XAPP900 (v1.0) September 12, 2005

Getting Started: FPGAs in Motor Control

Author: Frank Wirtz

Scope

This tutorial covers the implementation of a simulated AC Induction motor driver, and is intended to serve as a very basic introduction for new users of Project Navigator.

A Spartan™-3 FPGA is used to create three phases of motor drive using pulse width modulation (PWM) outputs. The techniques demonstrated by this tutorial include:

- waveform table lookup (sine, 3rd harmonic sine, and 60 degree clipped sine)
- PWM output using a pulse centered driver
- programmable deadtime.

Objective

Completion of this tutorial will result in a small hardware demonstration of the motor controller, the creation of three phases of AC waveforms that can be modified (wave type and frequency) and displayed on an oscilloscope. Additionally, users will become familiar with the sub components used to implement an AC Induction Motor driver solution.

System Requirements

Software

The minimum required software component for this tutorial is Xilinx ISE™ v7.1, installed in c:\xilinx.

If the logic simulation is to be run, an operable version of ModelSim XE/SE is required. This project has been tested using ModelSim SE5.8b.

Hardware

Use of this example as a hardware demonstration requires a Spartan-3 starter board from Digilent, Inc. (available from Xilinx), along with a simple R/C 320Hz low pass filter. Other hardware platforms may be used, however, modifications to the design files, such as *.ise and *.ucf, will be required to accommodate the different device / pin out of the design.

In addition, a means of programming the demo board, such as a Xilinx PCIV download cable, and a method for observing the waveform outputs, such as an oscilloscope.

Source Files

Source files for this tutorial may be found accompanying this application note at www.xilinx.com/bvdocs/desfiles/xapp900.zip. This package contains all of the project files needed to implement the motor simulator, and should be unzipped into c:\MotorControl.

Additional information on the logic based design may be found in XAPP448.

Implementation

Initializing the Project

Once the source files have been unzipped into C:\MotorControl, a directory structure similar to the following should have been created:

```
C:\MotorControl
    \Simulation
        \ClockDivider_v1_00_a
```

\Common_Types

\MotorController

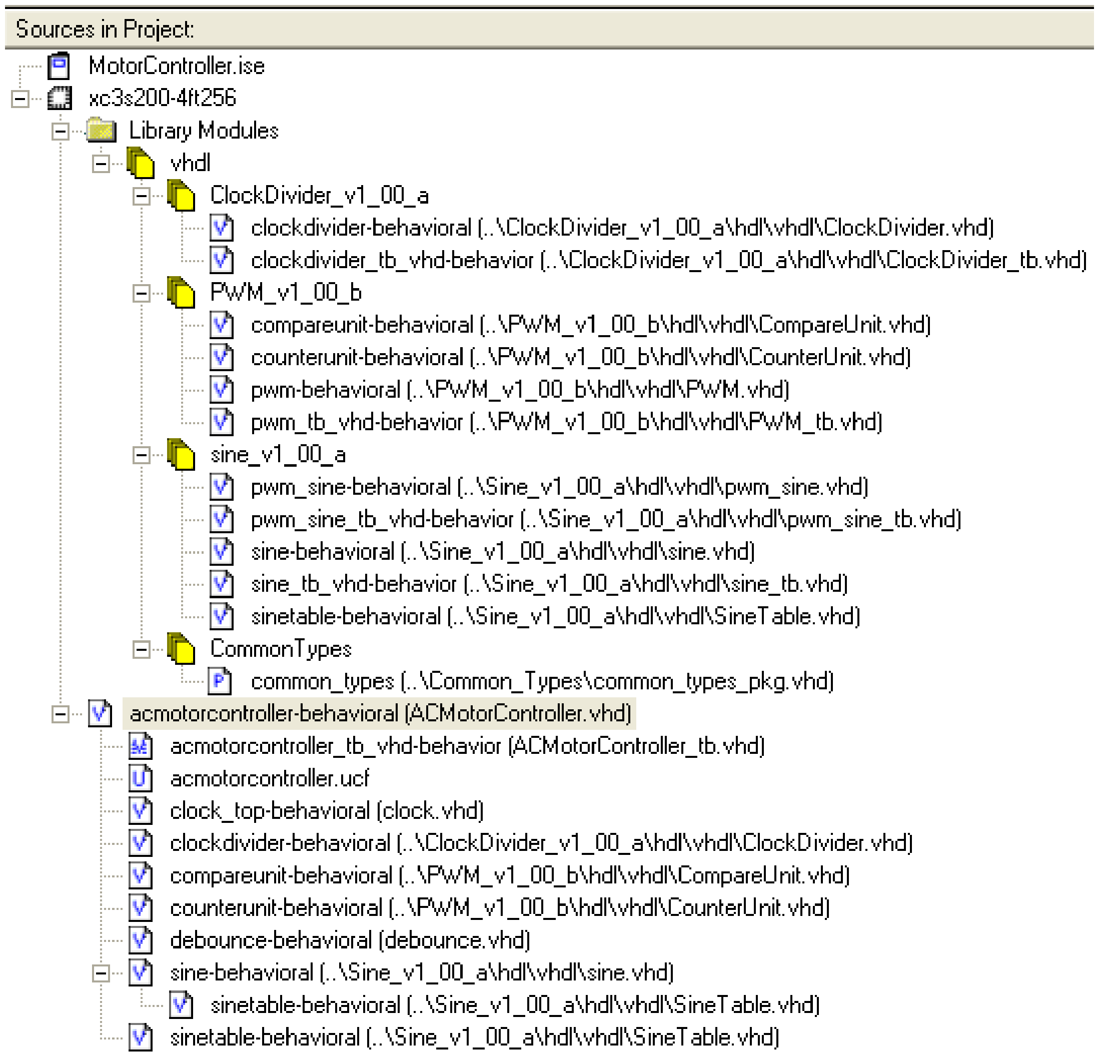
\PWM_v1_00_b

\Sine_v1_00_a

Launch Xilinx Project Navigator, and open the file:

C:\MotorControl\Simulation\MotorController\MotorController.ise

A project with a Module View as shown in Figure 1 will appear.



```
Sources in Project:
MotorController.ise
xc3s200-4ft256
  Library Modules
    vhdl
      ClockDivider_v1_00_a
        clockdivider-behavioral (..\ClockDivider_v1_00_a\hdl\vhdl\ClockDivider.vhd)
        clockdivider_tb_vhd-behavior (..\ClockDivider_v1_00_a\hdl\vhdl\ClockDivider_tb.vhd)
      PWM_v1_00_b
        compareunit-behavioral (..\PWM_v1_00_b\hdl\vhdl\CompareUnit.vhd)
        counterunit-behavioral (..\PWM_v1_00_b\hdl\vhdl\CounterUnit.vhd)
        pwm-behavioral (..\PWM_v1_00_b\hdl\vhdl\PWM.vhd)
        pwm_tb_vhd-behavior (..\PWM_v1_00_b\hdl\vhdl\PWM_tb.vhd)
      sine_v1_00_a
        pwm_sine-behavioral (..\Sine_v1_00_a\hdl\vhdl\pwm_sine.vhd)
        pwm_sine_tb_vhd-behavior (..\Sine_v1_00_a\hdl\vhdl\pwm_sine_tb.vhd)
        sine-behavioral (..\Sine_v1_00_a\hdl\vhdl\sine.vhd)
        sine_tb_vhd-behavior (..\Sine_v1_00_a\hdl\vhdl\sine_tb.vhd)
        sinetable-behavioral (..\Sine_v1_00_a\hdl\vhdl\SineTable.vhd)
      CommonTypes
        common_types (..\Common_Types\common_types_pkg.vhd)
  acmotorcontroller-behavioral (ACMotorController.vhd)
    acmotorcontroller_tb_vhd-behavior (ACMotorController_tb.vhd)
    acmotorcontroller.ucf
    clock_top-behavioral (clock.vhd)
    clockdivider-behavioral (..\ClockDivider_v1_00_a\hdl\vhdl\ClockDivider.vhd)
    compareunit-behavioral (..\PWM_v1_00_b\hdl\vhdl\CompareUnit.vhd)
    counterunit-behavioral (..\PWM_v1_00_b\hdl\vhdl\CounterUnit.vhd)
    debounce-behavioral (debounce.vhd)
    sine-behavioral (..\Sine_v1_00_a\hdl\vhdl\sine.vhd)
      sinetable-behavioral (..\Sine_v1_00_a\hdl\vhdl\SineTable.vhd)
    sinetable-behavioral (..\Sine_v1_00_a\hdl\vhdl\SineTable.vhd)
```

Figure 1: **Project Structure, Module view**

Project Navigator

Once the project has been opened, note that the initial user interface is split into four primary windows: Project, Process, Utility, and Message Console.

Project Window

The Project window indicates the hierarchy of the design, the part type, and the libraries associated with the design. Selecting any of the items (by highlighting with a left mouse click) will enable different operations to be performed on these items.

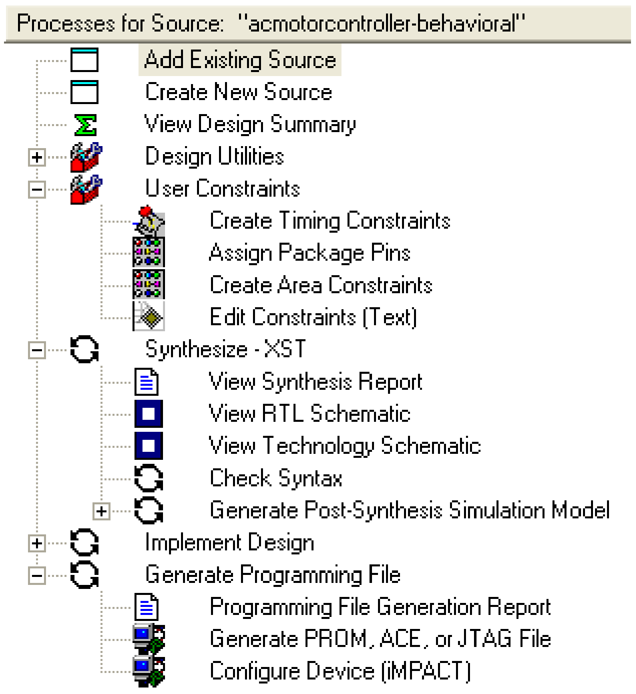
The actual source items that comprise the project may also be edited by double-clicking the file which will open the file in an editor in the Utility window.

Note that the project exists in a hierarchical fashion, with the top level file for the design (`acmotorcontrol.vhd`) existing at the root of the project with the subcomponents listed below. This design is implemented in VHDL, while the source files are indicated by a *.vhd. The test bench files, which provide signal and stimulus information to the simulation tool, are indicated by a *_tb.vhd, while the constraints file, which provides information such as pin location, and other design specific constraints, is indicated by the the extension *.ucf.

Process Window

Whenever a source file is highlighted, the process window will provide a list of available project actions that may be performed on that specific file. The process window also will provide status, indicating which operations have been completed (indicated by a marker of either a green check, or yellow exclamation mark) or those operations that have been attempted but failed due to an error (red “x”).

As an example, highlight the top level VHDL file `ACMotorController.vhd`. Note that in the Process window, a large group of actions are available as shown in Figure 2.



```
Processes for Source:  "acmotorcontroller-behavioral"
   Add Existing Source
   Create New Source
   View Design Summary
+  Design Utilities
-  User Constraints
      Create Timing Constraints
      Assign Package Pins
      Create Area Constraints
      Edit Constraints (Text)
-  Synthesize - XST
      View Synthesis Report
      View RTL Schematic
      View Technology Schematic
      Check Syntax
   +  Generate Post-Synthesis Simulation Model
+  Implement Design
-  Generate Programming File
      Programming File Generation Report
      Generate PROM, ACE, or JTAG File
      Configure Device (iMPACT)
```

Figure 2: **Initial Process Window**

Because this design will be downloaded to the demonstration board, it is necessary to create the programming file bitstream. To create the bitstream:

Highlight **Generate Programming File**, then right click, and select **Run**

or

Double left click on **Generate Programming File** to begin processing the design.

After the project has been run successfully, and a programming file has been generated, the process window should now resemble Figure 3. Note that associated with some processes, some warnings may have been generated. In most cases, these warnings, which are indicated by the yellow exclamation marks, are benign and may be ignored. As an example, a warning may be generated to indicate that portions of subcomponents may not have been used and were subsequently optimized out. Green check marks indicate that the process has completed without error or warning. A red "X" indicates that a process failed due to some error that will require user interaction to resolve.

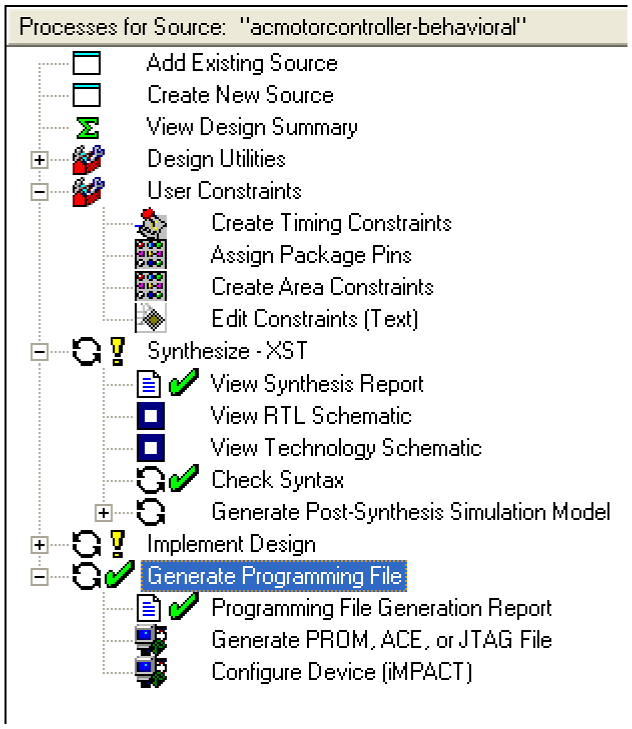

Processes for Source: "acmotorcontroller-behavioral"
Add Existing Source
Create New Source
View Design Summary
Design Utilities
User Constraints
Create Timing Constraints
Assign Package Pins
Create Area Constraints
Edit Constraints (Text)
Synthesize - XST
View Synthesis Report
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Generate Programming File
Programming File Generation Report
Generate PROM, ACE, or JTAG File
Configure Device (iMPACT)

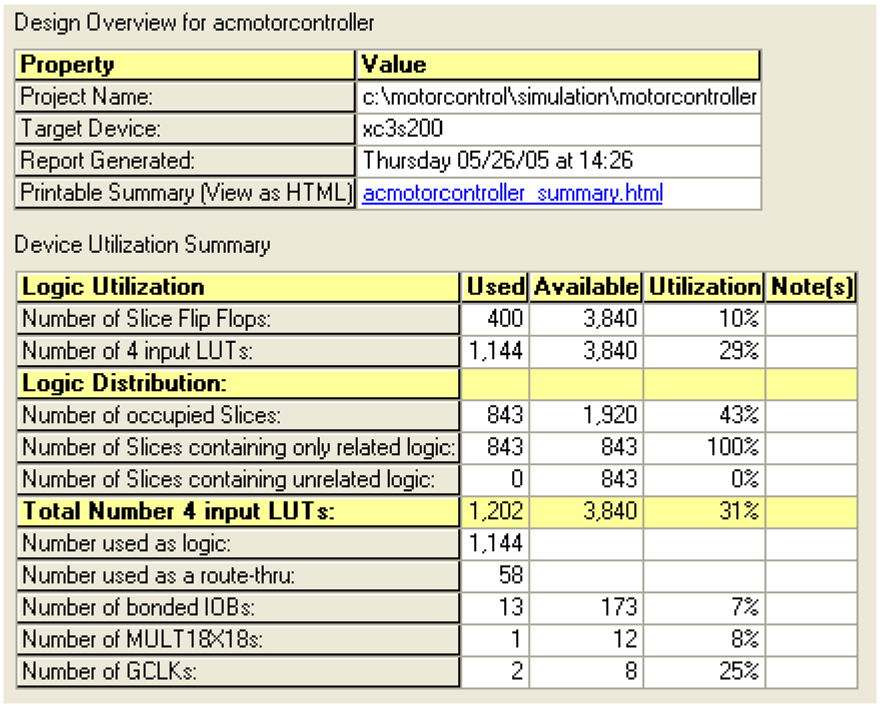
Figure 3: **Post Process Window**

Utility Window

The utility window section of the GUI provides a working space for activities such as editing files and reading reports. Additionally, Project Navigator uses this area to display the overall design summary information. The summary information, which is accessed by clicking on the Design Summary tab, is useful for obtaining a quick overview of the design and the resources required to implement the project. Once the design has been processed to the point that a programming file has been generated, the utility window should resemble Figure 4

Message Console

Note also that at the bottom of the Project Navigator GUI there is a window that provides processing and status messages regarding the project. This window has tabs that easily allows for filtration of the messages. Experiment by selecting the Warnings tab immediately below the console.



Design Overview for acmotorcontroller

Property	Value
Project Name:	c:\motorcontrol\simulation\motorcontroller
Target Device:	xc3s200
Report Generated:	Thursday 05/26/05 at 14:26
Printable Summary (View as HTML)	acmotorcontroller_summary.html

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops:	400	3,840	10%	
Number of 4 input LUTs:	1,144	3,840	29%	
Logic Distribution:				
Number of occupied Slices:	843	1,920	43%	
Number of Slices containing only related logic:	843	843	100%	
Number of Slices containing unrelated logic:	0	843	0%	
Total Number 4 input LUTs:	1,202	3,840	31%	
Number used as logic:	1,144			
Number used as a route-thru:	58			
Number of bonded IOBs:	13	173	7%	
Number of MULT18X18s:	1	12	8%	
Number of GCLKs:	2	8	25%	

Simulation

Knowing that the project has been successfully processed by Project Navigator is not an indication that the design operates as desired. The next step in this tutorial involves performing a behavioral simulation.

Behavioral Simulation

The behavioral simulation of the AC Motor Simulator is easy to accomplish. Within the Source Module window, highlight the test bench which is used to provide the stimulus for the design (`ACMotorController_tb.vhd`). Once this is highlighted, double click the **Simulate Behavioral Model** in the Process window as shown in Figure 5 to start the simulation.

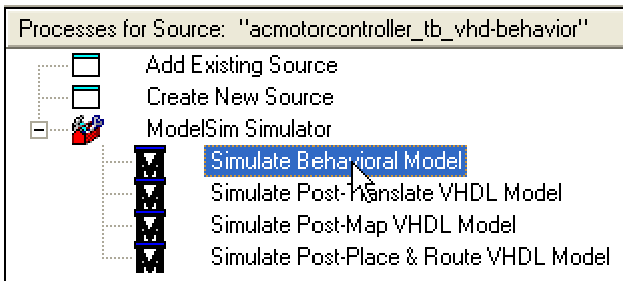

Processes for Source: "acmotorcontroller_tb_vhd-behavior"
Add Existing Source
Create New Source
ModelSim Simulator
Simulate Behavioral Model
Simulate Post-Translate VHDL Model
Simulate Post-Map VHDL Model
Simulate Post-Place & Route VHDL Model

Figure 5: **Simulate Behavioral Model**

Completion of simulation by ModelSim results in four new windows. Select and expand the wave window to view the resulting waveform outputs. Once the window is expanded, access the pull-down menus and select **View>Zoom>Full**. It may take a few moments to regenerate the waveforms.

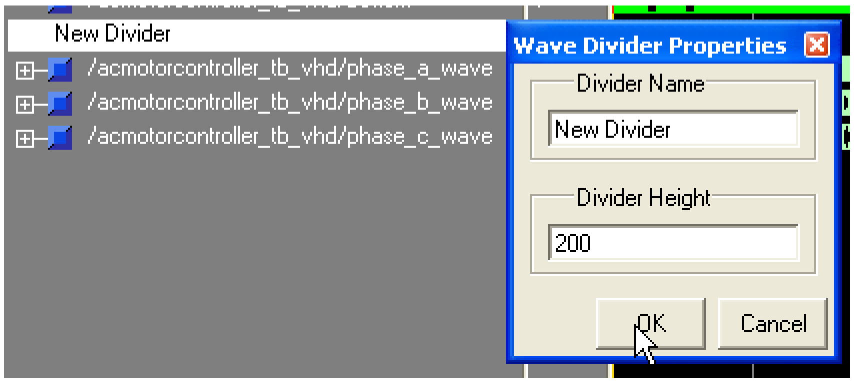
Formatting the Waveform Viewer

Motor Phase Drivers

At the bottom of the waveform list shown in Figure 6, there are three signals titled phase_a, phase_b, and phase_c. These are the output values of the PWM outputs and indicate the respective energy levels of the three motor windings. In the initial format, they are represented by binary values. This radix and format can be modified to show the resulting sine waves generated by the design.

Right click on the phase_a waveform name, and select **Insert Divider**. Set the divider height to 200; this will provide some spacing in the GUI for expanding the waveforms. Inserting a divider will insert a space between the phases and the signals above; in order to observe the waveform signals it will be necessary to now scroll the display downward.

See Figure 6 for additional information.


New Divider
/acmotorcontroller_tb_vhd/phase_a_wave
/acmotorcontroller_tb_vhd/phase_b_wave
/acmotorcontroller_tb_vhd/phase_c_wave

Wave Divider Properties
Divider Name: New Divider
Divider Height: 200

Figure 6: **Insert Wave Divider**

Once the divider has been inserted, modify the three wave buses to display the actual values in a sine fashion. Right click on the phase_a signal, then select **Format / Analog**. Set the multiplier value to 0.3 as shown in Figure 7.

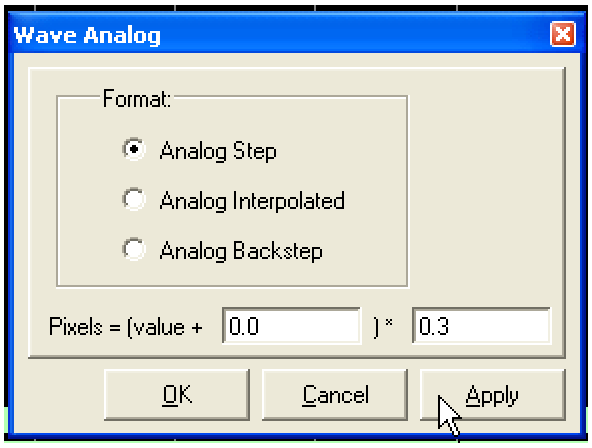

Wave Analog
Format:
[✓ Analog Step]
Analog Interpolated
Analog Backstep
Pixels = (value + 0.0) * 0.3

Figure 7: **Set Wave to Analog**

Next, right click on phase_a again, select **Radix / Unsigned**. Finally, right click on phase_a, select **Properties,** and modify the wave color. Perform the Format / Radix / Color modification on the two remaining phases. Successful completion of the waveform formatting will yield a wave display similar to Figure 8. Note that this simulator is capable of providing three types of waveform drive. The default drive upon startup is a primary plus third harmonic; this design also can produce sine waves and 60˚ clip drive.

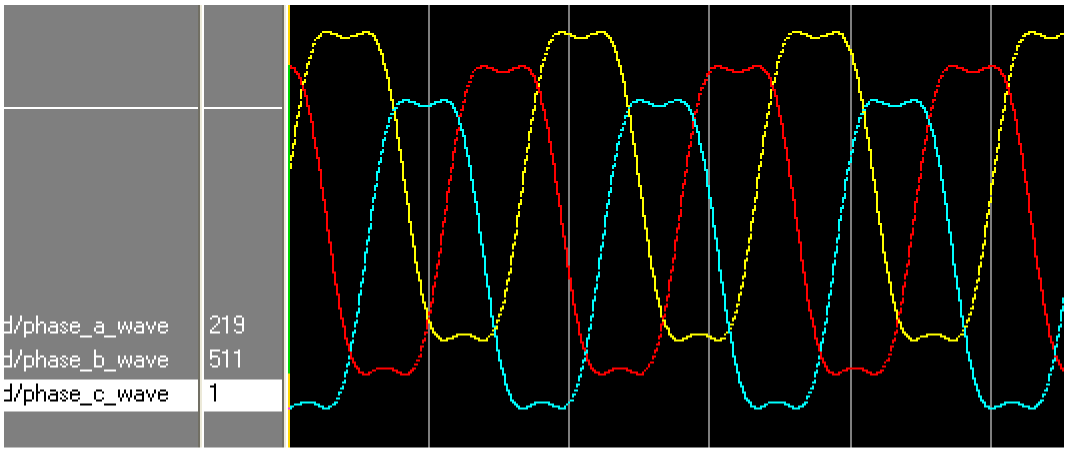


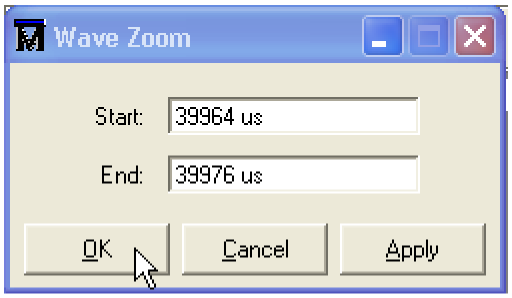
Figure 8: **Post Formatted Waveforms**

Deadtime Observation

This simulator mimics the drive that is provided to a half H-Bridge power driver. These drivers typically consist of two power transistors that provide a high side source and a low side sink. By enabling the high side of one phase driver, and the low side of a different driver, current is supplied through any of phase windings. To avoid catastrophic overcurrents in the power switch devices, it is not desirable to have both the high and low sides of any phase driver on at the same time.

Since power transistors are not capable of instantaneous switching, a small amount of *deadtime* is provided to ensure that both transistors are off before any transistor is enabled. This protects the power stage from being damaged by overcurrent caused by both transistors in one stage being on simultaneously. This can be viewed in the ModelSim waveform viewer by Zooming into the timebase.

From the ModelSim pull-down menus, select **View > Zoom > Zoom Range** and enter a Start value of 39964 s and an End Value of 39976 s and press **OK** to select as shown in Figure 9.

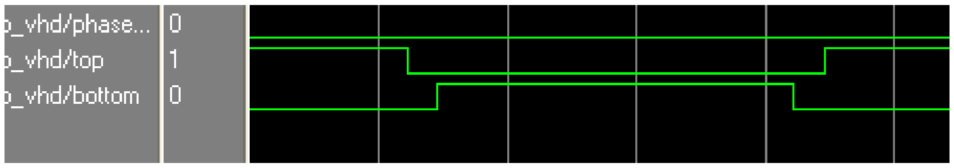


Start: 39964 us

End: 39976 us

Figure 9: **Adjusting Viewing Range of Simulator**

Observe the wave traces for signals *top* and *bottom*. Note that both signals are off for a period of time before either signal is enabled. The waveform should resemble Figure 10. Note that the amount of deadtime is capable of being configured within the project VHDL source.



Additional Simulation Experiments

Once the basic simulation has been run, the user may experiment by modifying the design files and the testbench files to observe the effect on the output of the simulation. Two additional experiments are as follows:

Deadband Modification

Open the file called `ACMotorController_tb.vhd` by double clicking on the filename in the Sources window. Locate in the Generic definition section, the generic called **C_DEADBAND**. Note that this is an integer, and may be any even whole positive number. Experiment with a different value, reprocess the design, simulate, and observe the resulting differences on the waveforms.

Wave Type Switching

This design has the ability to generate three different types of AC signals: Sine, 3rd harmonic Sine, and 60 degree Sine clipped. A simple modification to the test bench file will stimulate the design file to output the three different wave types.

To implement this modification, double click on the file named `ACMotorController_tb.vhd`. Scroll down to the area that has a comment called --place stimulus here, and type the following:

```
wait for 30 ms;
sineSelect <= '1';
wait for 1 ms;
sineSelect <= '0';
wait for 30 ms;
sineSelect <= '1';
wait for 1 ms;
sineSelect <= '0';
```

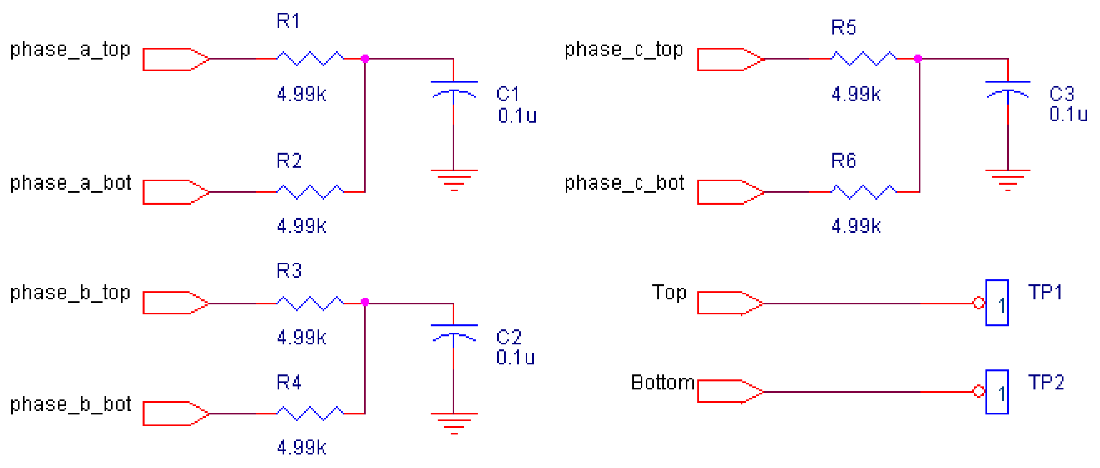
This inputs a stimulus to the design that toggles the sineSelect signal high for one ms and then returns it low. When implementing this design in hardware in the next section of this document, the equivalent stimulus would be pressing and releasing the sineSelect button on the demo board, which will modify the output pattern of the PWM drivers. Implement the above changes to the test bench file, and re-run the simulation to observe the waveform changes.

Hardware Demonstration

Motor Simulator Filters

In order to view the resulting waveforms associated with the motor driver tutorial, a minimum of one low pass filter must be created and placed on the outputs from the Spartan-3 Starter Board. All three phases of the motor controller may be observed by creating the three coil simulators. Additionally, the PWM Top and Bottom signals (from one phase) may be observed as well. Refer to Figure 11 for a schematic of the hardware simulation board.

For the purposes of this demonstration and the following photographs, the filters and test points were constructed using a *Digilent Breadboard 1*. The generated phase signals can be observed by using an oscilloscope to view the voltage across each phase capacitor.



For pinout definitions for the Spartan-3 demo board, please refer to Table 1.

Table 1: **Pinout Definitions for Spartan-3 Starter Board**

FPGA Pin	Spartan-3 Starter Board	Signal Name
T9	GCLK0	clk
L14	BTN3	reset
L13	BTN2	increment
M14	BTN1	decrement
M13	BTN0	sineSelect
D6	A2-5	phase_a_bot
D5	A2-7	phase_a_top
D7	A2-9	phase_b_bot
E7	A2-11	phase_b_top
D10	A2-13	phase_c_bot
D8	A2-15	phase_c_top
B4	A2-17	top
B5	A2-19	bottom

Downloading the Design

Attach the JTAG configuration cable to the Spartan-3 Starter board and to the appropriate port of the computer. Apply power to the Spartan-3 board; the LED on the PCIV cable module should be green.

In the Project Navigator Sources window, highlight the top level file (`ACMotorController.vhd`). Next, in the Processes window, double click on the **Configure Device (iMPACT)** operation. The tool may respond with a warning regarding Jtagclock; select **OK** and continue.

The iMPACT window should open showing a device JTAG chain. Note that two devices appear in the chain; one device is a Spartan-3 FPGA and the other device is a configuration PROM. Because this action will cause the downloading of the configuration file directly to the FPGA, the PROM will be placed in bypass, while the FPGA receives the `acmotorcontroller.bit` file.

To download the bitstream into the FPGA, right click on the XC3S200 device and select **Program**, and then **OK** to initialize download. Note that an option exists to verify the download to ensure that the device was properly configured.

Refer to Figure 12 for an example of programming using the iMPACT utility.

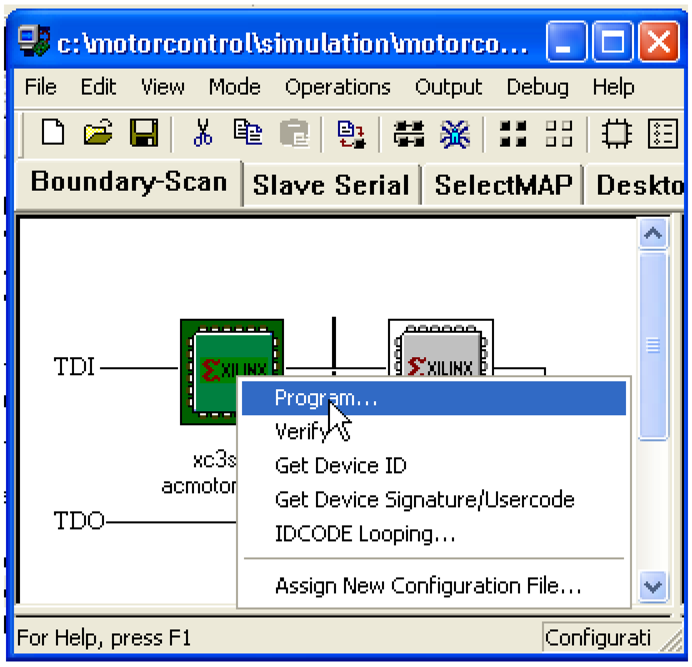
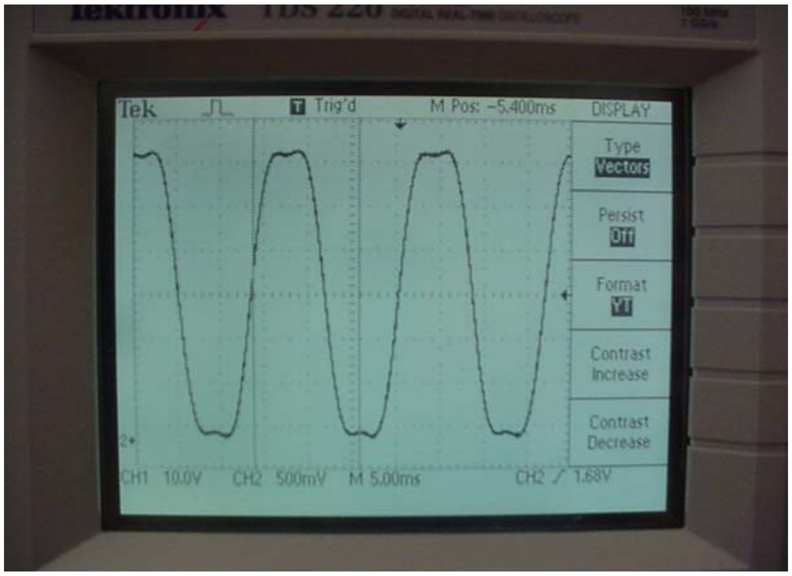

c:\motorcontrol\simulation\motorco...
File Edit View Mode Operations Output Debug Help
Boundary-Scan
Slave Serial
SelectMAP
Deskto
TDI
TDO
XILINX
XILINX
xc3s
acmotor
Program...
Verify
Get Device ID
Get Device Signature/Usercode
IDCODE Looping...
Assign New Configuration File...
For Help, press F1
Configurati

Figure 12: **Programming the FPGA**

Examining the Waveform Outputs

Once the board has been programmed and is driving the R/C loads, examine the waveforms across any or all of the capacitors. The board should come up in a state that creates the third harmonic waveform pattern. Refer to Figure 13 for an oscilloscope image of a third harmonic signal generated from the FPGA outputs.



Pressing BTN0 on the Spartan-3 demo board will vary the waveform from third harmonic, sine, and 60˚ clip.

The default frequency of the waveform is approximately 60 Hz; pressing BTN1 (decrement) or BTN2 (increment) repeatedly will vary the output frequency of the waveform from sub 1Hz to well over 200Hz. Depressing BTN3 will reset the hardware to default conditions.

Observing Deadtime

By attaching two probes to the test points *Top* and *Bottom*, the amount of deadtime between transistors switching may be observed. With a C_DEADTIME generic value of 4, the amount of deadtime inserted is approximately 240ns. See Figure 14 for a scope image of the deadtime.

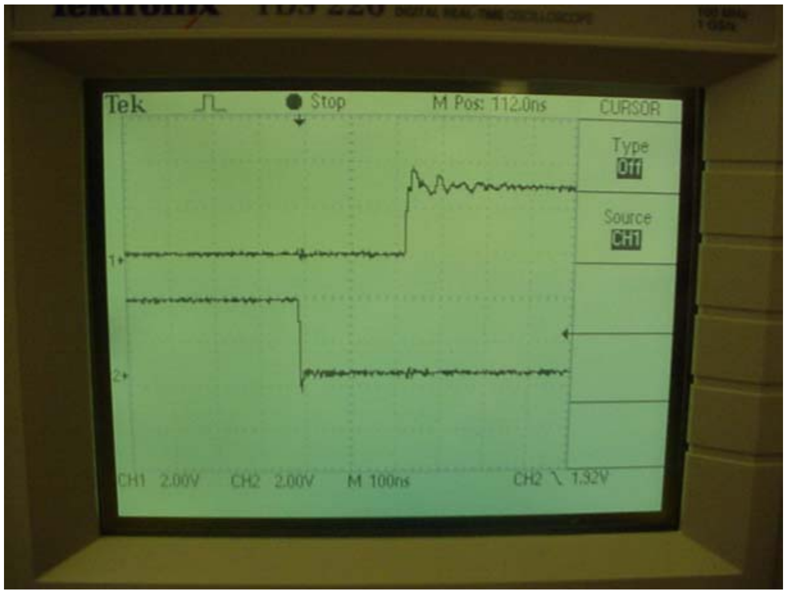


Figure 14: **Scope Shot of Deadtime**

Conclusion

With a small amount of external hardware, a demonstration system can be constructed that will closely simulate the signals associated with driving a three phase AC Induction motor.

For more in depth information on the modules associated with motor control, see *XAPP448 Logic Based AC Induction Motor Controller*.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
8/12/05	1.0	Initial Xilinx release.