Pulse Width Modulator for Hi Tec Servo Motors

Design Description:

This driver is written to work with the Hi Tec HS-635MG servo motor on the Altera DE2 with a 50MHz input clock. However, according to online, the pulse widths are compatible with all Hi Tec servo motors.

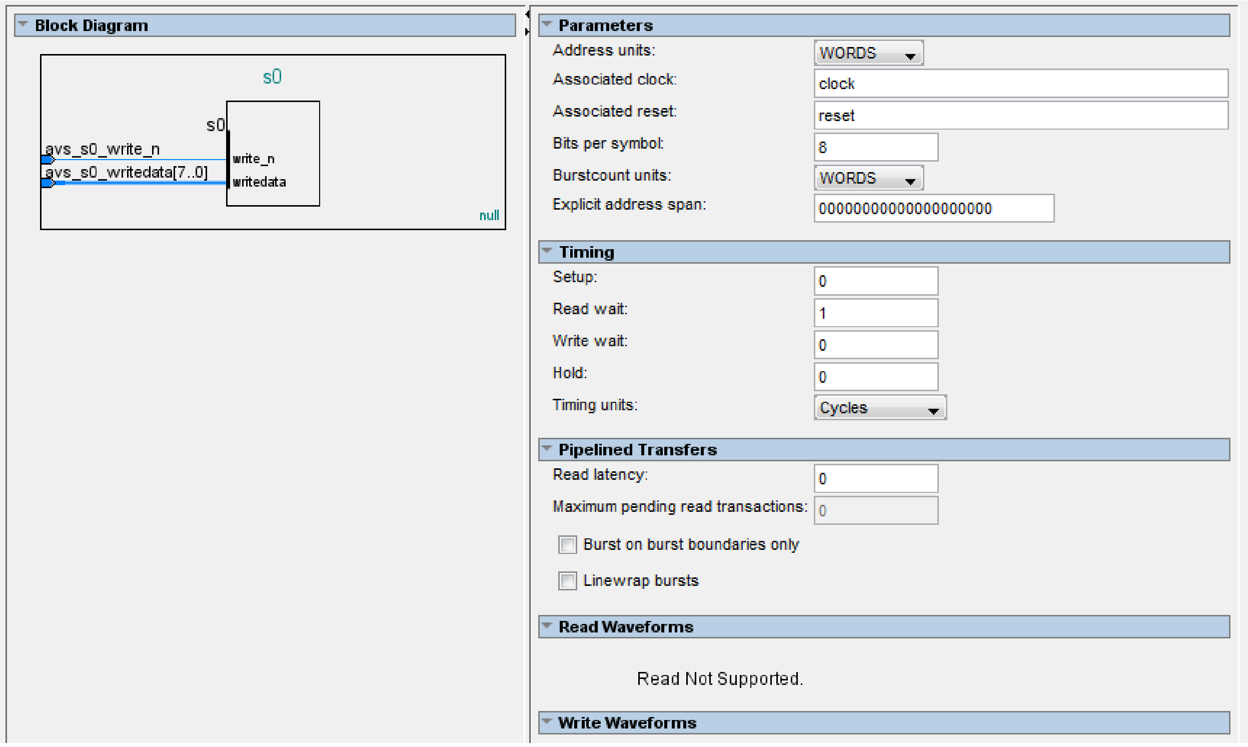
The servo motor being used (HS-635MG) is able to rotate a full 180^{o}, therefore the motor's available pulse widths range from 600µs to 2.3ms. In order to create these ranges, we counted the number of rising edges from the input clock. With an input clock of **50 MHz**, the period is 0.02µs. Therefore, by counting 30000 rising edges, at the same time setting the output to high or '1', a 600µs pulse is created. This is recreated for a pulse width of 2.3ms using a count of 115000.

We decided to output the pulse at a period of 20ms, as this is ideal for the Hi Tec motor. This was implemented by having another variable titled "period" set to count down with every clock rising edge from 10^6 (10^6 * 0.02µs = 20ms) to zero and then reset to 10^6.

Implementation:

Adding the component in *Qsys*:

- Synthesize the supplied VHDL code (*servo_motor.vhd*) and make sure all the "associated reset" fields have "reset" selected under the *Interfaces Tab*
- In order for the code to catch the write_n signal, edit the Timing section as displayed in *Figure 1*.
- Then click *Finish*.
- Click the newly created Component and add it to your Qsys machine.
- Complete the setup by exporting the conduit, connecting the clock, reset and Memory Mapped slave. As seen in *Figure 2*.
- Generate the Qsys System.



Parameters

Address units:	WORDS
Associated clock:	clock
Associated reset:	reset
Bits per symbol:	8
Burstcount units:	WORDS
Explicit address span:	00000000000000000000

Timing

Setup:	0
Read wait:	1
Write wait:	0
Hold:	0
Timing units:	Cycles

Pipelined Transfers

Read latency:	0
Maximum pending read transactions:	0

□ Burst on burst boundaries only

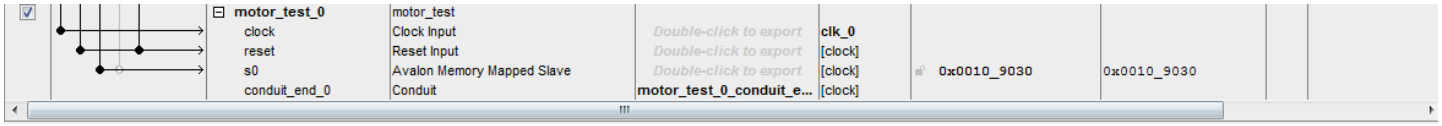
□ Linewrap bursts

Read Waveforms

Read Not Supported.

Write Waveforms

Figure 1: write_n signal timing



☑ ⊟ motor_test_0	motor_test				
clock	Clock Input	Double-click to export	clk_0		
reset	Reset Input	Double-click to export	[clock]		
s0	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0010_9030	0x0010_9030
conduit_end_0	Conduit	motor_test_0_conduit_e...	[clock]		

Figure 2: Connecting Motor Signals in Qsys

Adding the Motor in the VHDL Top-level:

- Find the title of the motor’s conduit title (ex: “motor_test_0_conduit_end_0_export”) and add it to your NiosII_system component as an output with std_logic.
- Then declare the GPIO pin that is going to be used in the entity (ex: *“GPIO_1 : out std_logic_vector(0 downto 0);”*
- Finally, in your port map, connect the GPIO pin to the motor’s conduit (ex: “*motor_test_0_conduit_end_0_export => GPIO_1(0)*”)
- Then compile and Program the DE2.

Running the Motor in Eclipse:

- See the attached “servo_motors.c” file

Possible Modifications:

- To adjust the count to a different Pulse Width:

$$Count = \frac{Desired\ Pulse\ Width\ (s)}{\frac{1}{Clock\ Frequency\ (Hz)}}$$

- This code can also easily be modified to create pulse widths, dynamically, using the 8 bit data input. The input could be a desired pulse width or a value to be assigned to count.