Universidade Federal do Rio Grande do Norte

Departamento de Engenharia de Computação e Automação
DCA0119 - SISTEMAS DIGITAIS

RELATÓRIO DO 3º PROJETO DE UNIDADE

Sistema embarcado em um microcontrolador para o controle de um secador de grãos

Integrantes:
Felipe Oliveira Lins E Silva - 20180154889
Emerson Wendlingger Dantas Sales - 20180154851
Luís Gabriel Pereira Condados - 2015093091
Luís Henrique Matias Viana - 20180155026
Professor orientador: Sérgio Natan

Natal-RN
2018

Universidade Federal do Rio Grande do Norte

Departamento de Engenharia de Computação e Automação

DCA0119 - Sistemas Digitais

RELATÓRIO DO 3º PROJETO DE UNIDADE

Relatório apresentado à disciplina de Sistemas Digitais, correspondente a 3º unidade do semestre 2018.2 do 7º período do curso de Engenharia de Computação e Automação da Universidade Federal do Rio Grande do Norte, sob orientação do **Prof. Sérgio Natan Silva**.

Natal-RN

2018

Conteúdo

1 Agradecimentos

Agradescemos ao André, que toma conta do laboratório de Hardware, por todo apoio dado. Sem ele o trabalho não seria possível.

2 INTRODUÇÃO

Este projeto consiste na implementação de um secador de grãos utilizando um microcontrolador **Atmega328p** com o kit de desenvolvimento **Altera DE2**, mais especificamente uma **FPGA**. O primeiro recebe dois sinais analógicos de entrada, respectivamente de temperatura e luminosidade, em sequência, envia via SPI para a FPGA, inicia o processo de secagem. LEDS serão acesos, conforme as especificações, durante o processo. O microcontrolador emite um sinal PWM ($X(t)$), que vai para um opto-acoplador, que associa a saída PWM ao circuito do motor.

O sinal $X(t)$ obedece a esta formula:

$$X(t) = \alpha \cdot Z(t) \cdot T(t) + \beta \cdot L(t)$$

Onde, α e β são constantes; $Z(t)$ corresponde ao formato da curva a qual queremos que modele o comportamento do motor; $T(t)$ e $L(t)$ são os valores obtidos, respectivamente, pelos sensores de temperatura e luminosidade.

3 ESPECIFICAÇÕES DO PROBLEMA

1. O sistema deverá ter dois sinais analógicos de entrada: um que tem como origem o sensor de Temperatura ($T(t)$), e outro que vem do sensor de luminosidade ($L(t)$).

2. O sistema deve possuir uma chave CH1 (HiZ e terra), a qual, quando em terra, dá início ao processo de secagem.

3. O sistema deve ter um LED (LED1) que tenha sua luminosidade proporcional a $X(t)$ - o qual é o sinal de saída, que controlará o motor.

4. O sistema deve ter um LED (LED2) o qual a luminosidade será proporcional a $T(t)$ ou $L(t)$ - situação do secador.

5. O sinal emitido pelo controlador deverá está em uma frequência entre 100 Hz e 100 kHz.

4 Implementação

Para a confecção do projeto, dividimos-o em módulos (blocos) – com a finalidade de fazer com que o desenvolvimento colaborativo seja o mais proveitoso possível. Bem como, em decorrência da modularização, facilitar o debug – visto que a modularização permite que saibamos facilmente quais partes não estão funcionando adequadamente. Tal projeto, é bem explicitado pelos esquemáticos abaixo e tivemos a seguinte implementação:

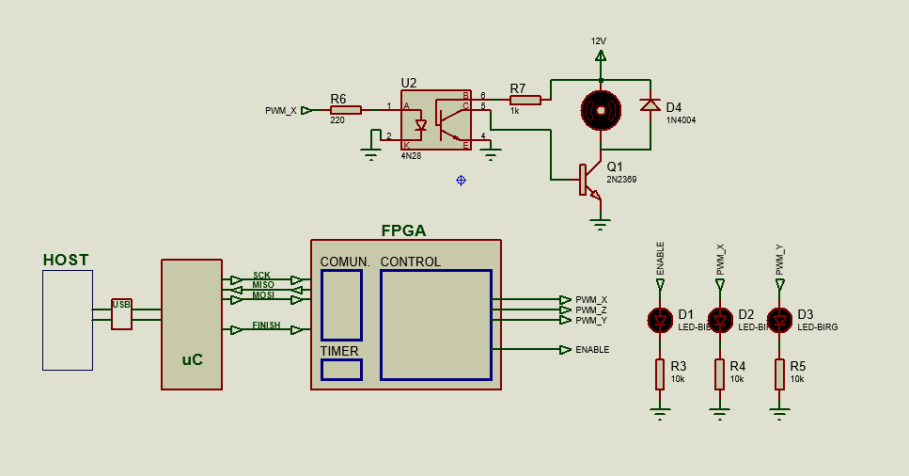

Figura 1: Esquemático do projeto completo

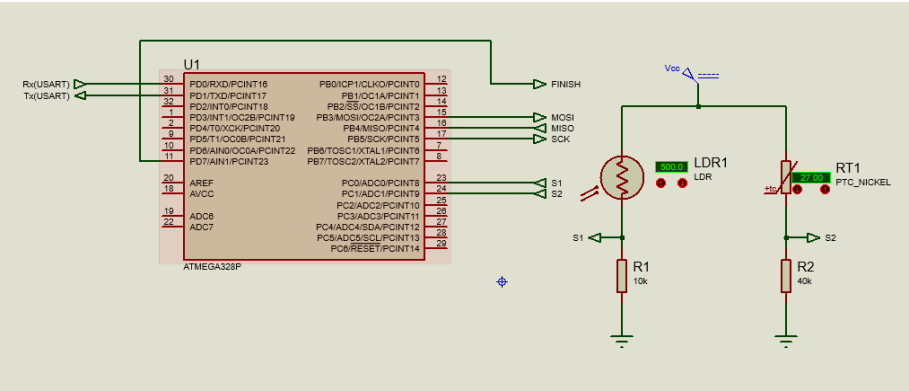

Figura 2: Esquemático do μC

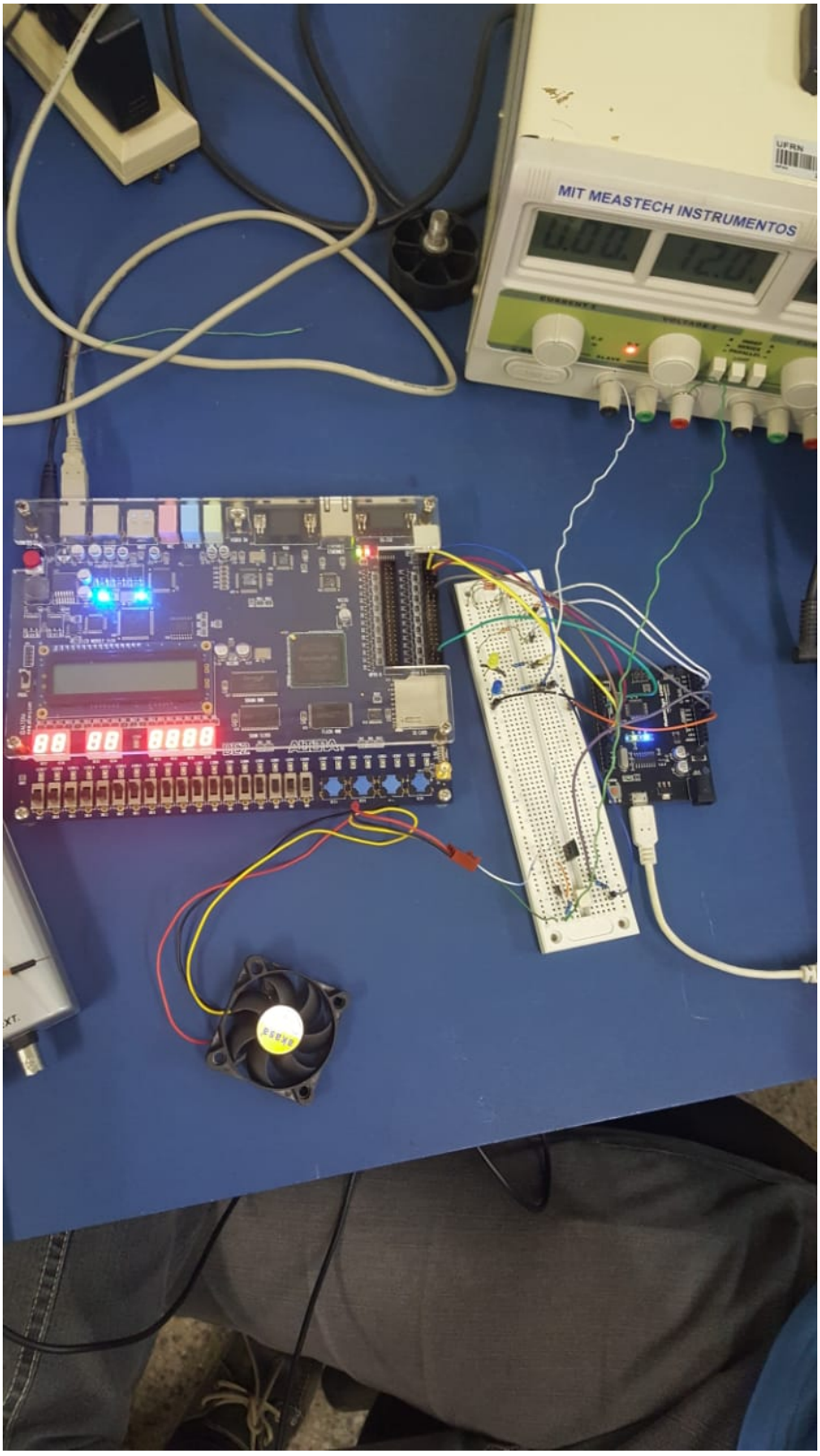

Figura 3: Implementação do circuito utilizando o $\mu CeaFPGA$

5 μControlador

O microcontrolador foi utilizado como interface para coletar os dados dos sensores de temperatura e luminosidade, e transmitir para a FPGA por via SPI. Abaixo, segue a função main com a implementação geral do código do Atmega328p. Para melhores detalhes sobre o código, o link do github estará no fim do documento.

```
int main()
{
  uint8_t temp;
  uint8_t lumin;
  uint8_t pwm;

  setup();
  while(true)
  {
     temp = readTherm();
     lumin= readLumin();

     pwm = SPI_MasterTransmit(temp); // manda temp e recebe o
                                     // ultimo PWM calculado
     SPI_MasterTransmit(lumin);      // envia informacoes o
                                     // ADC relativo a leitura
                                     // ADC do sensor
                                     // de luminosidade

     pwm = (pwm >> 1) | ((pwm & 0x01) << 7);

     USART_transmision(pwm);         // repassa a informacao do
                                     // pwm via serial USB da placa
                                     // arduino
  }

  return 0;
}
```

6 Blocos

6.1 Bloco SPI

Este bloco implementa o protocolo SPI entre o Atmega328p e a FPGA. Também, ele garante que os sinais de temperatura e luminosidade recebidos pelo μC, passarão de forma síncrona para o bloco de controle. Ele recebe como entrada: o clock do SPI (CSK); DATA_MOSI do SPI; Finish, que é uma flag utilizada para a transmissão e sinalizar que um sinal foi totalmente enviado; Z_PWM, que é um vetor utilizado para transmissão SPI para o Atmega328p; PWM_EN, que é o enable do PWM – que garante que a saída do PWM será 0 para $T > 30\ s$.

Figura 4: Bloco de comunicação SPI

6.2 Bloco Timer

Este bloco implementa o timer, com a precisão dada em ms. No bloco, é necessário uma entrada de um clock de 50 MHz e de uma entrada – do tipo switch – HALT. Esta desliga e liga o circuito.

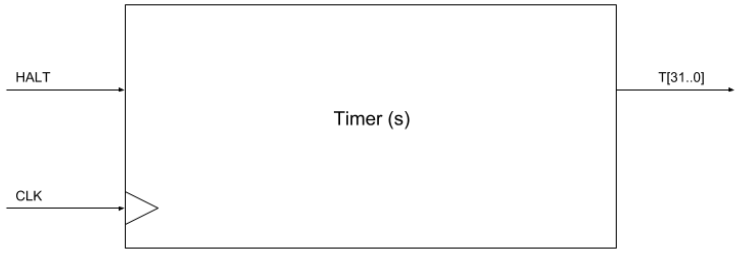

Figura 5: Bloco Timer

6.3 Bloco de Controle

O bloco de controle realiza o processamento dos dados recebidos pelo Atmega328p. Ele recebe o clock de $50MHz$ como entrada, os dois sinais oriundos do μC – e retorna os respectivos PWM e o enable do PWM.

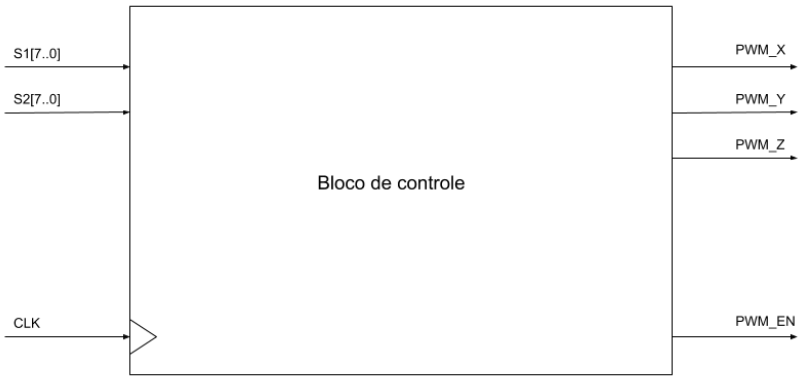

Figura 6: Bloco de Controle

6.4 Gerador do sinal Z(t)

Este bloco consiste na implementação do sinal $Z(t)$, o qual é uma função definida por intervalos. Para fazer este bloco, foi utilizado uma lógica utilizando comparadores, para encontrar em qual intervalo o tempo T se encontra, e um decodificador para a partir do intervalo encontrado – este resgata os valores que definirão a reta a partir de duas memórias ROM.

Tendo em vista que a função $Z(t)$ sempre tem forma $a \cdot t + b$ – as memórias ROM armazenam os valores das constantes a e b.

Figura 7: Bloco do gerador do sinal Z(t)

6.5 Bloco PWM

O bloco PWM serve para gerar todos os valores PWM (do $X(t)$, $U(t)$ e $Z(t)$). Ele recebe como entrada o clock de $50MHz$, os valores em ponto flutuante dos respectivos sinais e retorna os PWM associado a esses.

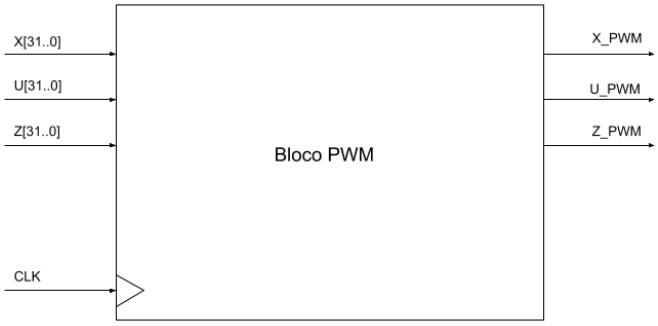

Figura 8: Bloco PWM

7 Resultados

Após implementar o sistema, e considerando as normalizações necessárias feitas para garantir o formato da curva de controle, obtivemos o seguinte gráfico – o qual foi obtido através do uso do protocolo USART entre o computador e o Atmega328p para transmissão dos dados entre a rede FPGA-ATMEGA-PC:

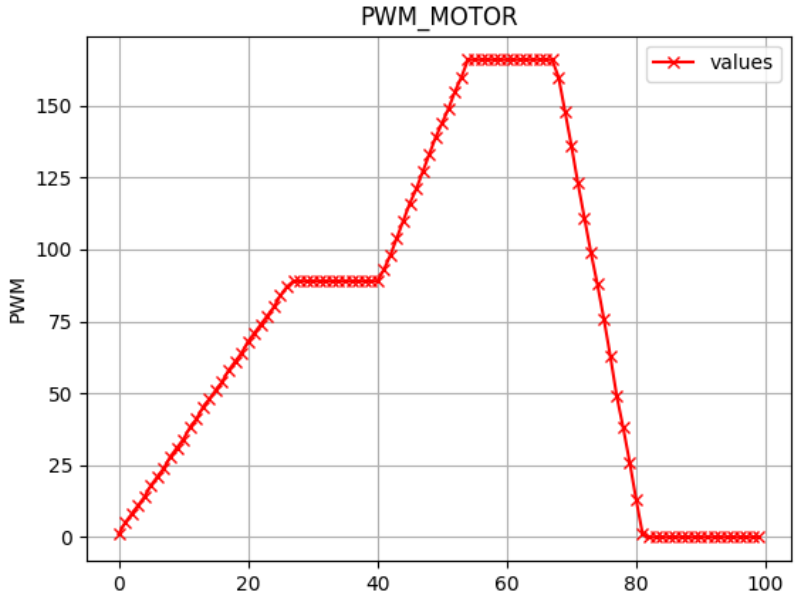

Figura 9: Gráfico do sinal de controle resultante

O qual é a cruva desejada para dados requisítos do projeto.

8 Vídeo demonstrativo e link para o repositório

Segue aqui o link para o vídeo de apresentação dos resultados:

https://www.youtube.com/watch?v=jDaGKF5uwjA

Link para o repositório onde o código está salvo:
https://github.com/Gabriellgpc/SDPU3_atmega_interface
https://github.com/51rL1N5/Seed-dryier-using-FPGA-and-Arduino

9 Anexo

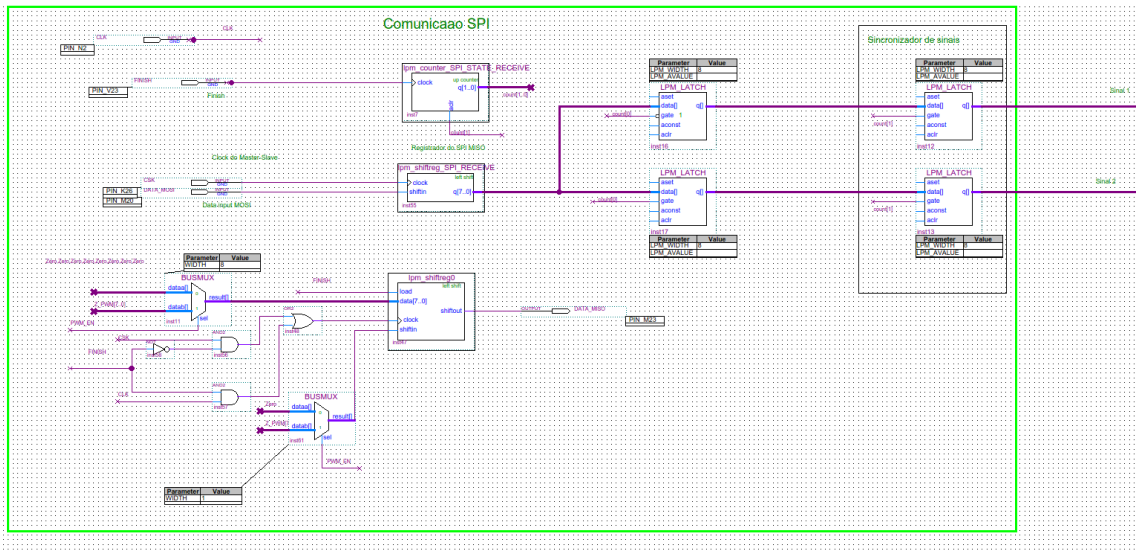

Figura 10: Bloco SPI

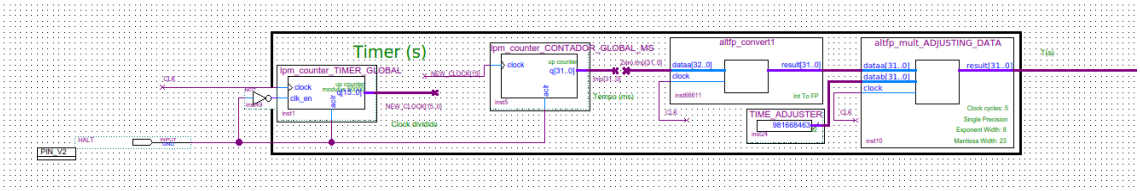

Figura 11: Bloco Timer

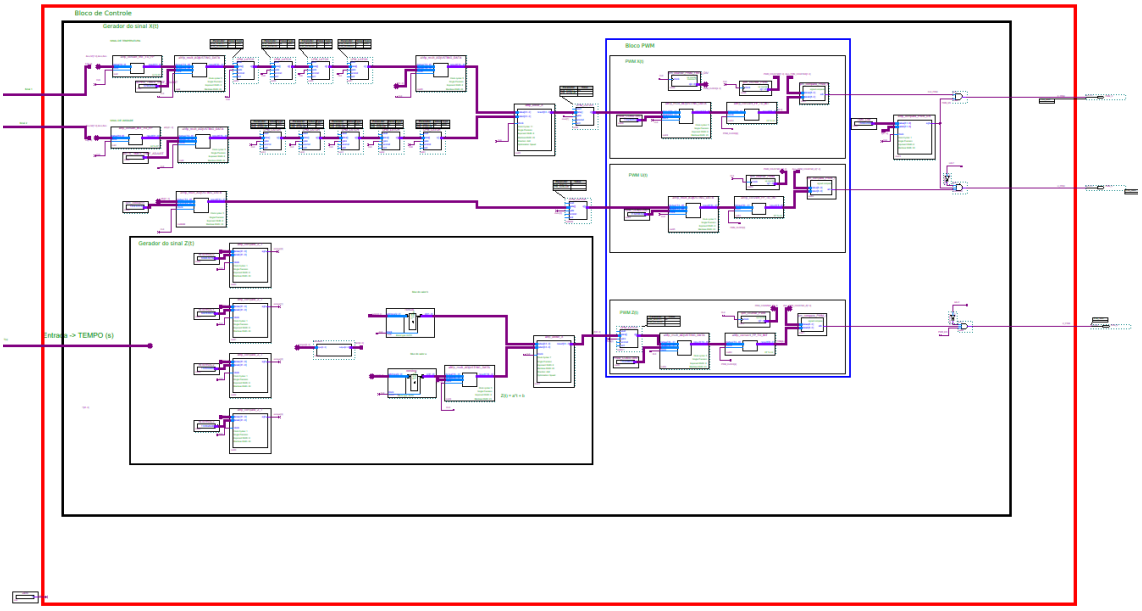

Figura 12: Bloco de Controle

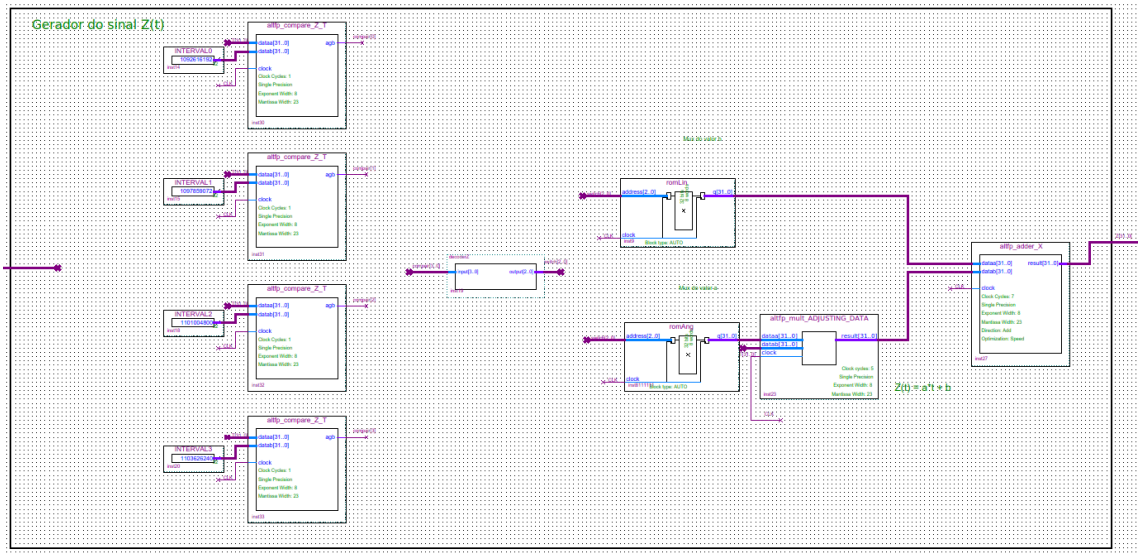

Figura 13: Bloco de geração de $Z(t)$

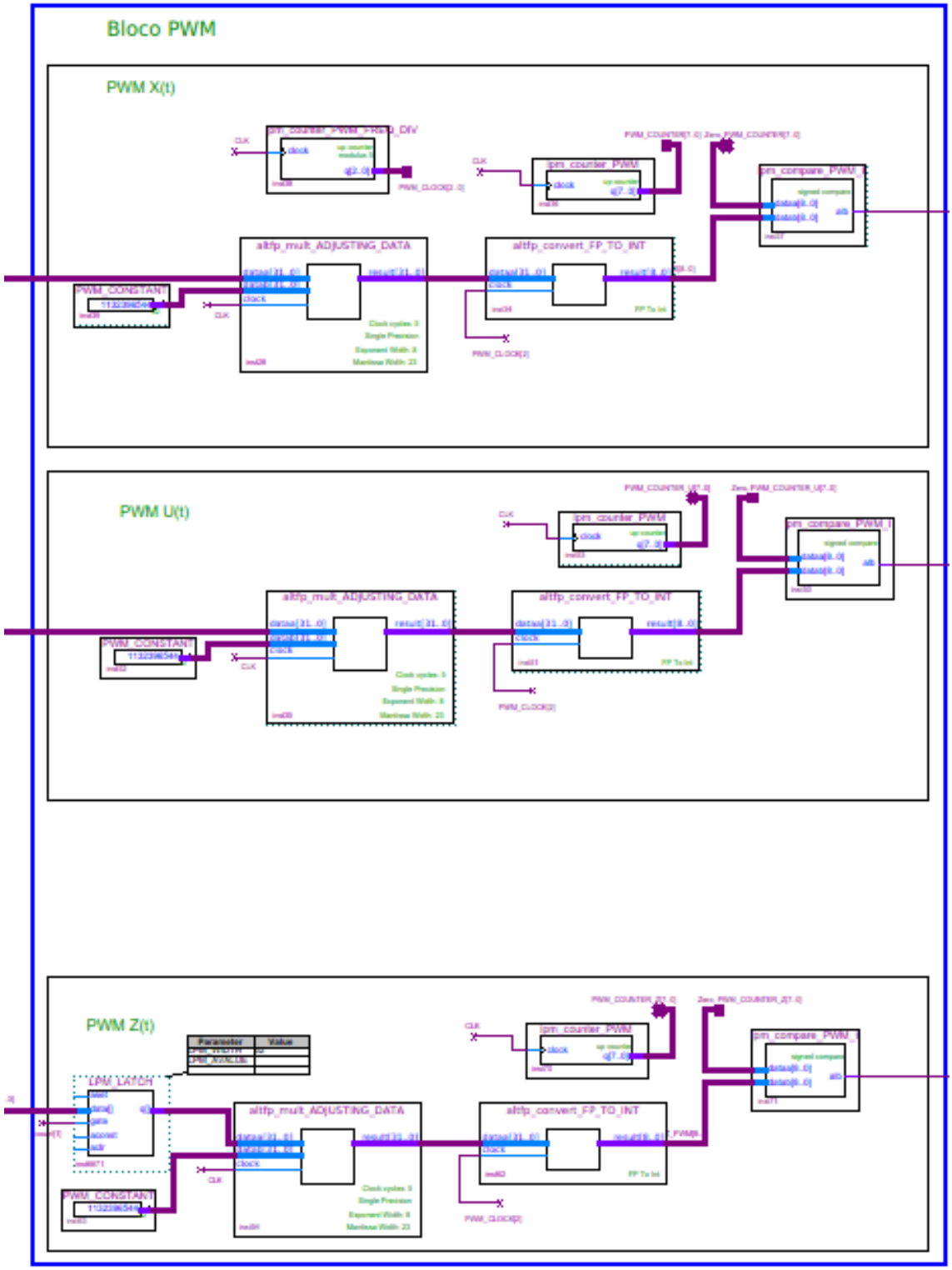

Figura 14: Bloco PWM

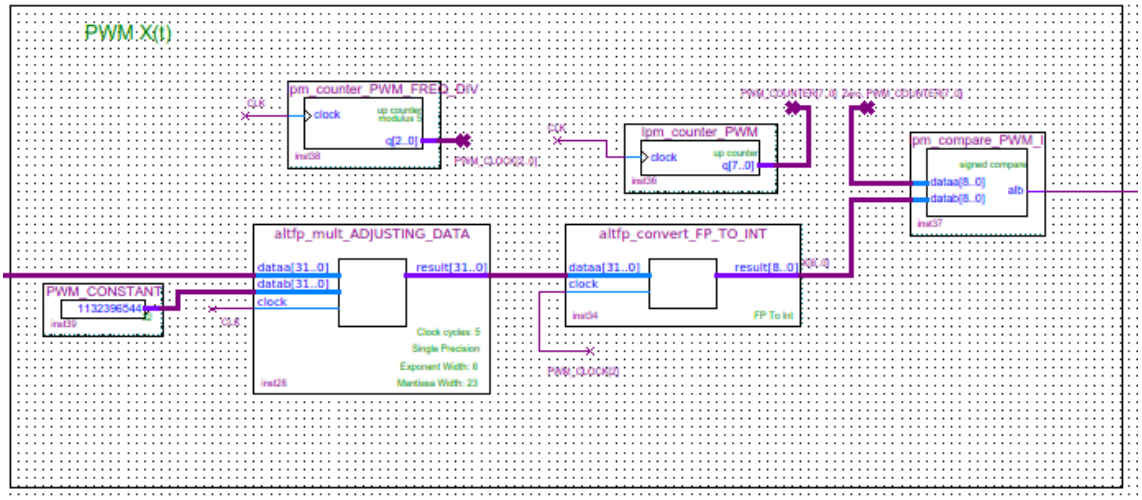

Figura 15: Bloco PWM – implementação detalhada do subbloco

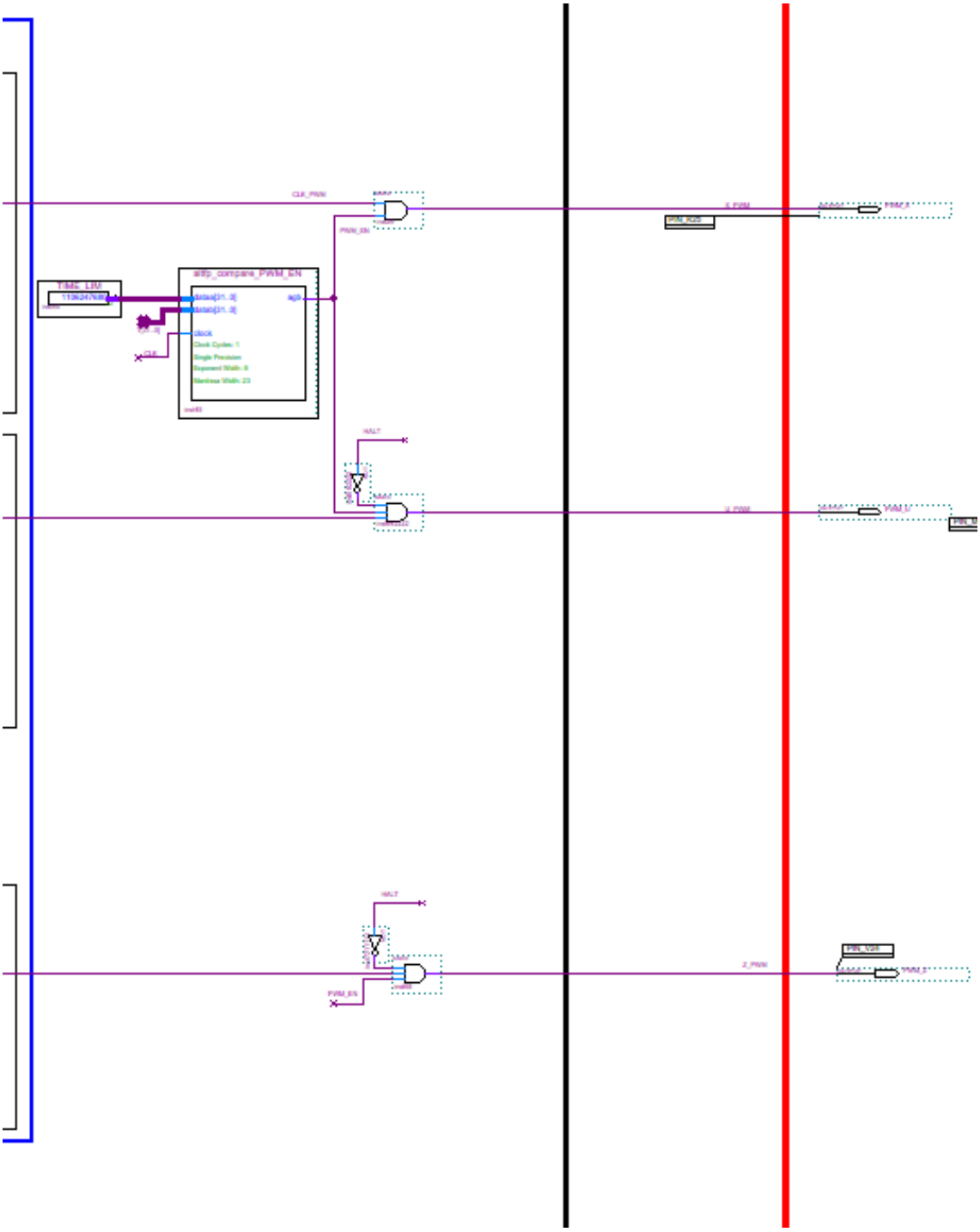

Figura 16: Bloco PWM enable