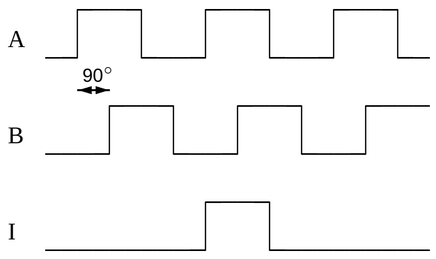
By:
Michael Vestergaard Jessen
Jimmy Alison Jørgensen

1 Quadrature encoder

1.1 Introduction

This IP module is for interfacing to encoders that supply a quadrature encoded signal. The IP module decodes quadrature signals. A quadrature encoder uses two output signals (A and B) to indicate both position and direction of rotation. Some encoders also supply an I output that indicates one hole revolution. Below is the A, B and I signals pictured.



A and B are 90° phase shifted, where A leading B indicates a rotation in clockwise direction and if B leads A it is a counter-clockwise rotation. I gives one pulse per revolution.

Each IP module can have up to 8 quadrature encoder interface instances, where each instance has 4 read-only registers:

Pulse count: Signed value containing the position of the encoder. Given as the number of A pulses since started where a clockwise rotation increases the count and a counter-clockwise rotation decreases.

Revolution count: Similar as the above, but with the number of I pulses instead.

Speed: Number of clock ticks between 2 up going flanks of A.

Acceleration: The difference between last speed measurement and current speed measurement.

The clock used for speed and acceleration is given as a input to the IP module. This should normally be set by the user to the sys_clk_pin net.

1.2 Design constraints

1.3 Features

- 32-bit OPB slave interface.
- Up to 8 quadrature encoder interface instances.
- Up to 32 bit pulse count, revolution count, speed and acceleration registers.
- Up to 32 bit clock counter used for speed.

1.4 Design implementation

The IP module is implemented with the following generics:

- C_QE_INST: Number of Quadrature Encoder instances
- C_QE_PULSE_COUNT_WIDTH: Width of pulse counter

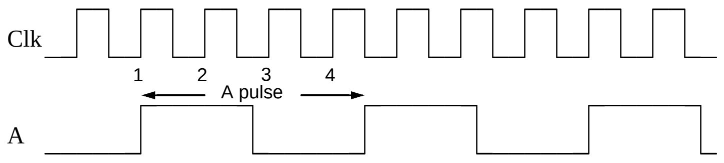
- C_QE_REV_COUNT_WIDTH: Width of revolution counter
- C_QE_SPEED_WIDTH: Width of the speed register
- C_QE_ACCELERATION_WIDTH: Width of the acceleration register
- C_QE_CLK_COUNT_WIDTH: Width of the timer used for speed
- C_QE_NUM_COUNT: Only valid if C_QE_SPEED_TYPE is set to 0, see below. Determines how many A pulses that should be detected before setting speed register to the number of clock ticks between the C_QE_NUM_COUNT number of A pulses.
- C_QE_SPEED_TYPE: Determine the method to measure speed. If set to 0 the speed is the number of clock ticks between C_QE_NUM_COUNT number of A pulses. The speed is set to 0 when the timer overflows. If set to 1 the speed is the number of A pulses in a given time interval - the interval is:

  $$2^{(\text{C\_QE\_CLK\_COUNT\_WIDTH}-1)} \text{ clock ticks}$$

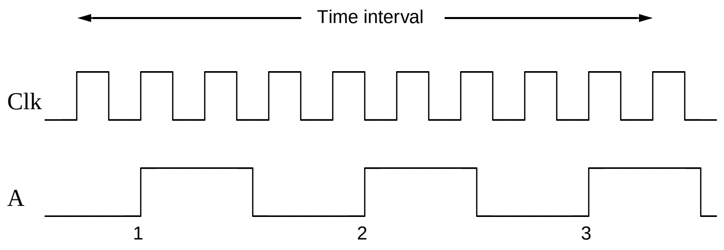
  since the timer counts on the clock and is signed.

If speed type is set to '0', clock ticks between A pulses, the width of speed should be no less than the width of the timer since it is directly derived from it. If speed width is lesser, information can be lost. The acceleration is the change in speed and can therefor have a width less than the timer and speed if needed.

If speed type is set to '1', A pulses in time interval, the width of the speed does not have to be wider than the maximum number of A pulses that can occur in the time interval specified by the timer width.
Speed type = 0:



Speed type = 1:



1.4.1 Converting speed to $\frac{rad}{sec}$

Speed type = 0:
with 512 counts per turn and a fpga clk of 50Mhz

```
r=(2*pi)/512 = rad per count
s=(1/50Mhz) * clk ticks per A pulse = time per A pulse
rad/sec = r/s
```

Speed type = 1: