PWM Fan Controller

September 2014 | Reference Design RD1060

Introduction

Fans are found in a number of electronic devices such as the laptop in the office and the oscilloscope in the lab. Fans in these devices are usually used as part of a thermal management strategy. By controlling the speed of the fan, various thermal management requirements can be met. A slower running fan can reduce the power consumption and at the same time lower the noise level of the system. A faster running fan, on the other hand, prevents the system from overheating. Many other fan applications can also be found in electronic systems.

There are three typical fans that are available on the market:

- 2-pin (GND, Power)
- 3-pin (GND, Power, Sense)
- 4-pin (GND, Power, Sense, Control)

The open drain sense signal of the 3-pin and 4-pin fans outputs the RPM (rotations per minute) of the fan in the form of a PWM (pulse width modulation) signal. The control signal of a 4-pin fan is a PWM input used to control the speed of the fan. As expected, the more complex the device or the more pins the fan has, the more expensive it is.

Using a FPGA and a MOSFET allows for the speed control of a simple 2-pin fan. When using a 3-pin fan and a FPGA, the sense signal from the fan completes the feedback loop. Below are examples of the 2-pin and 3-pin fans with an FPGA. The advantage of using a low-cost FPGA together with a fan device is that it has sufficient logic resources to create a complete thermal management system. Such a system can carry out functions like monitoring temperature sensors and displaying information on an LCD, in addition to fan speed control.

Figure 1. 2-Pin Fan Control

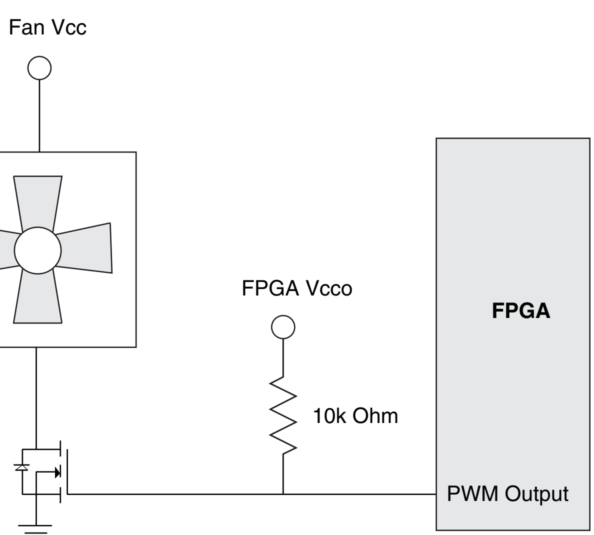
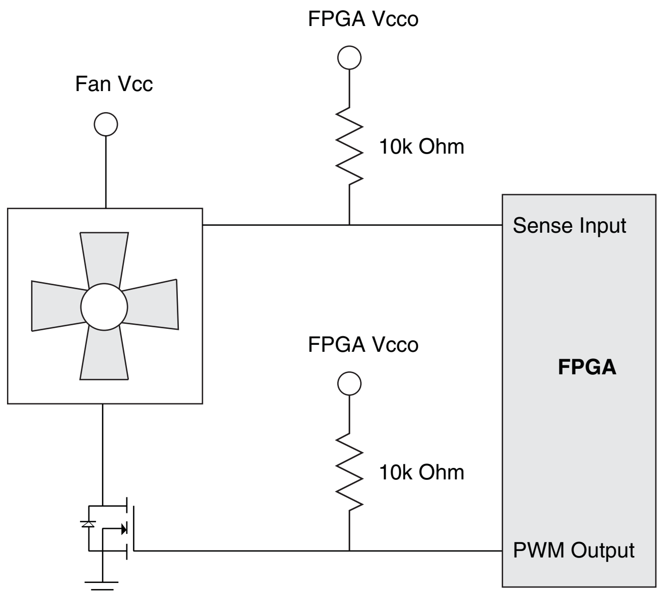


Figure 2. 3-Pin Fan Control



How the PWM Circuit Works

Functionally, the PWM circuit works as follows:

- When the MOSFET is turned on, the current flow through the fan is normal and the fan accelerates.
- When the MOSFET is turned off, the current flow through the fan is impeded and the fan decelerates.

When the MOSFET enables and disables the fan, the ground connection of the sense signal of the 3-pin fan gets distorted. In order to get around this and to get an accurate reading of speed on the sense pin, the following steps are performed and implemented in the design:

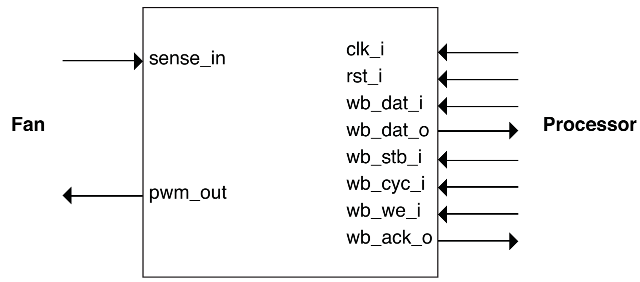
- Enable MOSFET, the current flow through the fan is normal
- Detect the first low-to-high transition which is the beginning of the sense period
- Detect the second low-to-high transition which is the end of the sense period
- Resume normal PWM MOSFET operation

The fan acceleration is negligible during the reading of the sense pin because the MOSFET is enabled for only a short period of time.

Register Transfer Level (RTL) Implementation

The RTL block diagram of the PWM Fan Controller is shown in Figure 3. It consists of one module, fan_controller_wb.v.

Figure 3. Block Diagram



Signal Definitions

Table 1. Signal Definitions

Signal Name	Signal Direction	Active State	Definition
sense_in	Input	N/A	Sense pin of the 3-pin fan
pwm_out	Output	N/A	Used to control the MOSFET
clk_i	Input	N/A	Clock source must run 2X the fastest sense period
rst_i	Input	High	Active high reset signal
wb_dat_i	Input	N/A	The data input during write cycles
wb_dat_o	Output	N/A	The data output during write cycles
wb_stb_i	Input	High	The strobe output signal indicates a valid data transfer cycle
wb_cyc_i	Input	High	When asserted, indicates that a valid bus cycle is in progress
wb_we_i	Input	1 = Write 0 = Read	This signal is negated during read cycles and is asserted during write cycles
wb_ack_o	Output	High	When asserted, indicates the normal termination of a bus cycle

The code has four main sections:

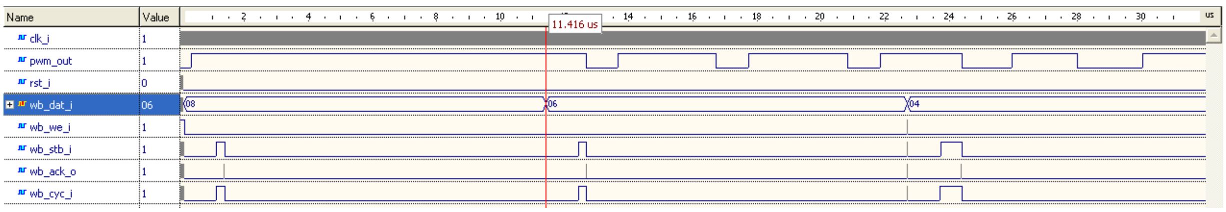
1. WISHBONE interface
   - Reads and writes data to and from the WISHBONE bus.
2. Clock divider
   - The clock is slowed down to operate in the fan's frequency range of Hz.
3. PWM output
   - A high signal enables the MOSFET, speeding up the fan while a low signal disables the MOSFET, slowing down the fan. The MOSFT gate is PWM, based on the desired fan speed.
   - A high-only signal operates the fan at the maximum speed.
   - Depending on the fan, a minimum PWM high time might be require to start the fan revolving.
4. Read the fan's sense pin
   - The MOSFET connects or breaks the GND signal to the fan. If the GND signal to the fan breaks, the sense signal can no longer be read.
   - When a read operation is issued, the fan's GND is connected. The sense pulse width is measured using the design's fast clock, clk_i.
   - During the read operation the increase in fan speed is negligible.

Timing Diagrams

The following timing diagrams show the major timing milestones in the simulation. *Note: To reduce simulation times at the top of fan_controller_wb.v the parameters REG_SIZE (used to measure fan sense pulse) and SCALER (used to slow down clk_i) can be adjusted.*

Shown below is the changing of the fan's speed based on the value from the WISHBONE bus. When wb_dat_i is provided with an 8 the pwm_out runs at maximum speed, pwm_out is high. When the speed is changed to 6 and 4 the fan slows and pwm_out has a lower percentage of time high.

Figure 4. Changing Fan Speed



Shown below is the measuring of the fan speed. The fan's sense signal is generated in the test bench based on the speed. When the wb_we_i signal is low a read operation begins. The signal pwm_out is driven high, the fan's sense signal sense_in pulse width is measured using clk_i. The sense pulse width is outputted to wb_dat_o. As the fan slows the sense_in signal slows as well and the read operation is longer.

Figure 5. Measuring Fan Speed

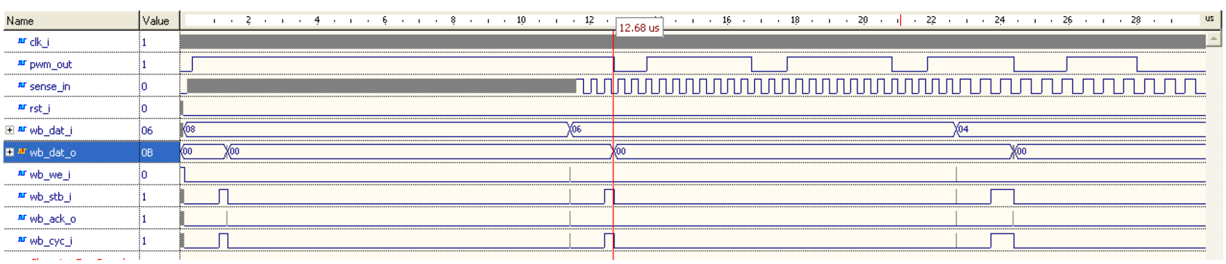
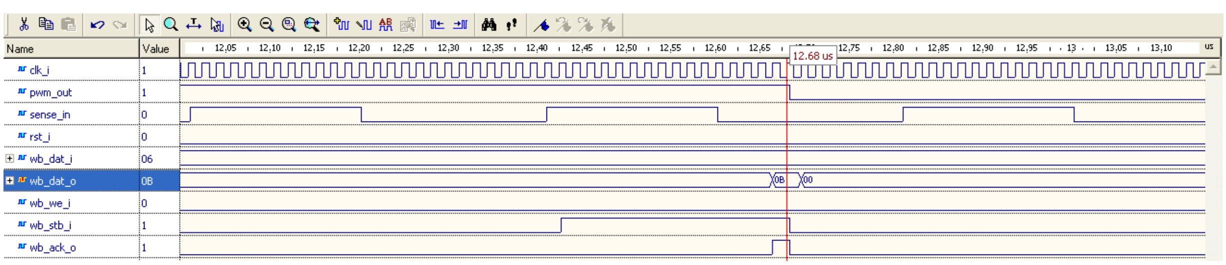


Figure 6. Measuring Fan Speed, Zoomed In



Implementation

Table 2. Performance and Resource Utilization

Device	Language	Speed Grade	Utilization	f_{MAX} (MHz)	I/Os	Architecture Resources
LatticeXP2™ [1]	Verilog-Syn	-5	142 LUTs	>50	24	N/A
	VHDL-Syn	-5	142 LUTs	>50	24	N/A
MachXO3™ [6]	Verilog-LSE	-6	124 LUTs	>50	24	N/A
	Verilog-Syn	-6	108 LUTs	>50	24	N/A
	VHDL-LSE	-6	124 LUTs	>50	24	N/A
	VHDL-Syn	-6	108 LUTs	>50	24	N/A
MachXO2™ [2]	Verilog-LSE	-4	124 LUTs	>50	24	N/A
	Verilog-Syn	-4	108 LUTs	>50	24	N/A
	VHDL-LSE	-4	124 LUTs	>50	24	N/A
	VHDL-Syn	-4	108 LUTs	>50	24	N/A
MachXO™ [3]	Verilog-LSE	-3	115 LUTs	>50	24	N/A
	Verilog-Syn	-3	99 LUTs	>50	24	N/A
	VHDL-LSE	-3	115 LUTs	>50	24	N/A
	VHDL-Syn	-3	99 LUTs	>50	24	N/A
ispMACH® 4000ZE[4]	Verilog	-5 (ns)	61 Macrocells	>50	24	N/A
	VHDL	-5 (ns)	61 Macrocells	>50	24	N/A
Platform Manager[5]	Verilog-LSE	-3	115 LUTs	>50	24	N/A
	Verilog-Syn	-3	99 LUTs	>50	24	N/A
	VHDL-LSE	-3	115LUTs	>50	24	N/A
	VHDL-Syn	-3	99 LUTs	>50	24	N/A
Platform Manager 2[7]	Verilog-LSE	1A	222 LUTs	>50	33	N/A
	Verilog-Syn	1A	222 LUTs	>50	33	N/A
	Verilog-LSE	1A	230 LUTs	>50	33	N/A
	Verilog-Syn	1A	230 LUTs	>50	33	N/A

1. Performance and utilization characteristics are generated using LFXP2-5E_5FT256C, with Lattice Diamond® 3.3 with Synplify Pro®. When using this design in a different device, density, speed, or grade, performance and utilization may vary.
2. Performance and utilization characteristics are generated using LCMXO2-1200HC-4TG100, with Lattice Diamond 3.3 design software with Lattice Synthesis Engine (LSE) and Synplify Pro. When using this design in a different device, density, speed, or grade, performance and utilization may vary.
3. Performance and utilization characteristics are generated using LCMXO2280-3FT25, with Lattice Diamond 3.3 design software with Lattice Synthesis Engine (LSE) and Synplify Pro. When using this design in a different device, density, speed, or grade, performance and utilization may vary.
4. Performance and utilization characteristics are generated using LC4128ZE-5TN100C, with ispLEVER Classic 1.8 software. When using this design in a different device, density, speed, or grade, performance and utilization may vary.
5. Performance and utilization characteristics are generated using LPTM10-124-3G128CES, with Lattice Diamond 3.3 design software with Lattice Synthesis Engine (LSE) and Synplify Pro. When using this design in a different device, density, speed or grade, performance and utilization may vary.
6. Performance and utilization characteristics are generated using LCMXO3-4300C-6BG256C with Lattice Diamond 3.3 design software with Lattice Synthesis Engine (LSE) and Synplify Pro. When using this design in a different device, density, speed or grade, performance and utilization may vary.
7. Performance and utilization characteristics are generated using LPTM21-1AFTG237C with Lattice Diamond 3.3 design software with Lattice Synthesis Engine (LSE) and Synplify Pro. The utilization number includes additional logic required by ASC-Interface. When using this design in a different device, density, speed, or grade, performance and utilization may vary.

Technical Support Assistance

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
July 2009	01.0	Initial release.
October 2009	01.1	Added support for ispMACH 4000ZE device family.
January 2010	01.2	Added support for LatticeXP2 device family.
		Added VHDL support for all device families.
November 2010	01.3	Added support for MachXO2 device family and Lattice Diamond design software.
December 2010	01.4	Added support for Platform Manager device family.
		Added support for Lattice Diamond 1.1 and ispLEVER 8.1 SP1 design software.
March 2014	01.5	Added support for MachXO3L device family.
		Updated Technical Support Assistance information.
September 2014	1.6	Updated Table 2, Performance and Utilization. Added support for Platform Manager 2 device family. Added implementation of VHDL for both LSE and Synplify Pro. Added support for Diamond 3.3. Added support ispLEVER Classic 1.8. Product name/trademark adjustment.