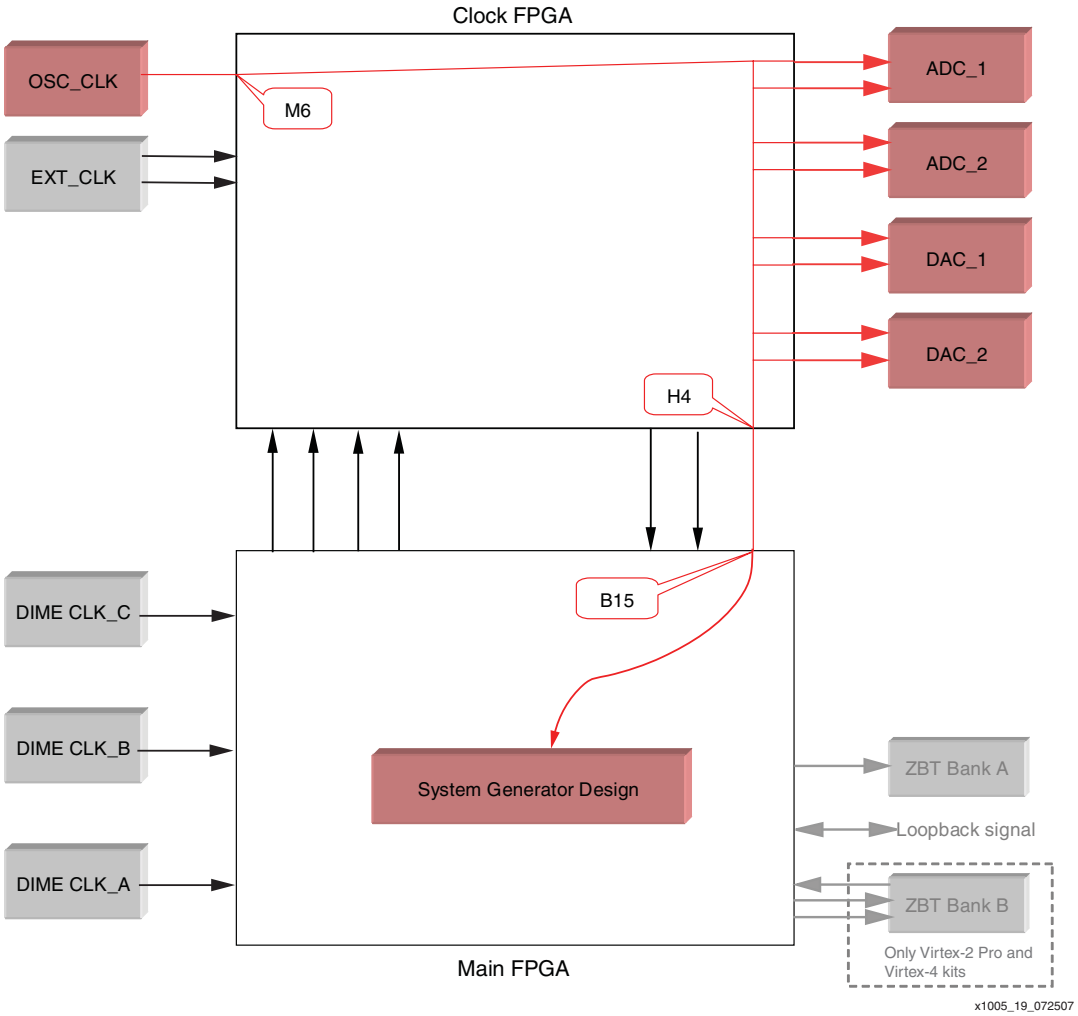


*Figure 20:* **Clock Routing for Example 2**


www.BDTIC.com/XILINX

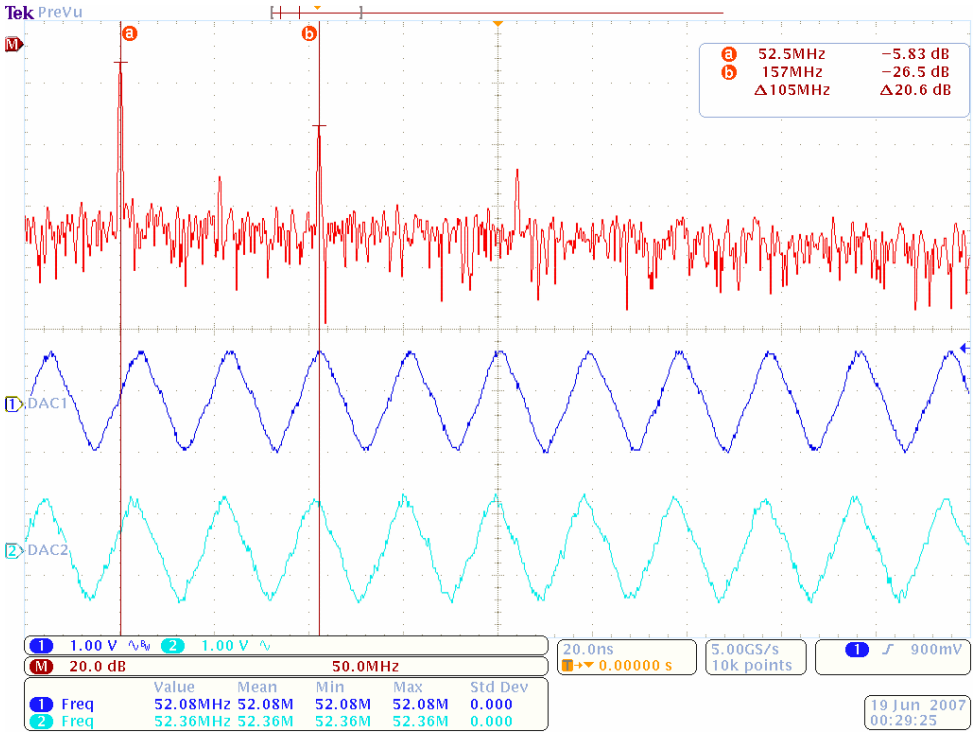


*Figure 21:* **Output on Both DACs Showing a Clock of 2 x 52.5 = 105 MHz, Which Corresponds to the Onboard Oscillator Clock**


www.BDTIC.com/XILINX

## Example 3

**Description**: This example shows how to use a DIME clock for clocking the ADCs/DACs and the System Generator for DSP design. In this example, `DIME CLK_A` is used to drive both the DACs and the System Generator for DSP demo design. For this setup, a new `.bit` file with the clock input on pin K7 was created for the clock FPGA and the System Generator for DSP design was modified to route the DIME clock to the correct output pin, in this case B13, as shown in Figure 19.

**Clock used**: `DIME CLK_A`

**Model configuration**: Demo model with clock input on pin B15

**Files**: -> Reference Design Files -> Example 3

**Clock routing**: See Figure 22.

**Output on the DACs**: The output on DAC1 and DAC2 as shown in Figure 23 runs at half the frequency of `DIME CLK_A`. The output signals on both DACs are again exactly the same and the spectrum of these signals shows that a signal with a frequency of 30 MHz is present. This corresponds to `DIME CLK_A` set to 2 x 30 = 60 MHz which is used to drive both counters. When the frequency of `DIME CLK_A` is changed using the FUSE software, the peak in the spectrum of the output signals shifts to half the frequency of the clock. This can be seen when looking at the images found in the Results folder of this example in the reference design files.

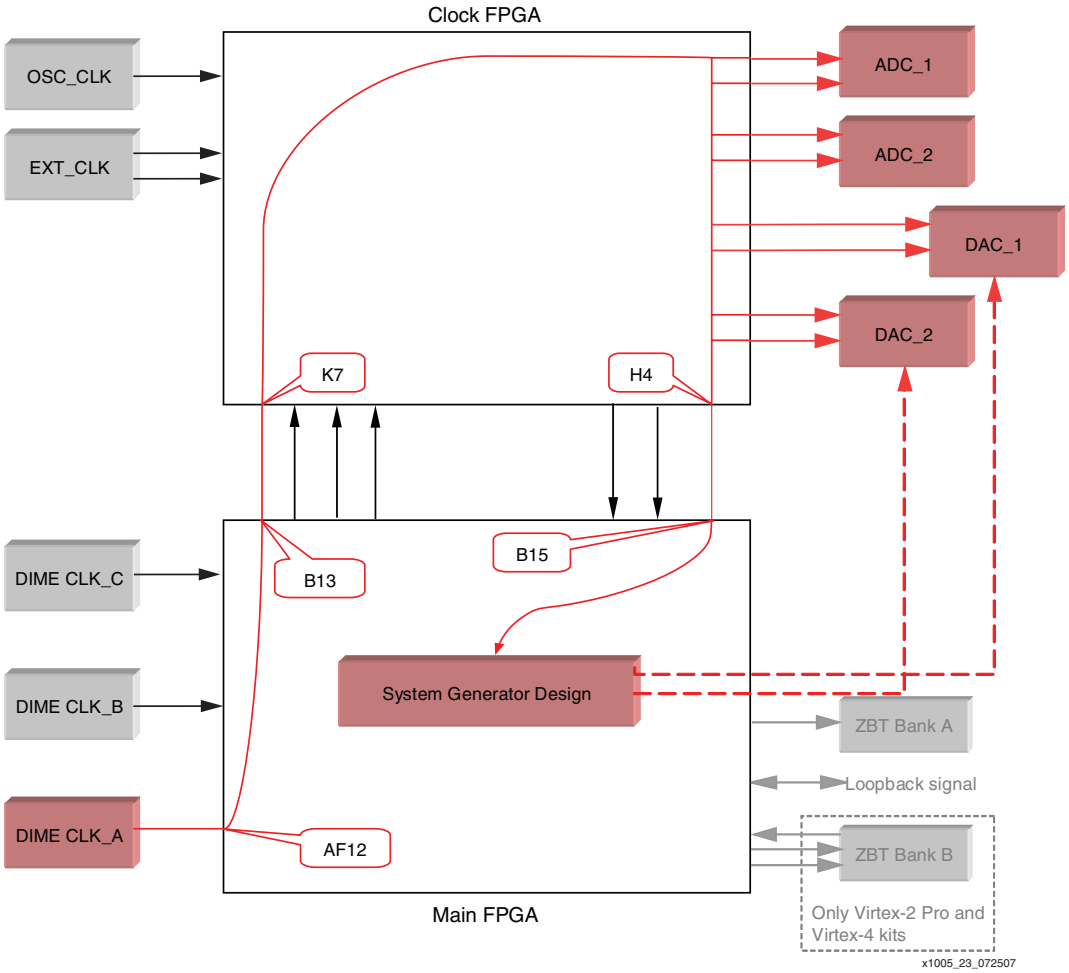


*Figure 22:* **Clock Routing for Example 3**


www.BDTIC.com/XILINX

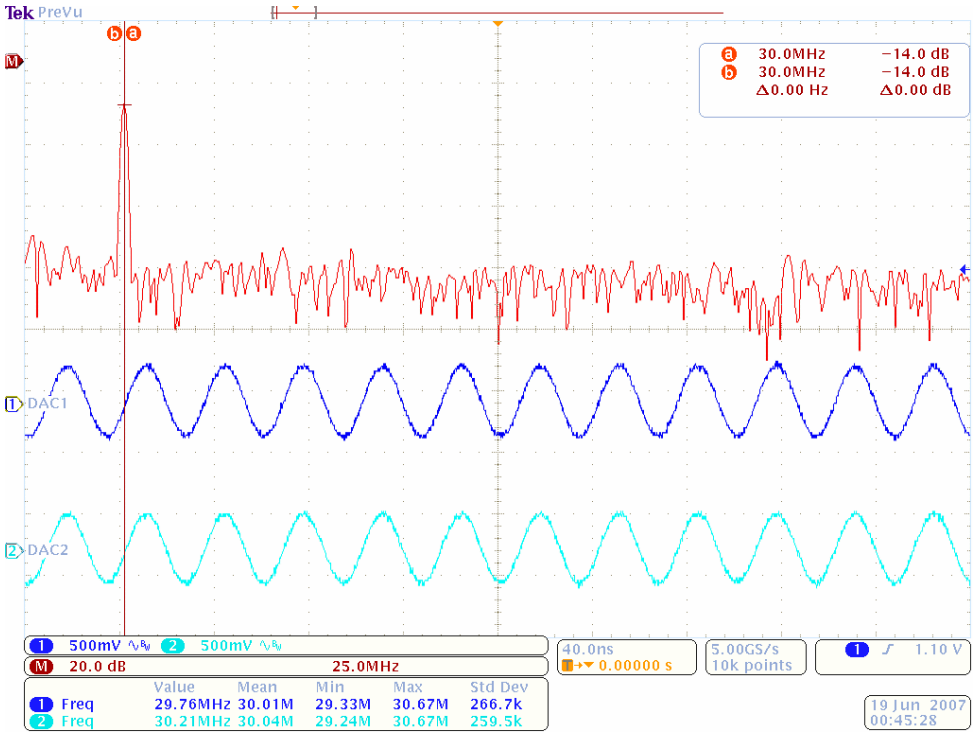


*Figure 23:* **Output on Both DACs Showing a Clock of 2 x 30 = 60 MHz, Which Corresponds to DIME_CLK_A set to 60 MHz**


www.BDTIC.com/XILINX

## Example 4

**Description**: This example shows how to use different clocks for a combined System Generator for DSP and ISE design where the ISE design is simply the System Generator for DSP output changed to clock one counter with a different clock. For this setup, a new `.bit` file was created for the clock FPGA which has the oscillator clock as input on pin M6 and `DIME CLK_B` as input on pin K7.

**Clocks used**: `OSC_CLK` and `DIME CLK_B`

**Model configuration**: Demo model with clock input on pin A16 combined with custom ISE code clocked from pin B14.

**Files**: -> Reference Design Files -> Example 4

**Clock routing**: See Figure 24.

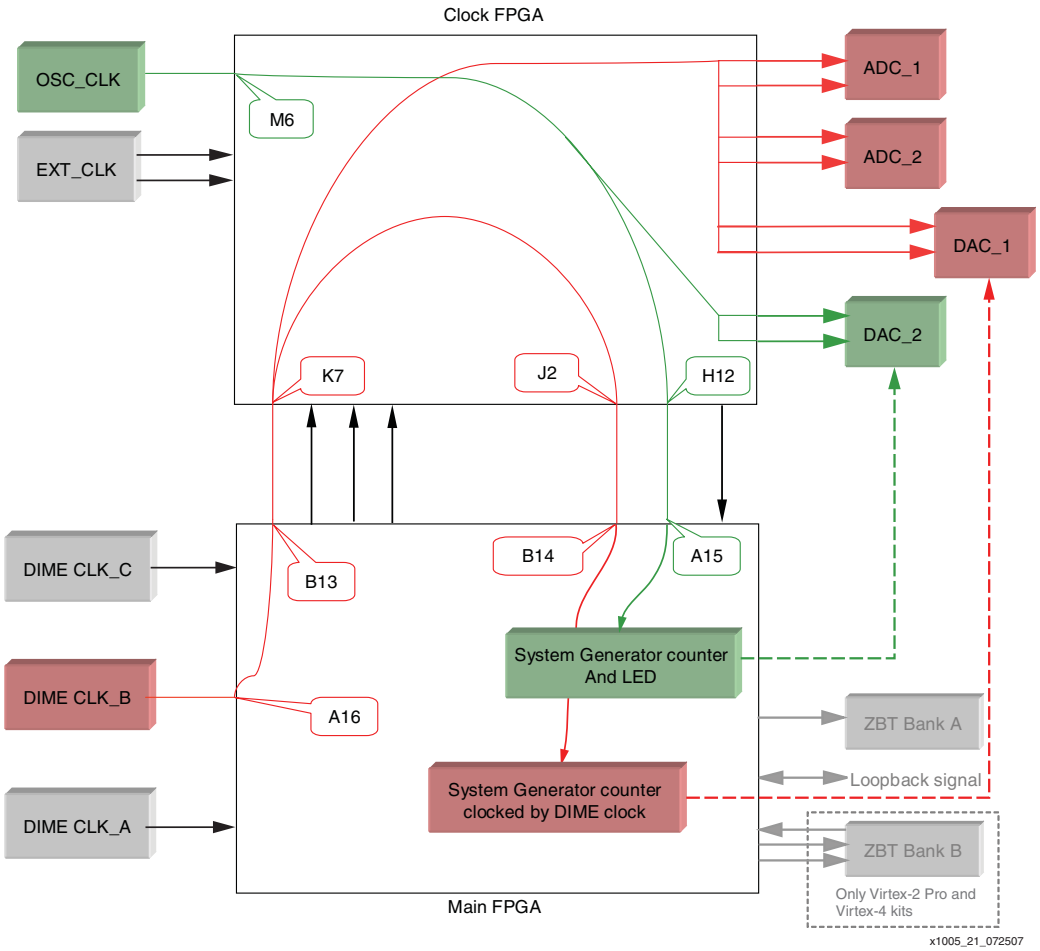


*Figure 24:* **Clock Routing for Example 4**

**Output on the DACs**: Three oscilloscope shots are used to discuss the output of Example 4. First, the output on DAC2 is shown in Figure 25, which shows that the output signal's spectrum has a peak at 52.5 MHz representing the 105 MHz oscillator clock on the XtremeDSP Development Kit for Virtex-4 FPGAs. When looking at the routing diagram, this is the expected output on DAC2.

Figure 26 and Figure 27 show the output on DAC1 for two different `DIME CLK_B` frequencies. Figure 26 shows the case in which the clock is set to 80 MHz in the FUSE GUI and Figure 27 shows the case when the clock is set to 40 MHz. This output is also expected when looking at the routing diagram.


www.BDTIC.com/XILINX

It is important to note that the design running on the main FPGA is driven by clocks from the feedback paths between the clock FPGA and the main FPGA and not directly from the DIME clock input pins, in this case, pin A16. This is the recommended way to route the clocks to minimize skew between the clocks driving the DACs and the clocks driving the design.

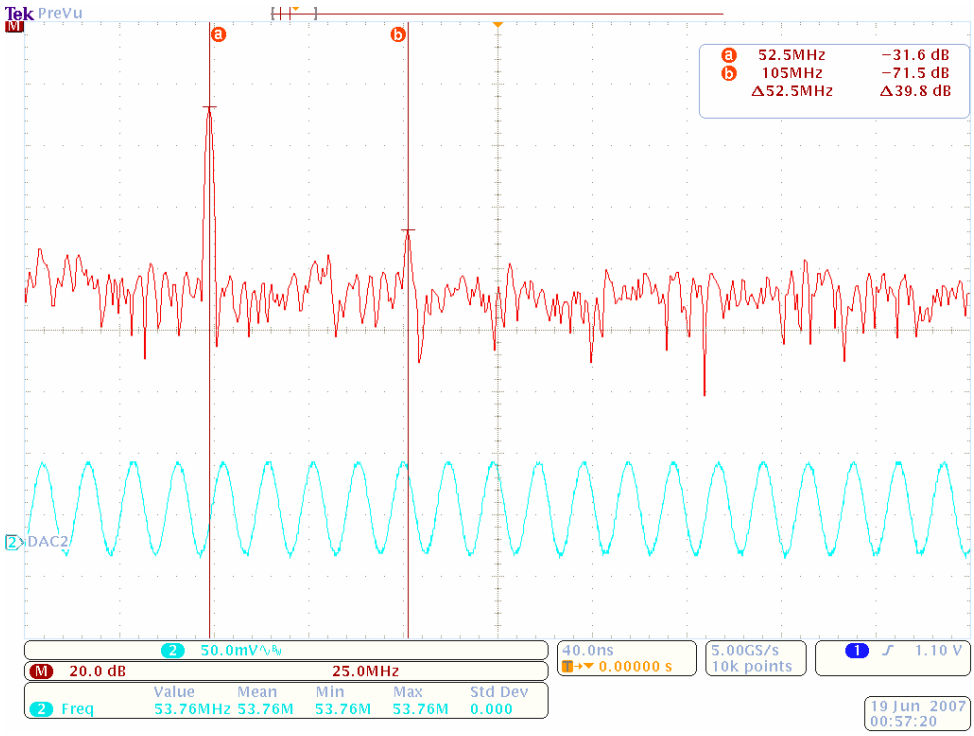


*Figure 25:* **Output on DAC_2 for Example 4 Showing a Clock of 2 x 52.5 = 105 MHz, Which Corresponds to the Onboard Oscillator Clock**

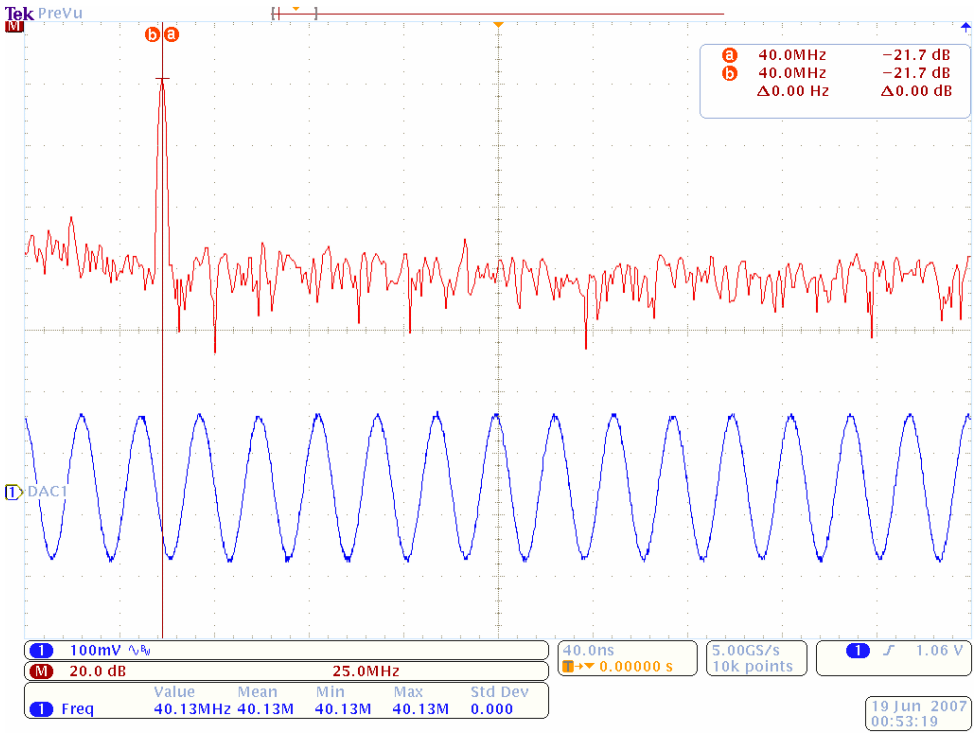


*Figure 26:* **Output on DAC_1 for Example 4 with DIME CLK_B = 80 MHz**


www.BDTIC.com/XILINX

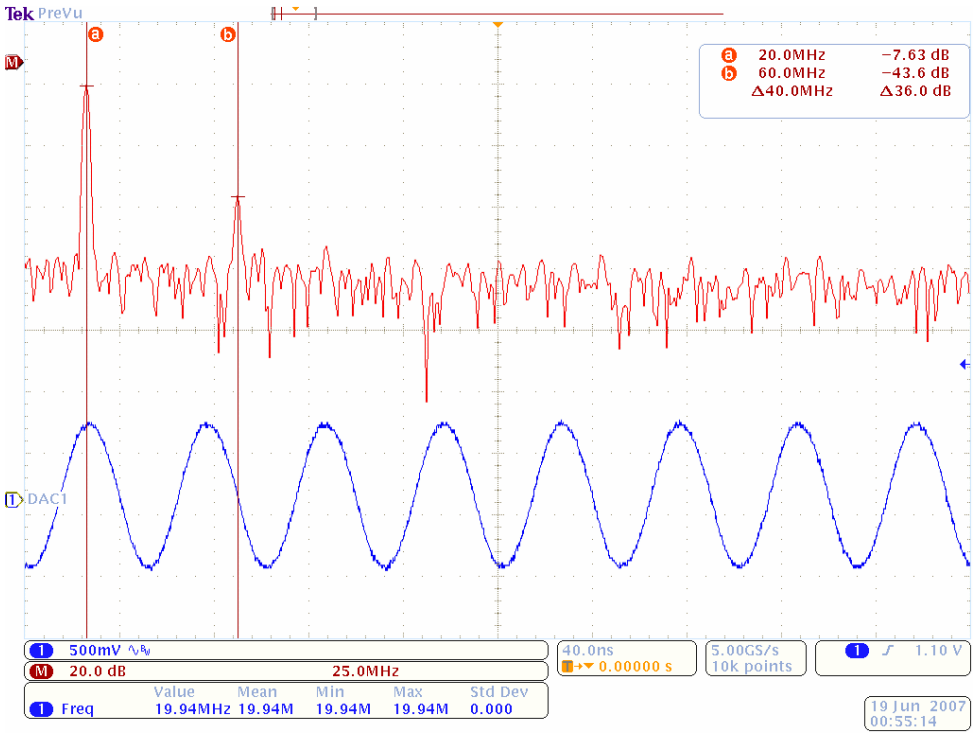


*Figure 27:* **Output on DAC_1 for Example 4 with DIME CLK_B = 40 MHz**


www.BDTIC.com/XILINX

## Example 5

**Description**: This example shows how to use a System Generator for DSP design with multiple subsystems where each one is clocked by a different clock resource on the kit. In the example, a multiple clock domain System Generator for DSP design was used to show how to use multiple clocks available on the kit in a single System Generator for DSP design. The design model is shown in Figure 28. Note that the paths to route the DIME clocks to the clock FPGA should be added manually as described previously in the "Manually Creating Clock Routing Paths for the DIME Clocks" section. Each subsystem in Figure 28 contains a System Generator for DSP token for which the clock input pins were set to match the clock routing diagram shown in Figure 30. The contents of each subsystem is shown in Figure 29.

**Clocks used**: `DIME CLK_A`, `DIME CLK_B`, and `OSC_CLK`

**Model configuration**: Multiple Clock Domain system described above in the description.

**Files**: -> Reference Design Files -> Example 5

**Clock routing**: See Figure 30.

**Output on the DACs:** Figure 31 shows the result on DAC1 where `CLK_B` was set to 100 MHz and Figure 32 shows the result on DAC2 where `CLK_A` was set to 60 MHz. In both cases, the LEDs flash at the speed related to the oscillator clock; thus, the LEDs flash at the same speed as in example 2. More results for different frequencies of the DIME clocks can be found under the Results folder of this example in the reference design files.

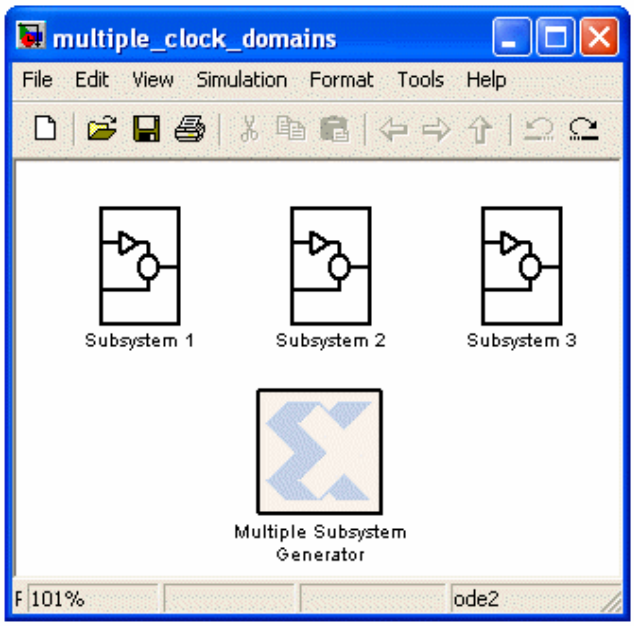


*Figure 28:* **Multiple Clock Domain System Generator for DSP Design Shows How to Use the Different Clocks in one System Generator for DSP Design**


www.BDTIC.com/XILINX

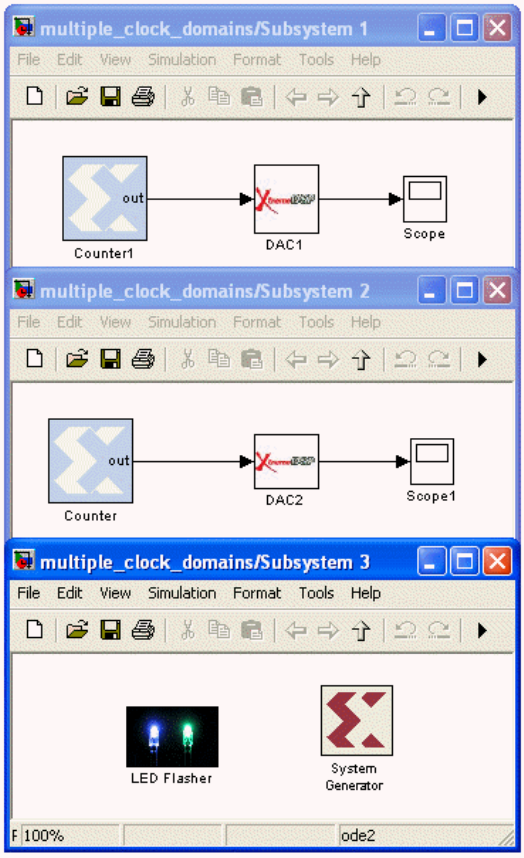


*Figure 29:* **Contents of Three Subsystems in Figure 28 Correspond to Subsystems Used to Describe the Routing for Example Shown in Figure 30**


www.BDTIC.com/XILINX

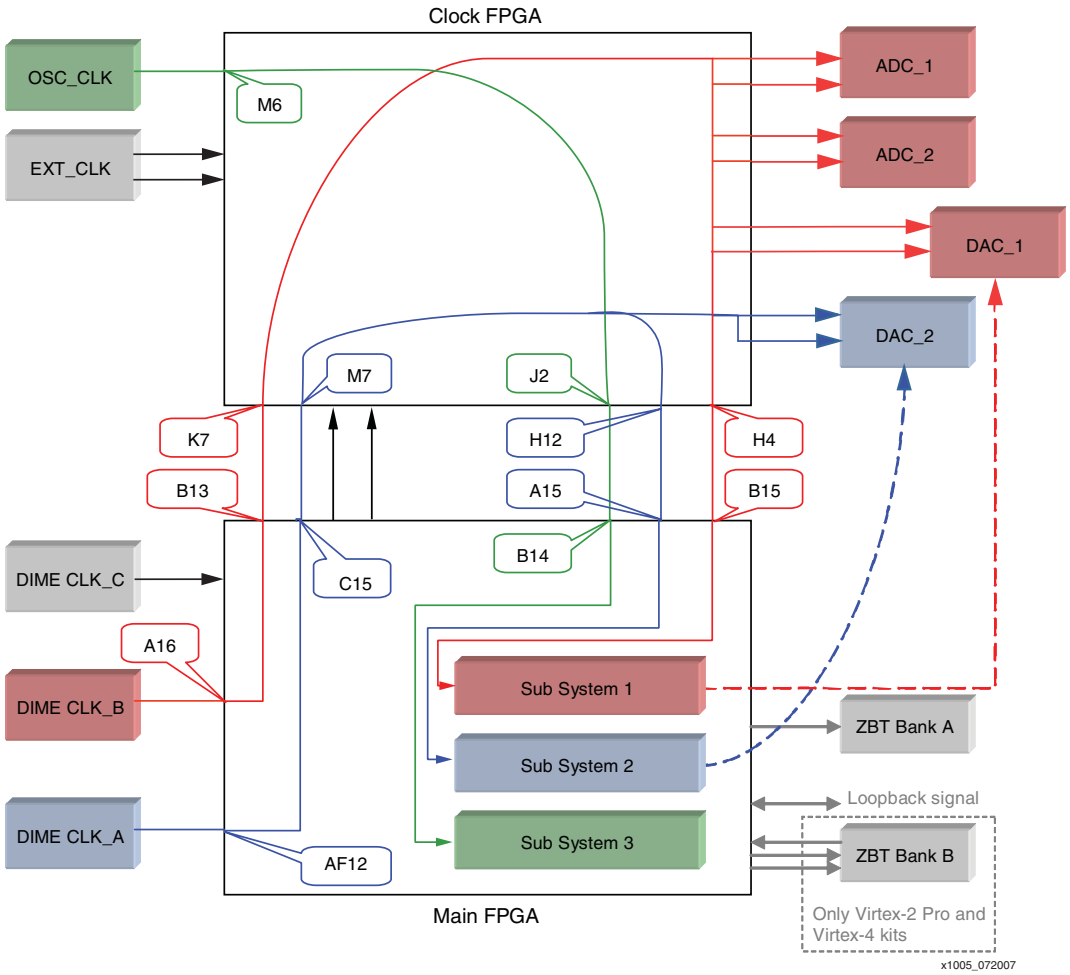


*Figure 30:* **Clock Routing for Example 5**


www.BDTIC.com/XILINX

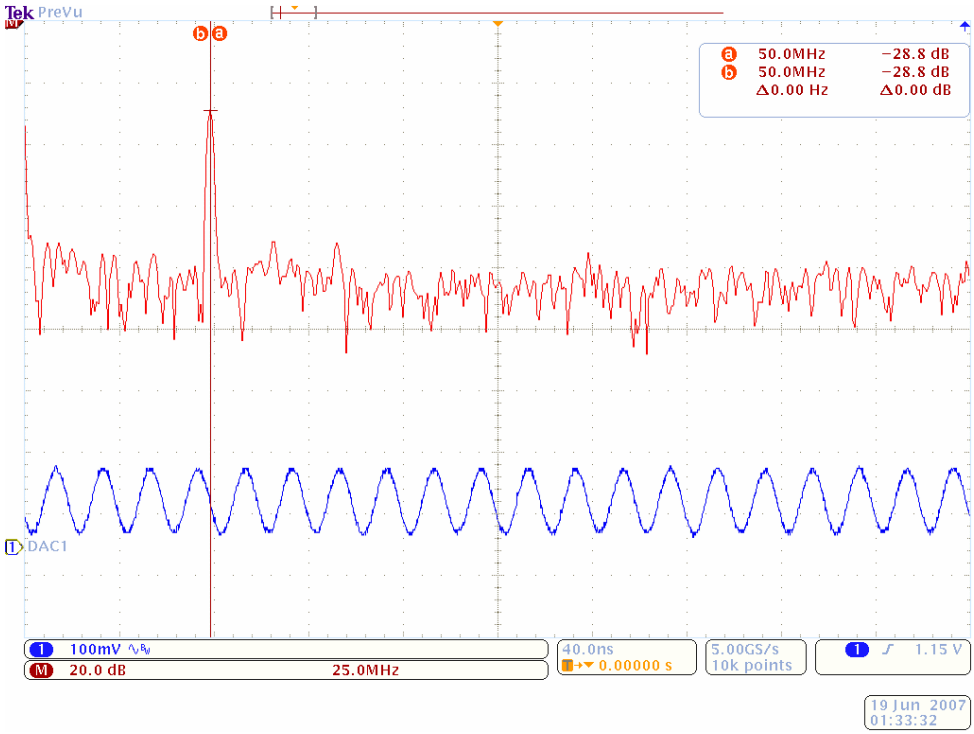


*Figure 31:* **Output on DAC_1 with DIME CLK_B = 100 MHz**

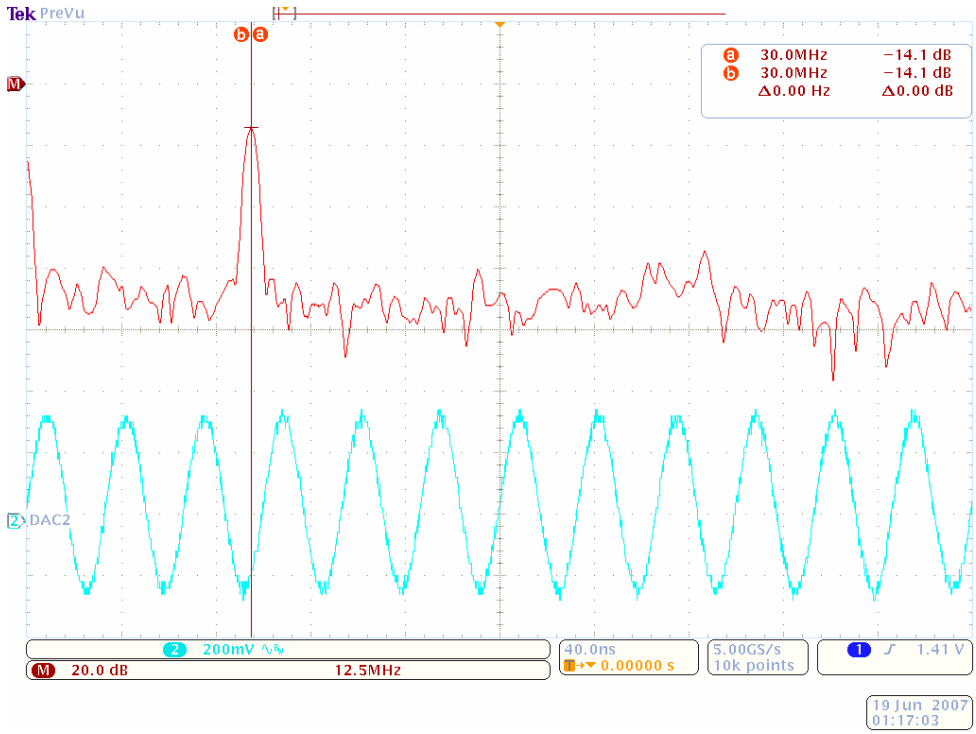


*Figure 32:* **Output on DAC_1 with DIME CLK_A = 60 MHz**


www.BDTIC.com/XILINX

## Reference Design Files

The reference design files are available for download from the Xilinx website at: xapp1005.zip. The ZIP file contains the HW-Cosimulation Example from the "Example Used to Demonstrate HW-Cosimulation Clocking" section and the five examples from the "Examples Using Board Clocking Resources" section. The HW-Cosimulation example consists of:

- System Generator for DSP Model
- The `.ncd` place and routed design along with the physical constraint file (PCF) which can be used to investigate the clocking scheme in FPGA Editor.

Each of the five examples is placed in a separate folder that contains the following files for both the XtremeDSP Development Kit for Virtex-II FPGAs (XC2V3000) and the XtremeDSP Development Kit for Virtex-4 FPGAs:

- Ready to use `.bit` files for both the clock and the main FPGA to demonstrate the operation of every example.
- Source files for the clock FPGA – Consists of a VHDL source file along with a User Constraints File (UCF).
- Source files for the main FPGA – Consists of a System Generator for DSP model which can be used to create the needed `.bit` file. For examples 3, 4, and 5, an archived project directory is included, containing the edited System Generator for DSP designs as described in the examples.
- Results directory – Contains oscillator scope shots for the specific example.

The HW-Cosimulation model and design examples assume the necessary tools are already installed and were tested in the following tool versions:

- System Generator for DSP 9.2i
- MATLAB® 2007a
- ISE 9.2i
- XtremeDSP Development Kit installation for the kit targeted
- Nallatech FUSE Probe v2.10

For more information on the files included with the design notes, see the `readme.txt` file in the downloadable ZIP file.

## Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 09/26/07 | 1.0 | Initial Xilinx release. |
| 10/03/07 | 1.1 | Added Figure 7. Minor edits. |

## Notice of Disclaimer

Xilinx is disclosing this Application Note to you "AS-IS" with no warranty of any kind. This Application Note is one possible implementation of this feature, application, or standard, and is subject to change without further notice from Xilinx. You are responsible for obtaining any rights you may require in connection with your use or implementation of this Application Note. XILINX MAKES NO REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL XILINX BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES ARISING FROM YOUR USE OF THIS APPLICATION NOTE.


www.BDTIC.com/XILINX