Pulse Width Modulator

1.0 Highlights

This section of the manual contains the following topics:

- Introduction
- IO Ports
- PWM Clock generator
- Duty Cycle Register
- Terminal Count (Period Width) Register
- Programming the PWM

1.1 Introduction

A Pulse width modulator (PWM) is a circuit which modulates a signal's duty cycle. This is usually done to convey either information over a communication channel or to control the amount of power sent to a load, such as a DC motor or LED. An analog signal has a continuously varying value, with infinite resolution both in time and magnitude. A PWM is used to digitally encode analog signal levels by applying voltage over controlled intervals of time. Through the use of counters, the desired duty cycle can be modulated to encode a specific voltage or current level. The duty cycle of a signal is the amount of time in the signal period that the pulse is high; this is usually specified as a percentage of the signal period.

The following are some of the key features of this module:

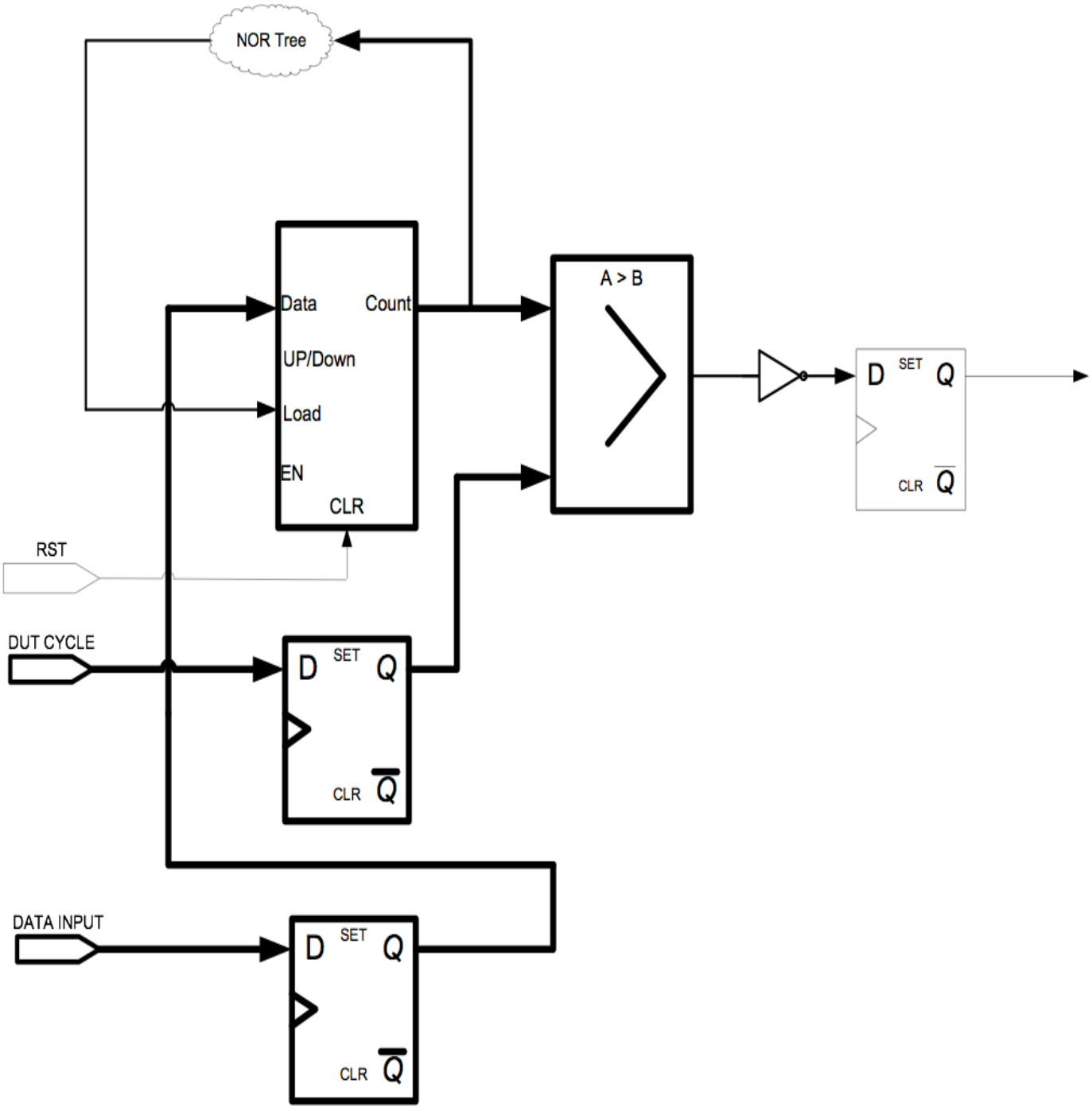
- Software programmable clock frequency
- Software programmable duty cycle
- Static synchronous design
- Fully synthesizable

1.2 Terms and Definitions

Table 1-1

Term	Definition
PWM	Pulse Width Modulation

PWM Module


NOR Tree
Data
Count
UP/Down
Load
EN
CLR
A > B
D
SET
Q
CLR
Q
RST
DUT CYCLE
D
SET
Q
CLR
Q
DATA INPUT
D
SET
Q
CLR
Q

1.2 I/O Ports

Port	Width	Direction	Description
RST	1 bit	Input	Synchronous reset active high.
CLK	1 bit	Input	System Clock.
CS	1 bit	Input	The PWM Chip Select input is active high and used in conjunction with the read and write strobes to enable register read and write operations.
WREN	1 bit	Input	The WREN input pin enables a write transaction. When asserted high in conjunction with WREN chip select the module latches the data on the WRDATA port into the register or sub-module targeted by the address.
RDEN	1 Bit	Input	
WRDATA	15 bits	Input	PWM module data input port.
RDDATA	15 bits	Input	PWM module data output port.
ADDRESS	4 bits	Input	This input vector is used to decode the internal and external addresses for PWM device, configuration registers selection.
CLK_EN	1 bit	Input	This input pin is used to enable the pulse width modulators main counter. It can be used to divide the input clock frequency down to acceptable ranges for the slower ICSs feed by the PWM.
PWM_EN	1 bit	Input	This input pin is used to enable the pulse width modulator.
PULSE	1 bit	Output	Modulating pulse output.

2.0 Functional Description

The PWM IC allows software to control the voltage flow to external analog devices. The PWM can be programmed to output a pulse where the width of the pulse can be modulated to any integer percentage of the output frequency. The modulation of the output signal is used to encode the voltage regulation value for external devices. Both the pulse width (duty cycle) and frequency can be programmed by software. The output pulse frequency is controlled by two factors: the PWM Terminal Count and the frequency of

the input CLK_EN signal. The PWM Terminal Count can be thought of as a maximum value for the internal counter.

3.0 Memory Mapped Registers

The PWM module has memory mapped registers that are accessible by software through the I/O ports. Below is a table listing all of the software accessible registers.

Table 3-1

Relative Offset	Register Name	Size	Read Write
0x00	PWM Terminal Count Register	15 bits	W
0x01	PWM Duty Cycle Register	15 bits	W

3.1 TERMINAL COUNT REGISTER

Table 3-1

Bits	Field Name	Description	Read/Write
31:8	-	Reserved	-
7:0	TCCNT	The Terminal Count register holds the terminal value for the up counter. It gives the software the ability to change the frequency of the outgoing signal by changing the termination count. The counter counts from zero to the value stored in this register, once it reaches this value it is then reloaded with the value 1 and the count starts over.	W

3.2 DUTY CYCLE REGISTER

Table 3-2

Bits	Field Name	Description	Read/Write
31:8	-	Reserved	-
7:0	DUTY CYCLE	The Duty Cycle register is a 32-bit software loadable register which holds the desired duty cycle for the pulse signal. The value placed in this register is compared with the current count of the counter. If the current count is less than the value in this register, the pulse output will be high; otherwise, it will be low.	W

4.0 Programming the PWM

4.1 Configuring the PWM Clock Enable Generator

The PWM uses a clock enable (CLK_EN) signal to control the counter. This signal should be provided externally and can be used to control the effective frequency of the PWM counter.

That being said to configure the PWM one must first configure the appropriate frequency for the dual modulus generator's clock enable. Use the following steps to configure the PWM's pulse frequency.

1. Determine the appropriate dual modulus clock enable frequency.
2. Use the app provided with the SoC to determine the appropriate configuration parameters for the dual modulus clock generator. For information on programming the dual modulus clock generator please see the Clock Divider Specification guide.
3. Taking into account the dual modulus generator's clock enable frequency, use the following equation to determine the value to be written to the PWM's Terminal Count configuration register.

   divisor_value = (dual_modulus_frequency / (desired_baud_rate * 2)) – 1

4. Write the "divisor_value" to the PWM's Terminal Count configuration register.

4.2 Setup for the PWM

Use the following steps to set up the PWM:

1. Ensure that an appropriate clock enable signal is provided to the CLK_EN input.
2. Load the Terminal Count register with the appropriate value to achieve the desired output frequency.
3. Load the Duty Cycle register with the desired duty cycle value.
4. Set the PWM_EN signal high to enable the PWM operation.

Remember to keep the clock resolution in mind when choosing terminal count and duty cycle values.

Example Program:

```
Source_CLK = 50000000; // 50Mhz clock
*PWM_Clock = Source_CLK / 1000000 / 2; // Setting the PWM to a 1us resolution
*PWM_TC_300hz = 1000000 / 300; //Setting the signal period to 300 Hz

*pwm _dutycycle = 1500; // Setting the current duty cycle to 1.5 microseconds.
```