Logic / Home

PWM Generator (VHDL)

Created by Scott Larson, last modified on Jan 09, 2015

Code Download

Version 2.0: pwm.vhd

Transistion between duty cycles always starts at center of pulse to avoid anomalies in pulse shapes

Version 1.0: pwm_v1_0.vhd

Initial Public Release

Features

- VHDL source code of a PWM generator component
- Configurable duty cycle resolution
- Configurable number of outputs/phases
- Configurable PWM frequency
- Modulation around the center of the pulse
- PWM inverse outputs

Introduction

This details a pulse width modulation (PWM) generator component for use in CPLDs and FPGAs, written in VHDL. The component outputs PWM signals based on the duty cycle set by user logic. The center of each pulse occurs at the PWM frequency, and the pulse width varies around the center. If set to multiple phases, the component generates one PWM signal for each phase, evenly spaced. For example, when set to three phases, it generates three PWM outputs 120° out-of-phase with one another. The component was designed with Quartus II, version 12.1 and tested with ModelSim-Altera 10.1b. Resource requirements depend on the implementation. Figure 1 illustrates a typical example of the PWM generator integrated into a system.

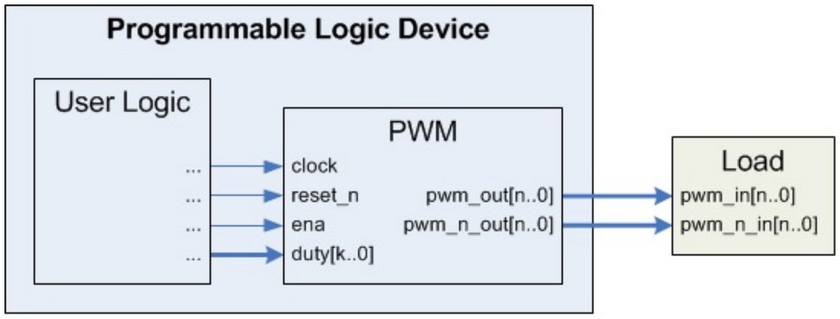

Figure 1. Example Implementation

Theory of Operation

The system clock divided by the PWM frequency equals the number of system clock pulses in one PWM period. Counters define this PWM period for each phase. There is one counter for each PWM phase, with their values offset by the phase. Each counter increments on each system clock and clears once it reaches the end of its period.

The duty cycle determines the points during the period when the PWM signal's rising and falling edges occur. Figure 2 illustrates the basic concept used to determine these positions. The signal's falling edge happens at ½ duty cycle, and its rising edge happens at the end of the period minus ½ duty cycle. Once the counter reaches each of these positions, the PWM signal is toggled as appropriate. Since a half duty cycle can never exceed a half period, the falling edge always occurs before the rising edge.

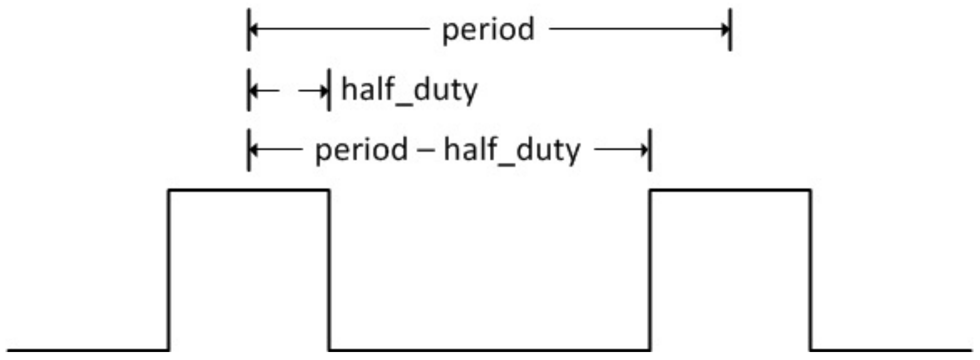

Figure 2. Waveform of a Pulse In-Phase with the PWM Period

Configurable Parameters

The PWM generator is configured using four GENERIC parameters, set in the ENTITY. Table 1 lists the parameters. The PWM generator does not require a specific input clock, so long as the user sets the *sys_clk* parameter to the clock frequency provided. The parameter *pwm_freq* corresponds to the PWM frequency. The *bits_resolution* determines the resolution of the pulse width. For example, a value of 8 provides 8 bits of resolution. Therefore, the pulse width's resolution is 2^8 or 256, so in this case, the finest possible pulse width adjustment is the period (i.e. 1/*pwm_freq*) divided by 256. The parameter *phases* sets the number of outputs and their relation to one another. The number of PWM outputs is *phases*, and these outputs are 360°/*phases* out-of-phase with one another.

Table 1. Generic Parameters

Generic	Data Type	Description
sys_clk	integer	System clock frequency in Hz.
pwm_freq	integer	Frequency of PWM in Hz.
bits_resolution	integer	The number of bits of resolution setting the duty cycle.
phases	integer	The number of output PWMs and phases.

Since the PWM period is defined in system clocks as *sys_clk*/*pwm_freq*, this ratio also affects the duty cycle resolution. A duty cycle resolution is not achievable if it exceeds the number of system clocks in the PWM's period. Similarly, the achieved duty cycle is subject to single bit rounding errors if the period is not an integer multiple of the resolution.

Port Descriptions

Table 2 describes the PWM generator's ports.

Table 2. Port Descriptions

Port	Width	Mode	Data Type	Interface	Description
clk	1	in	standard logic	user logic	System clock.
reset_n	1	in	standard logic	user logic	Asynchronous active low reset.

Port	Width	Mode	Data Type	Interface	Description
ena	1	in	standard logic	user logic	0: PWM continues outputting current duty cycle. 1: latches in the new duty cycle and adjusts the PWM outputs at the center of their pulses.
duty	M*	in	standard_logic_vector	user logic	New duty cycle.
pwm_out	N^	out	standard_logic_vector	load	Output PWM signals. The PWM modulates around the center of the the pulse. The phases are evenly spaced over the period.
pwm_n_out	N^	out	standard_logic_vector	load	Inverse of the PWM outputs.

Notes

* M is the duty cycle's specified resolution in bits, set by the *bits_resolution* generic.

^ N is the specified number of outputs (and phases), set by the *phases* generic.

Controlling the Duty Cycle

User logic can control the duty cycle by latching in new duty cycle values on the *duty* port. The PWM generator latches in values on this port on all system clocks when the *ena* port is set to '1'. Figure 3 shows a ModelSim simulation changing the duty cycle. Note when changing duty cycles that there is one pulse of intermediate width between the original and new pulse widths. This is due to the new duty cycle taking effect in the center of the pulse, so that pulse's width is half of the old pulse width plus half of the new pulse width.

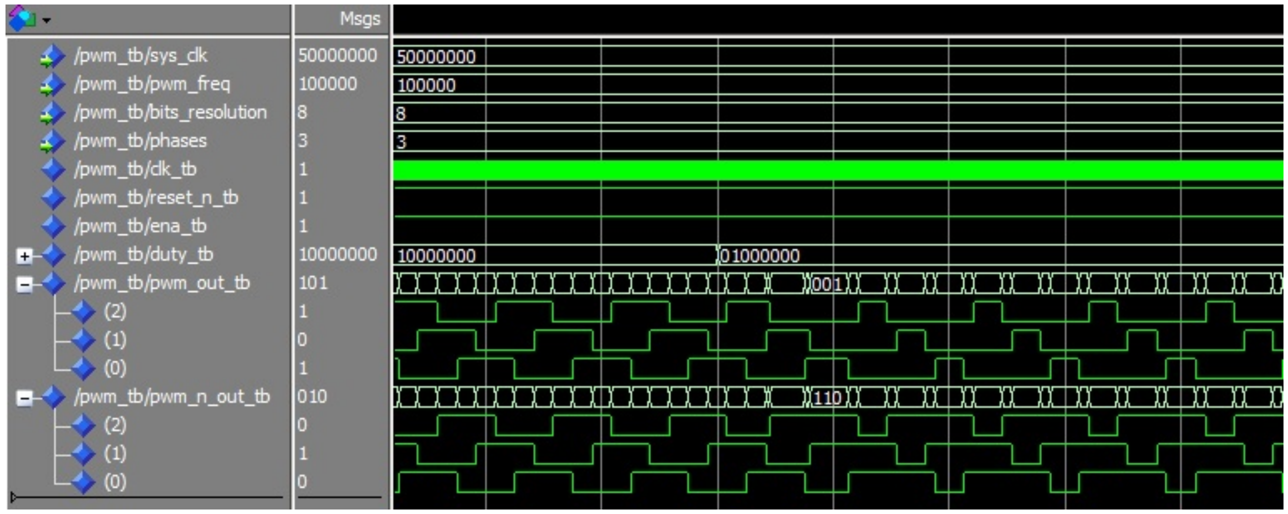

Figure 3. Simulation of Changing the Duty Cycle

Reset

The *reset_n* input port must have a logic high for the PWM generator component to operate. A low logic level on this port asynchronously resets the component. During reset, the component clears the period counters and sets both the PWM outputs and PWM inverse outputs to '0'.

Conclusion

This PWM generator is a programmable logic component that produces PWM and PWM inverse outputs. The system clock, PWM frequency, duty cycle resolution, and number of phases are configurable. An interface to user logic controls the PWM duty cycle.

Feedback for Our Sponsor