Hardware PWM Generator

General Description

A Pulse Width Modulation (PWM) signal generator works by varying the duty cycle of a square wave while keeping the period fixed. The hardware PWM generator was designed for implementation on a Xilinx XC4005XL FPGA. It contains circuitry that allows easy interface to an 8051 microcontroller.

The hardware PWM generator provides a 10-bit user selectable period and a programmable dead zone to prevent the PWM signal and its complement to be active at the same time, an important feature in H-bridge/motor applications.

Applications

Motor speed control
Switching power supply
Communications
Dimmers
Heat control

Pin Configuration

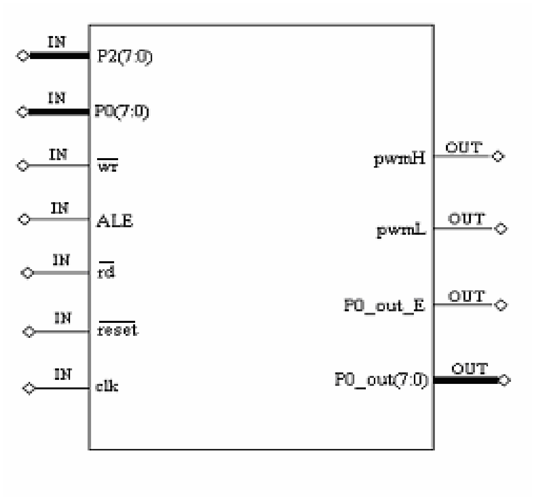

Figure 1. Pin configuration

Features

- **Built-in 8051 interface hardware**
- **10-bit user selectable period**
- **Complementary outputs provided for use in H-Bridge applications**
- **4-bits programmable dead zone counter**

Contact Information

Dr. Daryl Beetner
University of Missouri-Rolla
Department of Electrical and Computer Engineering
1870 Miner Circle
Rolla, MO-65409-1060

http://www.ece.umr.edu/courses/cpe214

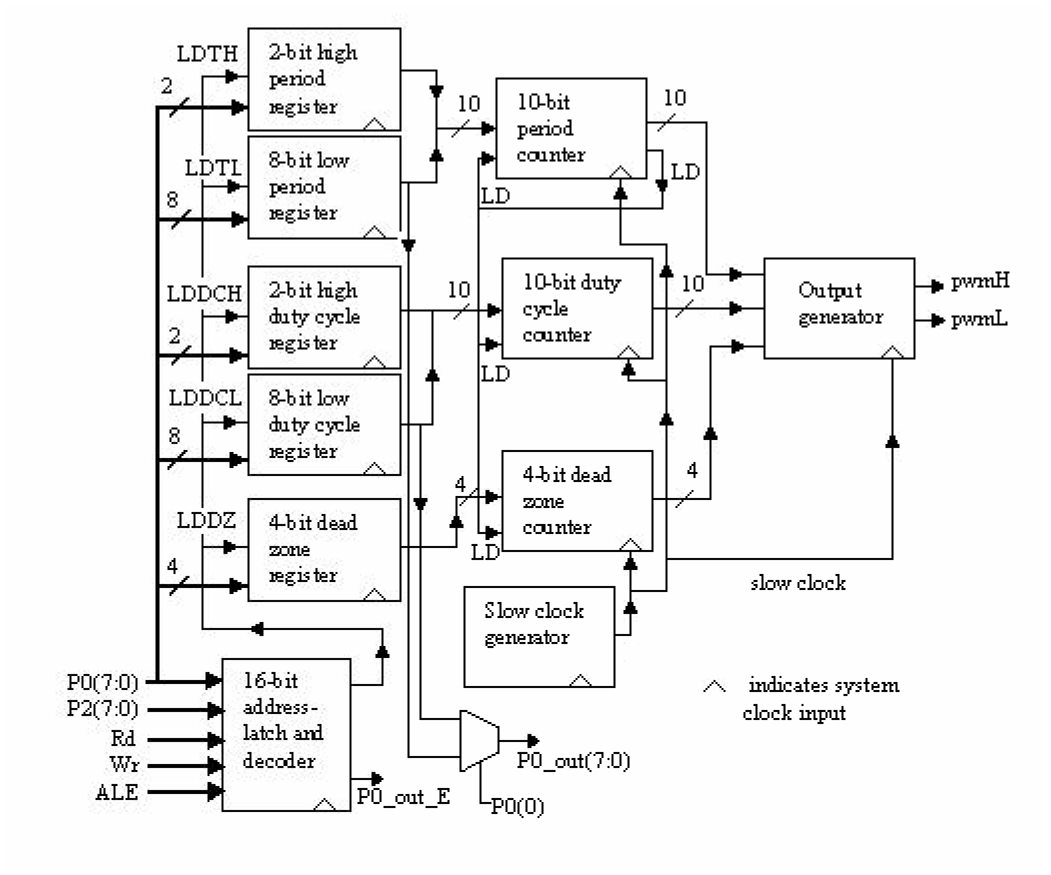

Figure 2. Functional diagram

PIN Name	Description
P0(7:0)	Multiplexed Address/Data bus
P2(7:0)	High Byte address bus
ALE	Address Latch Enable – address is latched on falling edge
Clk	System clock
RD/	Active low read enable
WR/	Active low write enable
RESET/	Active low reset
PwmH	PWM output
PwmL	Complementary PWM output
P0_out(7:0)	8-bit output used for testing the model
P0_out_E	Enable for tri-state buffer

Detailed Description

Model Function

Registers

The hardware PWM generator consists of several registers and counters as shown in Figure 2. Two 2-bit registers and two 8-bit registers are used to store the 10 bit period and duty cycle. A 4-bit register is used to store the programmable dead-zone value. These registers are written to through an 8051 interface, in which Port 0 is an 8-bit multiplexed address-low/data bus and Port 2 is the upper 8 bits of the address bus.

Decoder

The decoder is used to map registers within the 8051's external address space. The registers locations are shown below.

Address	Register
0xFFFB	Low 8 bits of period
0xFFFC	High 2 bits of period (least significant bits used)
0xFFFD	Low 8 bits of duty cycle
0xFFFE	High 2 bits of duty cycle (least significant bits used)
0xFFFF	4-bit programmable dead zone (only least significant 4 bits used)

Counters

The period, duty cycle and dead zone counters are countdown counters, which are loaded with the values from their respective registers. These counters are used for timing inside the model. They count down to zero, then stop and wait until the period counter reaches zero, at which point all counters are reloaded with the current values in their respective registers. This helps to prevent abrupt changes in PWM output. Changes to the period and duty cycle thus take effect upon the next reload of the period counter.

Slow Clock Generator

The clock to the counters is provided from a slow clock generator, such that a low frequency PWM signal can be generated. Using the main 12MHz system clock to generate a low frequency signal would make the PWM too large to synthesize easily.

Output Generator

The output generator is responsible for generating pwmH and its complementary (pwmL) signal. The pwmH signal is set high as the period and duty counters begin to count down from their maximum values. Once the duty cycle counter reaches zero, the pwmH signal is set low, and the dead zone counter begins to count down. Once the dead zone counter reaches zero, the pwmL signal is set to high. When the period counter reaches the value stored in the dead zone register, the pwmL signal is set to low. When the period counter reaches zero, the pwmH signal is set high again. This way the pwmH and pwmL signals are never active at the same time.

8051 Interface

The hardware PWM generator output can be easily controlled with an 8051. P0, P2, ALE, RD/, and WR/ are simply connected to the associated signals on the 8051. Figure 3 shows the appropriate connections within the XC4005 schematic.

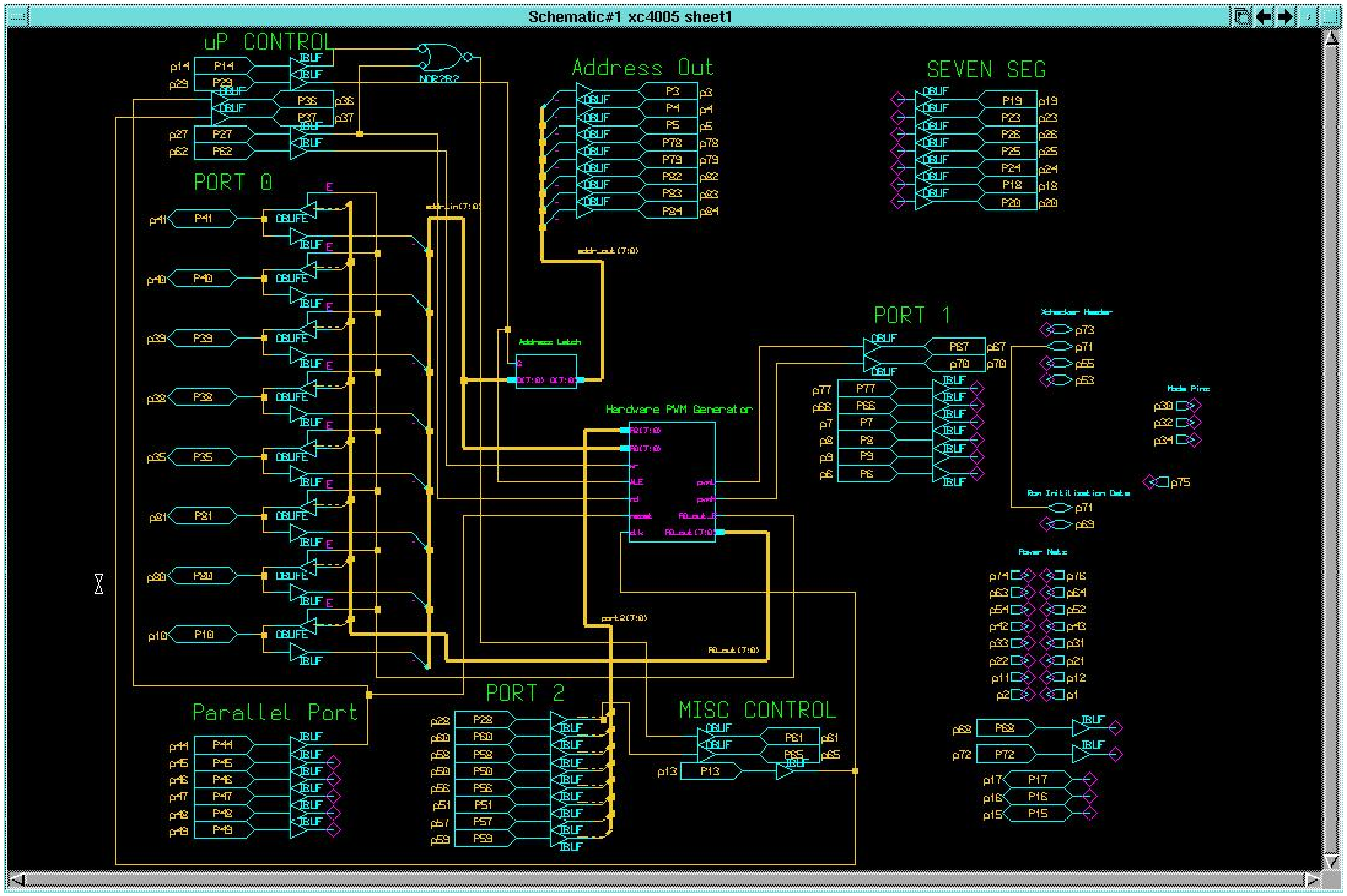

Figure 3. XC4005 schematic with hardware PWM generator

Below is a sample C program for the 8051 to control the duty cycle of the hardware PWM generator using push-buttons. The pushbuttons are connected to ground and to pins P1^0, P1^1, and P1^2.

```
#include <reg51.h>
#include <absacc.h>
#define BASE 0xfff8                   // base address
#define TL XBYTE[BASE+3]              // low 8 bits of period (0xFFFB)
#define TH XBYTE[BASE+4]              // high 2 bits of period (0xFFFC)
#define DCL XBYTE[BASE+5]             // low 8 bits of duty cycle (0xFFFD)
#define DCH XBYTE[BASE+6]             // high 2 bits of duty cycle (0xFFFE)
#define DZ XBYTE[BASE+7]              // 4 bits of dead zone (0xFFFF)

#define uchar unsigned char
#define uint unsigned int
#define CLKPERIC 12                   // 12 for regular 8051, 6 for 89C51Rx2's
#define FCLK 12                       // clock frequency in MHz

// In hardware (XC4005 schematic) PWM signal is connected to port 1 pin 5 as shown in
// Figure 3

sbit INC=P1^0;                        // input pin, if zero increase the duty cycle
sbit DEC=P1^1;                        // input pin, if zero decrease the duty cycle
sbit STOP=P1^2;                       // input pin, if zero stop generating PWM signal,
                                      // by making duty cycle zero

// this function is used to get rid of push buttons bouncing
/* delay for t msec.  Use timer 0 */
void msec(uint t){
#define T1000 (-1000+22)*FCLK/CLKPERIC
 TMOD=(TMOD&0xF0) | 0x01;             /* 16bit timer mode */
 while (t>0) {                        /*delay 1 msec */
  TH0= (T1000) >> 8;                  /* upper half of -1000 (0xfc) */
  TL0= (T1000) & 0xff;                /* lower half of -1000 (0x18) */
  TR0= 1;                             /* start timer 0 */
  while (~TF0);                       /* wait for TF0=1 */
  TR0= 0;                             /* stop timer and clear overflow bit */
  TF0= 0;
  t=t-1;
 }
}
```

```
void main(void){

 TL= 255;                    // fix the frequency of the PWM signal to 256 * slow clock
                             // period
 TH= 0;                      // high 2 bits of period register
 DZ= 1;                      // dead zone=1
 DCH=0;                      // high 2 bits of duty cycle
 DCL=100;                    // initial value of duty cycle
 STOP=1;                     // inputs pulled high initially
 INC=1;
 DEC=1;

 while(1) {
         if (~STOP){
                 DCL=0;                      // stop generating PWM signal
                 msec(700);                  // delay to take care of push-button bouncing
         }
         else {
                 if (~INC) {
                         DCL=DCL+5;          // increment duty cycle
                         msec(700);          // delay to take care of push-button bouncing
         }
                 else {
                         if (~DEC) {
                                 DCL=DCL-5; // decrement duty cycle
                                 msec(700);  // delay to take care of push-button bouncing
                 }
         }
         }
}                                            // end of while
}                                            // end of main
```