TDC Instruction

Content

[**I. Introduction 1**](#_Toc31636)

[**II. The Clock & Reset block 2**](#_Toc22490)

[**III. The TDC Fine block 7**](#_Toc21333)

[*A.* Wave Union Launcher 8](#_Toc11186)

[*B.* DSP48E1 delay lines 8](#_Toc3871)

[*C.* Multi-edge detection module 11](#_Toc30844)

[*D.* Multi-edge encoder module 12](#_Toc13805)

[**IV. Configuration.v 13**](#_Toc13230)

# Introduction

1. Architecture

The brief architecture of a TDC consists of a Clock & Reset and a TDC Fine block, the diagram of such architecture is illustrated in Fig.1. The Clock & Reset block is employed to generate a 172MHz system clock, reset signals for all modules, and a 172.023MHz pulses for calibration. The TDC Fine block, composing of the Wave Union Launcher and the Encoder Block, is the core of this architecture. In addition, The top module (tdc.v) has 4 input signals, i\_clk, i\_reset, i\_trigger (unused), i\_control (unused), and a output signal o\_dout[15:0], where the o\_dout output the measurement results of TDC fine part. More details of these modules will be described later.

Note: The project is based on Vivado2019.2, and the top module is tdc.v .

TDC (tdc.v)

TDC Fine block (tdc\_fine.v)

Wave Union Launcher

(wave\_union.v)

Encoder Block

(encoder.v)

i\_trigger (input)

i\_control (input)

o\_dout [15:0]

(output)

Clock & Reset block (crg.v)

pulse\_gen

clk\_wiz\_2

i\_clk (input)

i\_reset (input)

clk\_wiz\_1

clk\_wiz\_0

Fig.1. The architecture of the TDC.

The i\_control is planned to be used as a manual control signal in order to switch calibration mode and normal work mode. Due to the limited time, we have not realized this function. Therefore, the circuit always maintains the auto-calibration mode. The i\_trigger should be connected to the pulse to be measured, but in this case, we didn’t use it.

图表, 瀑布图

描述已自动生成

Fig.2. The schematic of the TDC in Vivado2019.2.

# The Clock & Reset block

This module has two functions. The first function is to generate TDC module clock and reset signal, and the second one is to generate a 172.03MHz pulse into TDC Fine block as a calibration signal. The interconnection of the block can be shown in Fig.3, which instantiated three “Clocking Wizard” IP with different configurations, seen in (Fig.7 ~ Fig.12). The i\_clk signal is connected to the onboard system clock in Zedboard (100MHz), and then clk\_inst0 uses it to generate 2 100MHz output clocks as the clock sources for clk\_inst1 and clk\_inst2. An advantage of placing clk\_inst0 here is that Vivado can route and wire easily. The clk\_inst1 uses the clk\_out1 (100MHz) to produce a 172MHz system clock (o\_clk) for TDC Fine block, while clk\_inst2 utilizes clk\_out2 to generate a 172.03MHz and then output to pulse\_inst block.

图示

描述已自动生成

Fig.3. The schematic of the crg\_inst in Vivado2019.2.

The main purpose of pulse\_inst module is frequency division. If the 172.03mhz pulse is input into the DSP delay line, it may cause one pulse to be sampled multiple times in the DSP delay line (Fig. 4). Generally, the total propagation delay time of the DSP delay line will be larger than the period of 172.03MHz calibration pulse, which will cause the pulse sampled multiple times (Fig.4 (A)). Our solution is to subdivide the calibration pulse. On one hand, the pulse can maintain the phase the same as the original calibration pulse. On the other hand, it will not be sampled multiple times.

Sample points

DSP delay line

172MHz pulse without subdivided

BIN1

BIN2

BIN3

(A)

Sample points

DSP delay line

172MHz pulse with subdivided factor 6

BIN1

BIN2

BIN3

(B)

Fig.4. The comparison of calibration pulse based on different subdivided factors.

The pulse\_gen.vhd is written by VHDL rather than Verilog, other files are written by Verilog. We can modify the subdivided factor by changing the MAXCOUNT (Fig. 6).

图片包含 文本

描述已自动生成

Fig.5. The Subdivided factors (MAXCOUNT) in pulse\_gen.vhd.

It is noteworthy that the o\_stop signal needs to be passed BUFG before outputting to Flip-Flop. In that case, we can ensure that the o\_stop signal reaches all Flip-Flop at the same time as possible. However, the method will cause a larger phase difference between the o\_stop and the o\_pulse because using BUFG will increase the length of wiring. An available solution to minimize the phase difference is to insert a serial of CARRY4 as delay components to o\_stop signals, seen in Fig.6.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Fig.6. Inserting delay components

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Fig.7. The configuration of clk\_wiz\_0 (Board)

电脑软件截图

描述已自动生成

Fig.8. The configuration of clk\_wiz\_0 (Output Clocks)

图形用户界面, 文本, 应用程序

描述已自动生成

Fig.9. The configuration of clk\_wiz\_1 (Board)

图形用户界面, 文本, 应用程序

描述已自动生成

Fig.10. The configuration of clk\_wiz\_1 (Clocking Opitons)

图形用户界面, 应用程序

描述已自动生成

Fig.11. The configuration of clk\_wiz\_1 (Output Clocks)

图形用户界面, 应用程序

描述已自动生成

Fig.12. The configuration of clk\_wiz\_2 (Output Clocks)

# The TDC Fine block

The project mainly focuses on building the fine counting part, which is composed of 5 independent components, a wave union launcher, single delay line implemented by several DSPs, two-level registers, a multi-edge detection encoder block, and a calibration module, seen in Fig.13. The purpose of the wave union module, which is realized through many CARRY4 cascades, aims to produce multi-pulses into the delay line, and the functionality of the two-level register array is to reduce the impact of data competition and risk. The “multi-edge detection & Sum of the position of edges” module also called multi-edge encoder can be configured to different working patterns, detecting rising edge, detecting falling edge, and any edges. And the positions of these edges in the delay line are encoded into 8421 codes to facilitate subsequent calculation. (CLK1 = CLK0 currently)

图示

描述已自动生成

Fig.13 The structure of TDC Fine part

## Wave Union Launcher

The launcher consists of 5 CARRY4 components with different configurations. One of reasons choosing the CARRY4 is that has an extremely short propagation delay initially designed for constructing Carry Lookahead Adder (CLA). In addition, each CARRY4 contains four 2 to 1 multiplexers (MUX), four selection signals (S0, S1, S2, S3), a carry input (CIN), a carry output (COUT) and four data outputs (CO0, CO1, CO2, CO3). When the selection signal is 0, the multiplexer will output the signal of channel 0, and when the selection signal is 1, it will output the signal of channel 1. The trigger signal is directly connected to the CIN port of the first CARRY4, and the COUT port of each CARRY4 is connected to the CIN port of the next CARRY4. Such a configuration pattern can minimize the propagation delay between CARRY4 blocks so as to avoid too long a pulse width. Furthermore, the S0 port of the last four CARRY4 is connected to the trigger signal as a switch that decides whether the signal from the previous stage can be transmitted to the next stage. And the bypass input ports (DI0) of the first multiplexer in the last four CARRY4 are assigned to 1, 0, 1 and 0 respectively to initial the state of the launcher. In this project, the launcher can be flexibly extended by attaching more CARRY4 components. For instance, a 4 positive edge and 3 negative edges ISR WUL is composed of 7 CARRY4. (The Wave Union Launcher can be extended by adding CARRY4 components)

图示

描述已自动生成

Fig.14 The schematic of Wave Union Launcher (wave\_union.v)

## DSP48E1 delay lines

In the same bin count delay chains, the shorter the propagation delay, the higher the accuracy TDC can achieve. Commonly, ASIC has a shorter transmission delay compared with FPGA, which is owing to shorter interconnection wiring in ASIC. Therefore, in contrast to CARRY4 delay line, constructing a delay line based on integrated DSPs in FPGA can achieve a smaller bin width. In the project, the single delay line is composed of 16 DSPs with a total of 768 output ports (768 bins). Except for the first DSP48E1, others are configured into a 48bit post adder to sum the values of CARRYCASCIN and input C (Fig. 15) without any pipelines (Fig. 16), where the input signal C has 48 bits, and CARRYCASIN has only one bit. Finally, the result of their addition will be output to P[47:0] and the carry data will be assigned to CARRYCASOUT.

Note: The first DSP48E1 use the C + CARRYIN to replace the C + CARRYCASCIN in Fig.15, and other parameters maintain unchanged.

图形用户界面, 表格

描述已自动生成

Fig.15 The DSP configuration 1.

图形用户界面

描述已自动生成

Fig.16 The DSP configuration 2.

The structure of the delay line is presented in Fig. 17. It is worth noting that the first DSP of the delay line must use the CARRYIN port to connect the output of the WUL instead of the CARRYCASIN. Because the CARRYCASIN is used to connect previous level DSP’s CARRYCASOUT. If the output of WUL is directly connected to CARRYCASIN, then the Vivado will inform a synthesis error. The CARRYCASIN of the other DSPs are connected to the CARRYCASOUT of the previous stage to form a delay chain. On the other hand, the 48 bits input C of each DSP should be set with 48’b1111…1111 in order to generate carry to the next stage when the CARRYCASIN is high. At first, without any trigger into CARRYIN, the 768 ports will output 1. When a signal inputs to the CARRYIN port of the first DSP, the output of the carry line will successively change from 1 to 0 from left to right. However, the adder inside the DSP may have a carry-lookahead, which will lead to output jump and dislocation. In other words, it may occur a case that the Nth output port is changed to 0 before the (N-1)th output port. The total length of the delay line is 768 bins, and a more length line can be achieved by appending the DSP components.

DSP48E1

CARRYIN

CARRYCASOUT

P[47:0]

DSP48E1

CARRYCASIN

CARRYCASOUT

P[47:0]

DSP48E1

CARRYCASIN

CARRYCASOUT

P[47:0]

DSP48E1

CARRYCASIN

CARRYCASOUT

P[47:0]

Fig.17 The structure of the DSP delay line.

文本

描述已自动生成

Fig.18 The code of the DSP delay line (tdc\_fine.v).

## Multi-edge detection module

Multi-edge detection is a significant step to read out the position of the edges because the design of the module determines whether the data can be obtained accurately. A common issue is how to avoid “data bubble”. In the ideal cases, the data of the DSP delay line is recorded through two-level registers and is transmitted into the detection. Usually, the 1 to 0 transitions are clean thermometer codes such as 11111111 to 00001111. However, due to the uneven propagation delays in FPGA such as the carry lookahead structure in DSPs, it may randomly occur a signal malposition called “bubble” such as 11111111 to 00001011. Therefore, addressing the problem requires a proper edge detection design. A rising edge detection can be shown in Fig. 19. The schematic demonstrates that the function of rising edge detection is realized by plenty of three input AND gates, which recognizes the transition “100” rather than “10” to eliminate bubble to some extent. In addition, the same structure can be used for detecting falling edge by modifying the input pattern that is opposite to with detecting the rising edge. In this case, “000010” mentioned above cannot be considered as a falling edge because the module recognizes the falling pattern “011” instead of “01” pair to reduce the data error. If there are larger bubbles causing the previous circuit invalid, such as 00001001, the input of AND gate can be increased to address the problem. Since the large bubble often occur in DSP delay line based on experiment, it is recommended to use four input AND gates to detect “0011” pattern.

**图示

描述已自动生成**

Fig.19 The structure of edge detection

In Fig. 20, we can detect both positive edge and negative edges.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Fig.20 The code of edge detection (encoder.v).

## Multi-edge encoder module

Multi-edge encoder is an extremely significant module, which transfers the data from the multi-edge detection module into binary code. Generally, the delay line based on CARRY4 is encoded by the priority encoder, which only converts the edge located in the highest position of the delay line into binary code. In this project, it is necessary to sum all edge positions in the delay line, so the priority encoder is not suitable in this case. In contrast to the priority encoder, the multi-edge encoder encodes the position of each edge in the delay line and adds them up to gain the total result [[[1]](#endnote-1)]. The schematic of the module in Fig. 21 consists of 16 encoder units. Each block has 48 input ports, connecting the output of multi-edge detection, binary code of the sum of the edge position, and block offset that is the product of the number of edges in the block and offset indexes. The encoder unit is designed with 48 input ports in order to align the number of output ports on the DSP so that the same number of encoder units can be added when the delay chain is extended by attaching more DSPs to the tail. The output of the 16 blocks is summed together through 31 adders arranged in 5 rows, and each row represents a pipeline, which means that the output data of the encoder module needs 5 clock cycles calculation to obtain the final accumulation result (OUT[15:0]). All of the components are driven by a 172Mhz clock. Besides, the multi-edge encoder can be extended by adding “encoder unit” and pipelines to maintain the original system frequency.

**图示

描述已自动生成**

Fig.21. the structure of the multi-edge encoder (The figure is out of data, in our latest code, the multi-edge encoder is composed of 16 encoder modules rather than 4 encoder modules)

# Configuration.v

We can change the configuration in this file. However, please use MULTIEDGE\_ENCODER when use wave union launcher + single DSP delay line/single CARRY4 delay line.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

1. [↑](#endnote-ref-1)