UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

RELATÓRIO DA 2º UNIDADE
IMPLEMENTAÇÃO DO ALGORITMO DO MÉTODO K NEAREST NEIGHBORS (KNN) PARA VERIFICAÇÃO DE ESTADOS EMOCIONAIS UTILIZANDO COMPUTAÇÃO RECONFIGURÁVEL

MARIA ALICE DE MELO SOUSA: Nº 20200149247

RIJKAARD MELO: Nº 2016018734

SAMUEL CAVALCANTI: Nº 20200149318

TIAGO FELIPE DE SOUZA: Nº 20190153105

Natal-RN
2022

MARIA ALICE DE MELO SOUSA: Nº 20200149247

RIJKAARD MELO: Nº 2016018734

SAMUEL CAVALCANTI: Nº 20200149318

TIAGO FELIPE DE SOUZA: Nº 20190153105

IMPLEMENTAÇÃO DO ALGORITMO DO MÉTODO K NEAREST NEIGHBORS (KNN) PARA VERIFICAÇÃO DE ESTADOS EMOCIONAIS UTILIZANDO COMPUTAÇÃO RECONFIGURÁVEL

Segundo relatório apresentado à disciplina de Sistemas Digitais, correspondendo a avaliação da 2º unidade do semestre 2021.2 do curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, sob orientação do **Prof. Dr. Marcelo Augusto Costa Fernandes.**

Professor: Dr. Marcelo Augusto Costa Fernandes.

Natal-RN
2022

Lista de Figuras

Sumário

1 CONSTRUÇÃO DO ALGORITMO

A construção do algoritmo foi baseada no método do K NEAREST NEIGHBORS (KNN), possui 4 entradas, medidas por sensores, são eles: heart_beat, blood_volume, galvanic_skin_response e respiration_sensor. Obtemos como saídas 4 leds que representam 4 estados emocionais, são eles: scary, boring, amusing e relaxing, em que, a intensidade de cada saída depende da saída do KNN, onde quando mais próximo uma amostra está de um sentimento maior é a intensidade do led.

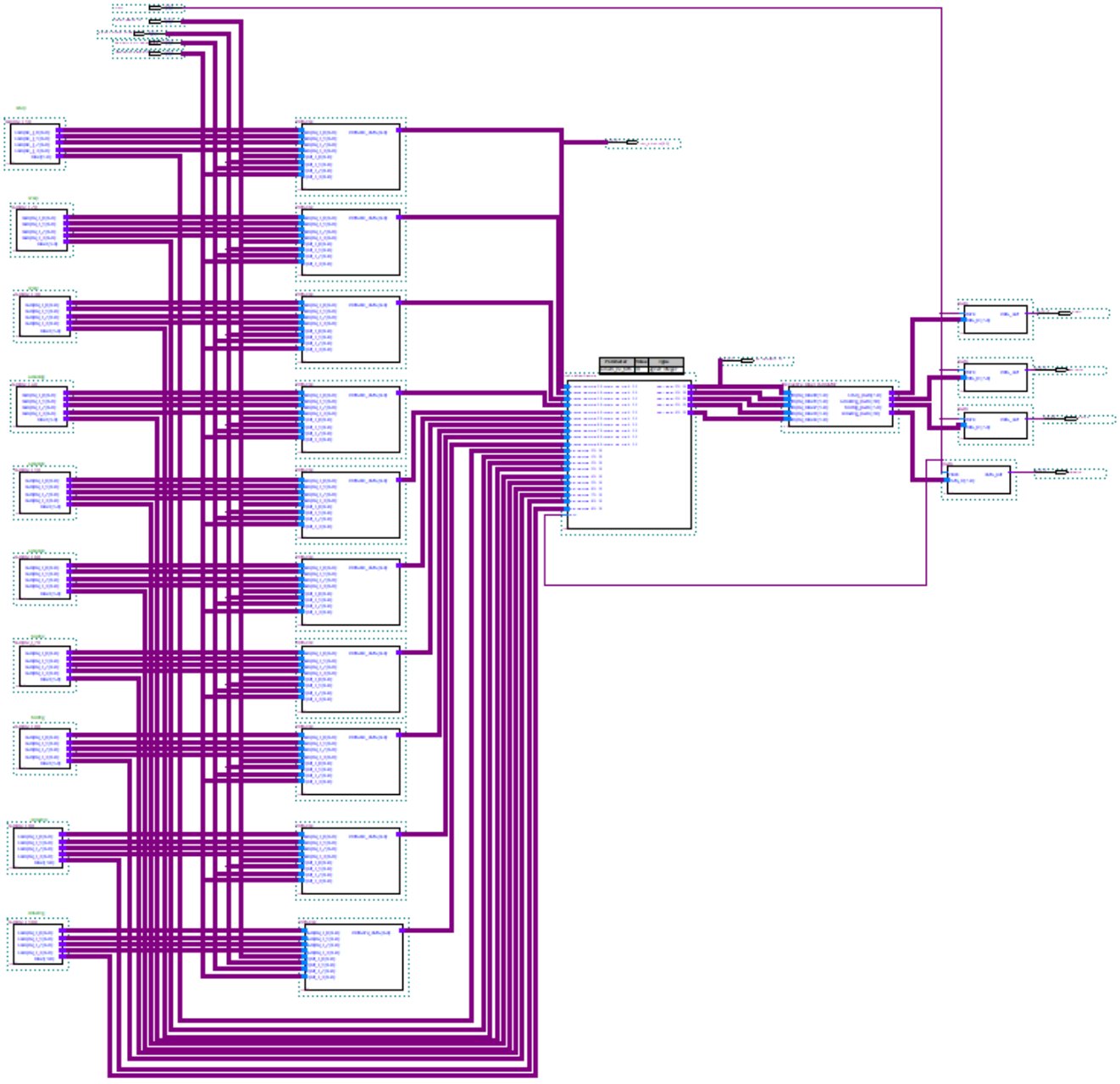

Figura 1: Circuito do Algoritmo KNN

A entrada do circuito são os quatro sensores cujo os valores variam de zero até mil e vinte três, por tanto uma entrada de dez bits. O conjunto de dados foi representando como 10 blocos chamado de *sample_i_XX* , onde XX pode ser 10,20,30,...,90,100. Um bloco *sample_i_XX* tem 5 saídas sendo uma delas um rótulo de sentimento e as colunas do seu vetor, se XX for 10, então o valor de cada coluna é 10, esses valores foram escolhidos para facilitar reconhecimento dos testes

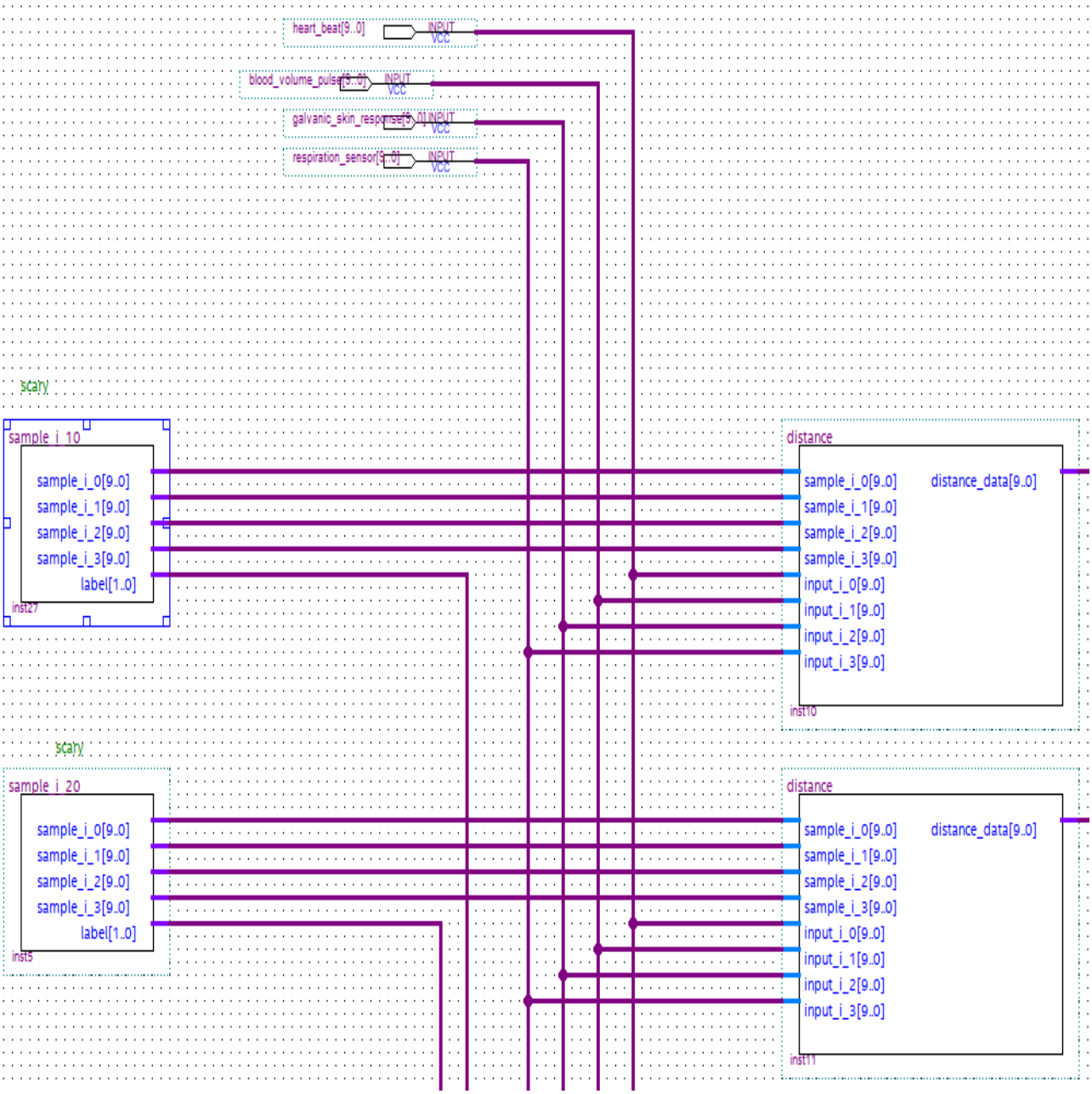

Figura 2: Entradas e Samples direcionadas para os módulos de distância

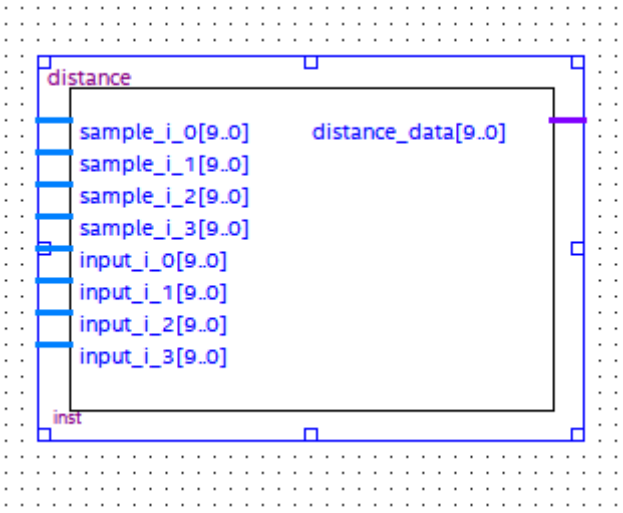

Figura 3: Módulo da distância

Os inputs e cada *sample_i_XX* está conectado a um módulo de distância que calcula a norma L1 também conhecida como distância de Manhattan, que nada mais é que a soma do modulo da diferença entre a entrada e um *sample*

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

Figura 4: Norma **L1**

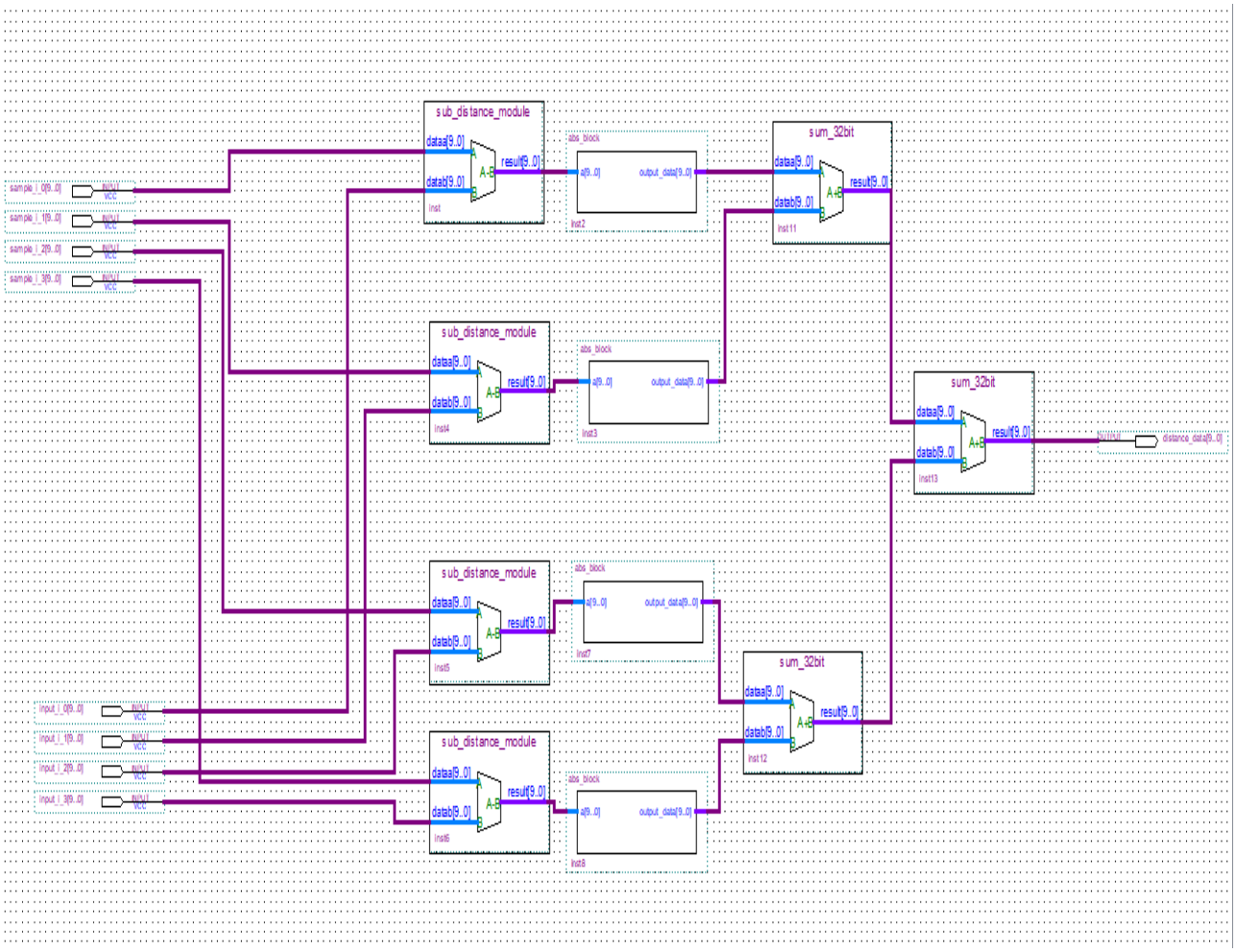

Figura 5: Módulo da distância - componentes internos

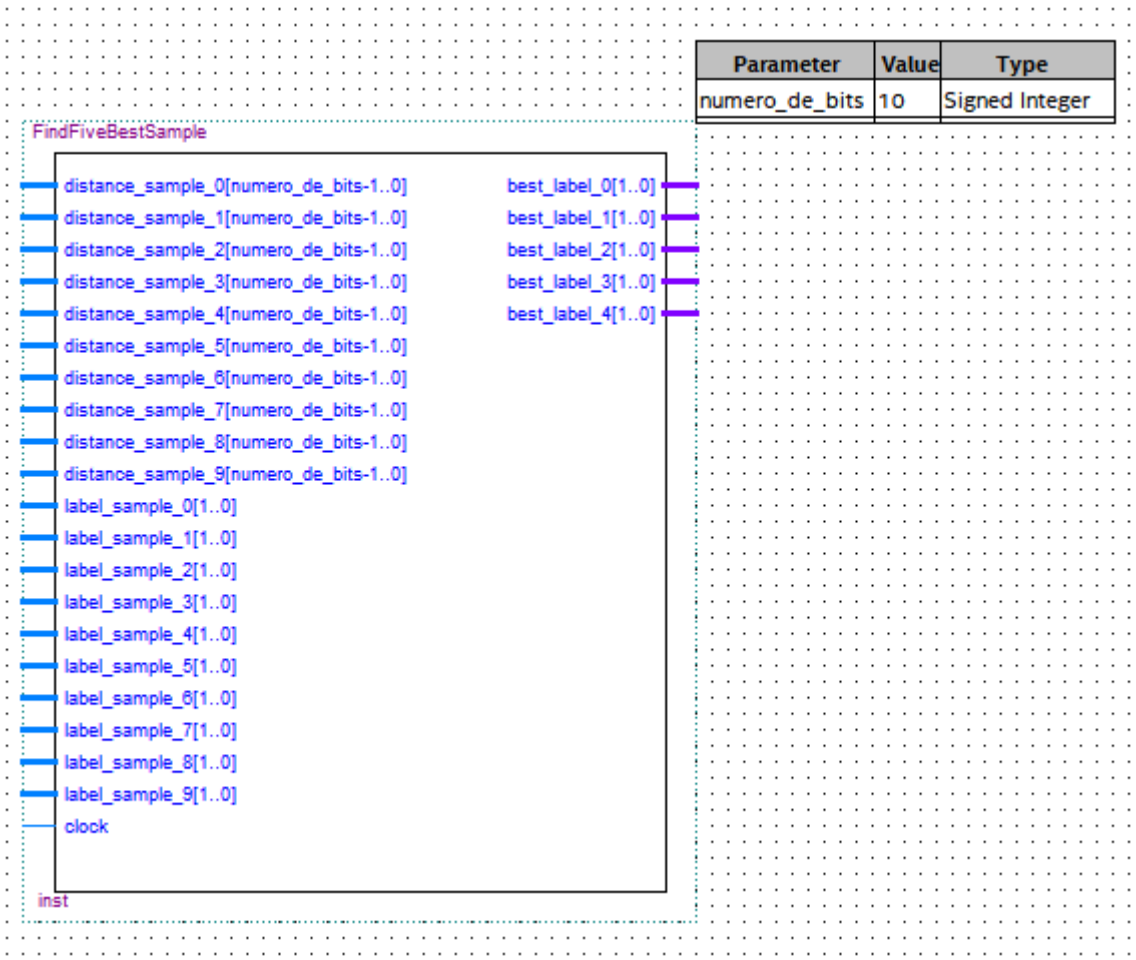

Figura 6: Módulo que encontra as 5 melhores amostras

Perceba que até o momento o fator limitante da implementação é o caminho crítico, pois até o momento não foi utilizado o clock, mas para encontrar as cinco menores distâncias foi utilizado o clock. A cada ciclo de clock é feito uma sequencia de comparações para encontrar a menor distância. Para que o resultado da comparação não retorna-se a mesma menor distância cinco vezes, foi adicionado um registrador que armazena um vetor de nove bits onde cada bit diz se um *sample* pode ou não ser utilizada na sequência de comparações, com isso após a sequência de comparações retornar o índice da menor distância esse índice é usado para desabilitar o melhor *sample* assim no próximo ciclo de clock a sequência de comparações deve retornar o segundo melhor *sample* e esse fluxo se repete até cinco vezes, onde na quinta vez, o componente atribui os 5 melhores nos fios de saída e o registrador de índices é resetado. Esse componente foi implementado em VHDL e também foi observado que reimplementar o bloco em VHDL reduziu o número de blocos lógicos e registardes.

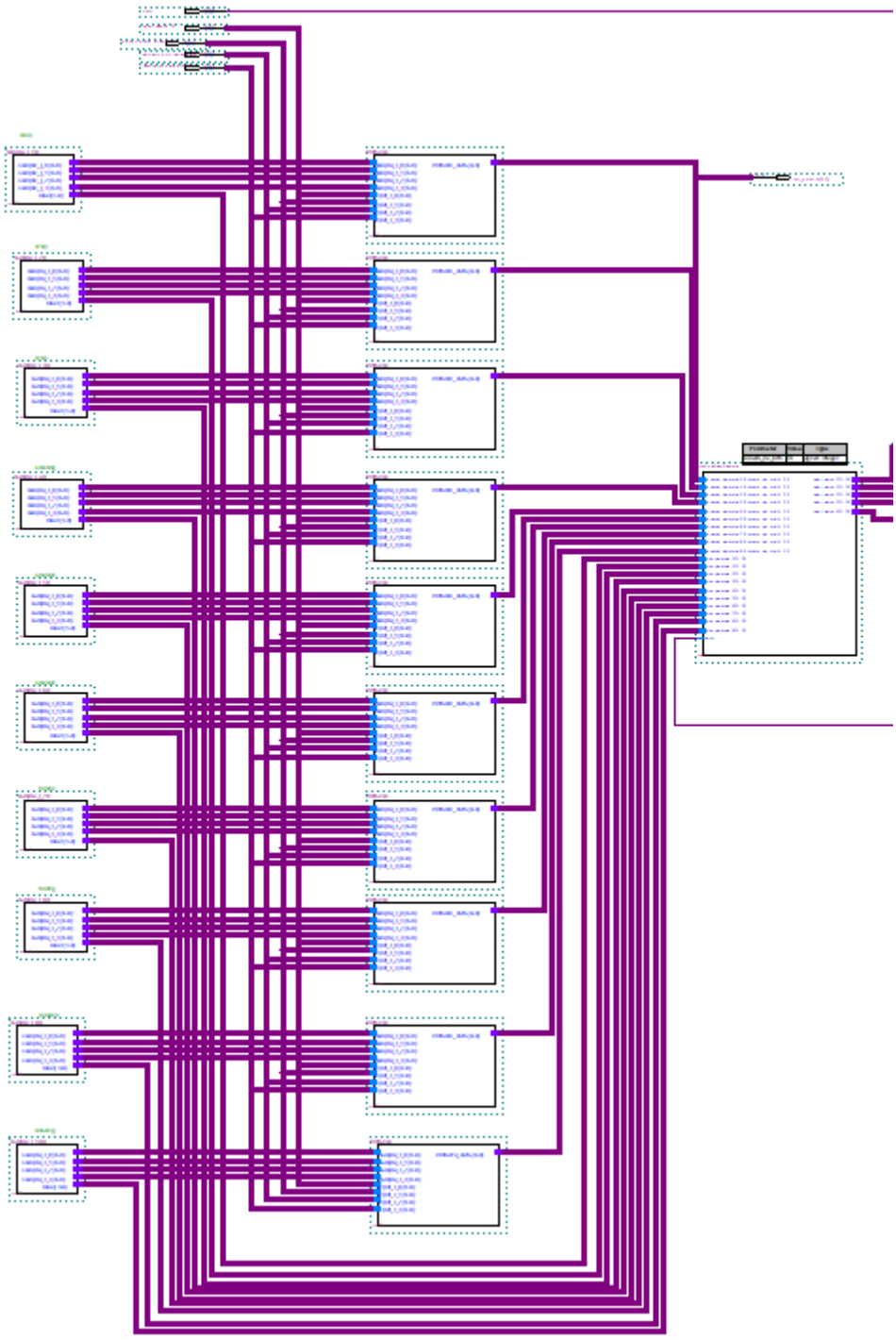

Figura 7: Circuito com entradas e amostras até o módulo que busca as 5 melhores amostras

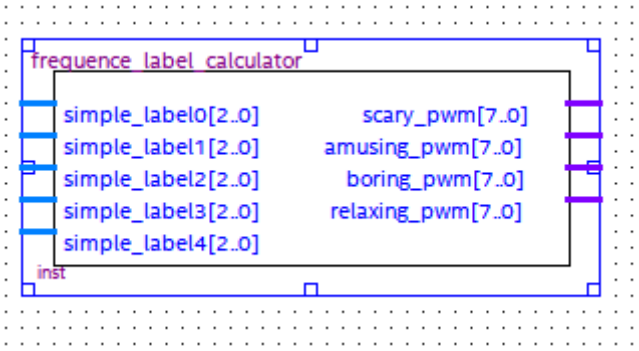

Figura 8: Módulo do cálculo da frequência de *labels*

Após ter encontrado as 5 melhores amostras, na verdade os 5 melhores *labels* é necessário transformar essa informação em *Duty cycle* de cada led. Para isso foi utilizado a seguinte equação

$$\textbf{quantidade de vezes que um rótulo se repete} \times \frac{255}{\textbf{número total de rótulos}} \tag{1}$$

Sabendo que o número total de rótulos é 5, então

$$\textbf{quantidade de vezes que um rótulo se repete} \times 51 \tag{2}$$

Por exemplo, supondo que o modulo receba como entrada o seguinte vetor: [scary, scary, scary, relaxing, boring]. Então, como o scary se repetiu 3 vezes, relaxing e boring uma vez, então o *Duty cycle* do scary é 60%, uma vez que $51 * 3 = 0.6 * 255$, onde 255 é maior valor do contador do pwm.

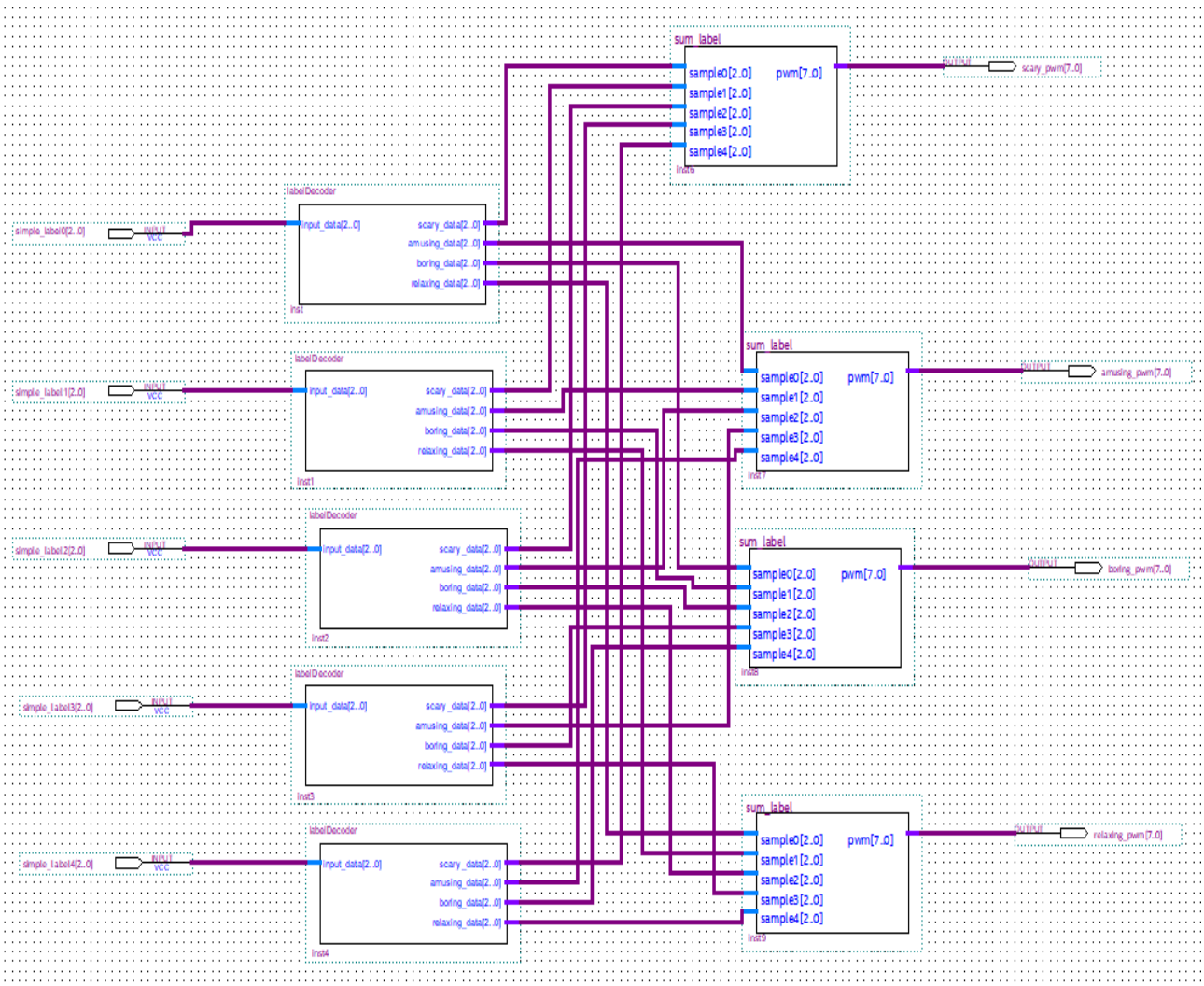

Figura 9: Módulo do cálculo da frequência de *labels* - componentes internos

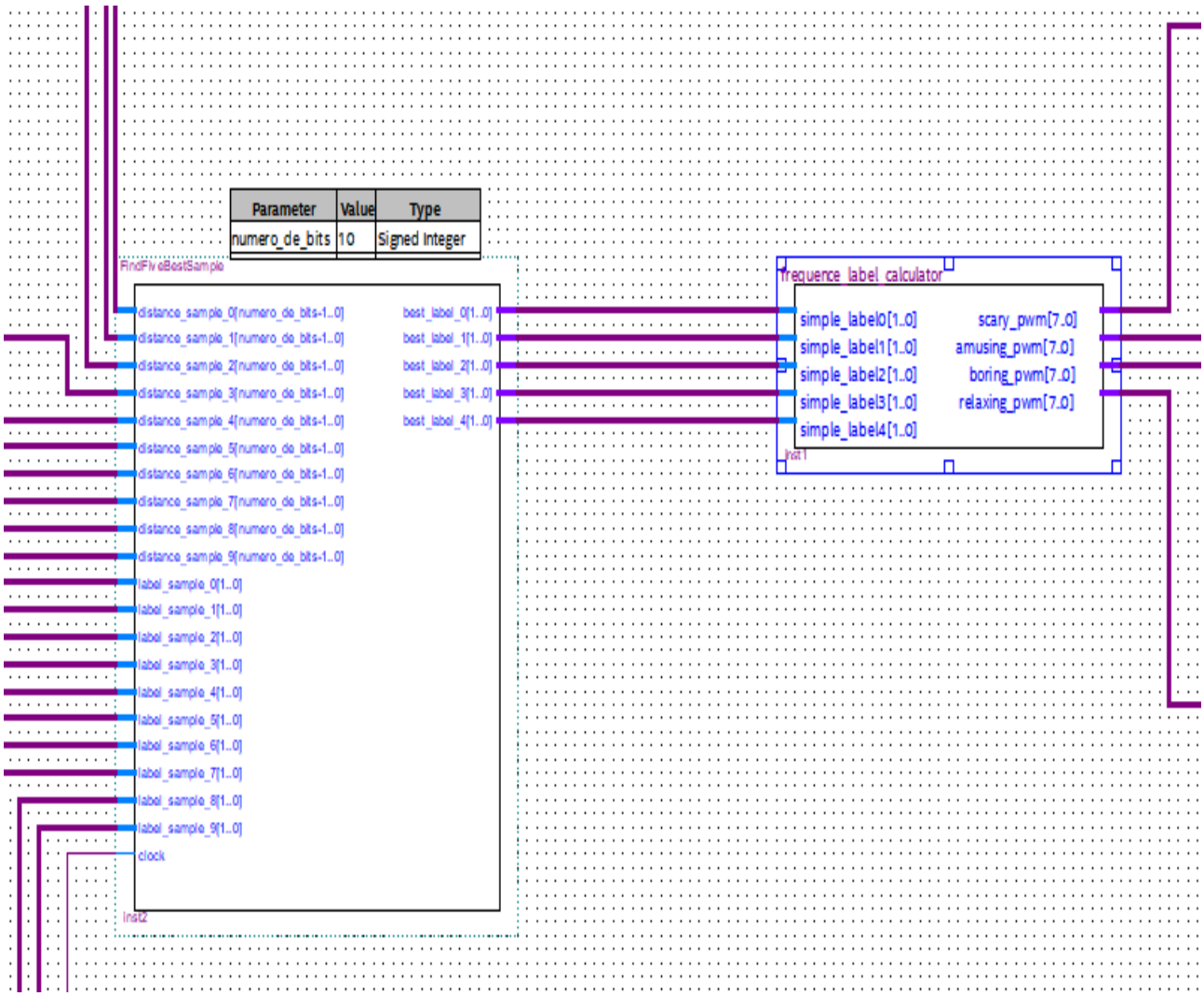

Figura 10: Melhores amostras saindo para o calculador de frequências

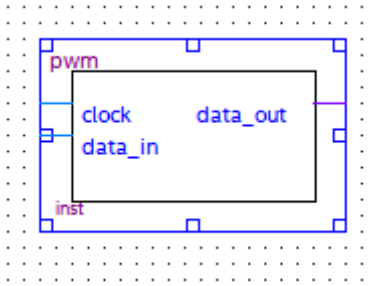

Figura 11: Módulo PWM

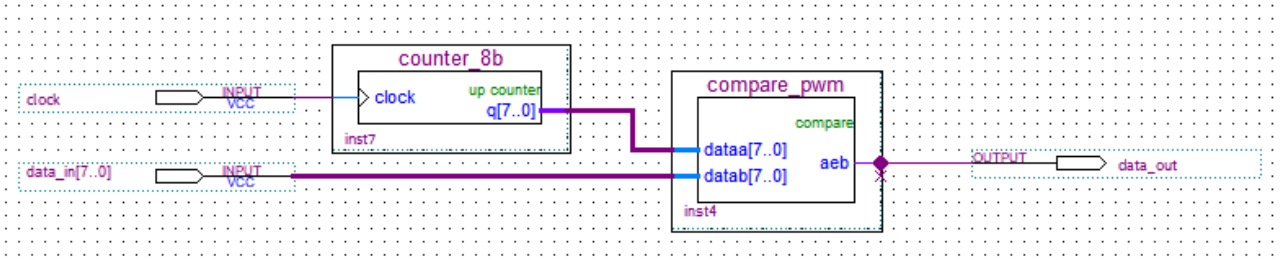

Figura 12: Módulo PWM - componentes internos

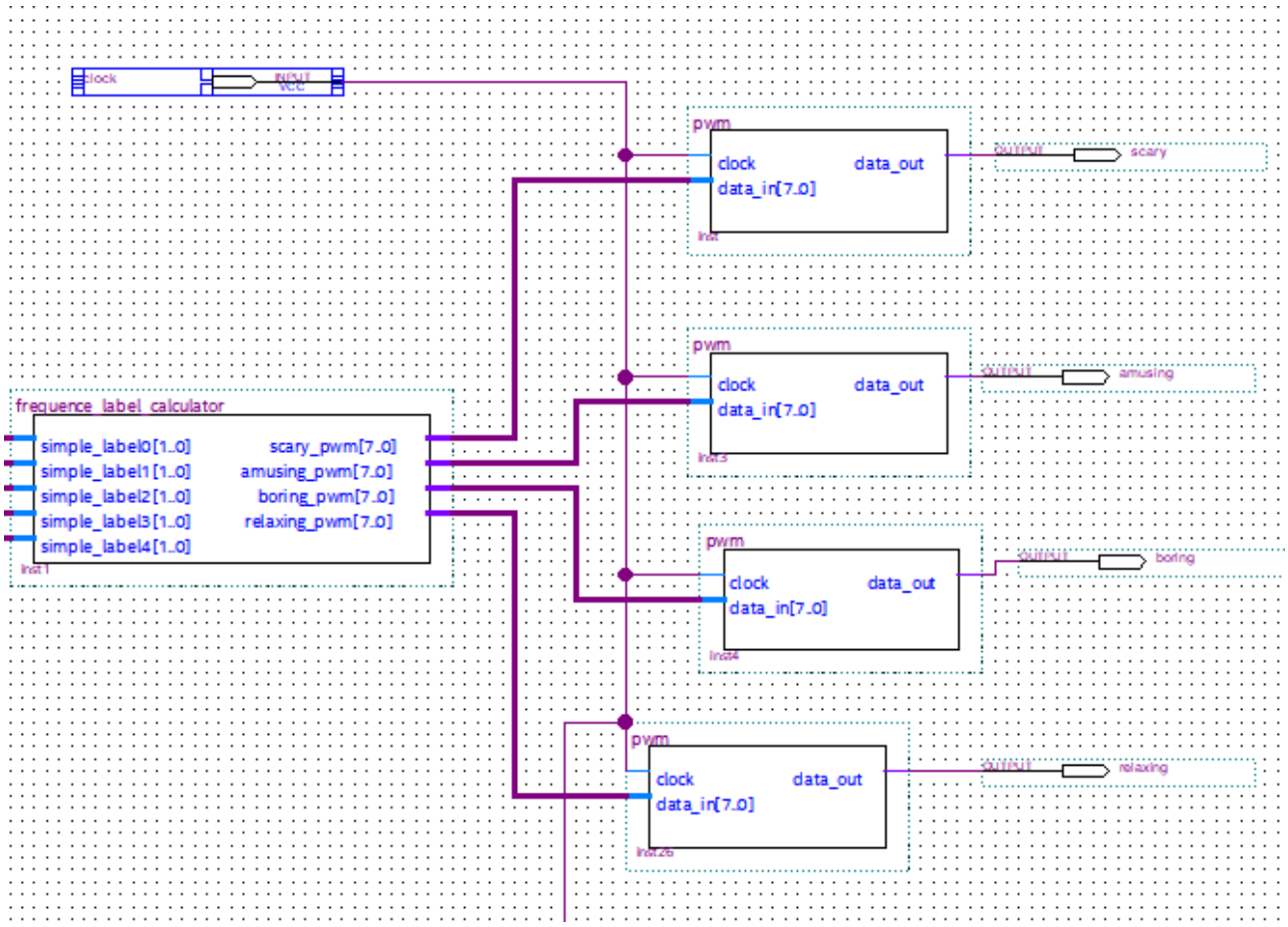

Figura 13: Saídas PWM

Por fim cada valor calculado no Módulo do cálculo da frequência de *labels* é conectado a uma saída pwm de cada sentimento.

2 COMPILAÇÃO

2.1 Detalhes da compilação

Podemos observar na imagem abaixo, que se refere ao sumário da compilação do projeto final, que utilizamos o software **Quartus Prime** em sua versão 17.0.0, Lite Edition, e utilizamos a família de placas reconfiguráveis **Cyclone V**, pois podemos obter os níveis de potência, custo e desempenho necessários para aplicação, visto que a potência total pode ser até 40% menor que as da família **Cyclone IV**, e também, o desempenho de mais de 4.000 MIPS, para menos de 1,8W de potência. Utilizamos também, 73 registradores, entre os fornecidos pelo próprio software e o que foram feitos em VHDL. Por fim, foram utilizados 65 pinos de um total de 268, o que corresponde a 24% de uso total.

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Feb 17 21:26:00 2022
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	knn
Top-level Entity Name	knn
Family	Cyclone V
Device	5CGXFC7C6F23I7
Timing Models	Final
Logic utilization (in ALMs)	804 / 56,480 (1 %)
Total registers	73
Total pins	65 / 268 (24 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

Figura 14: Flow status de compilação

2.2 Recursos

Os recursos utilizados no projeto foram:

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	835
2		
3	Combinational ALUT usage for logic	1447
1	-- 7 input functions	1
2	-- 6 input functions	220
3	-- 5 input functions	55
4	-- 4 input functions	26
5	-- <=3 input functions	1145
4		
5	Dedicated logic registers	67
6		
7	I/O pins	65
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	clock~input
12	Maximum fan-out	67
13	Total fan-out	4797
14	Average fan-out	2.92

Figura 15: Recursos

2.3 Frequências Máximas

As frequências máximas encontradas para os valores de temperatura de -40ºC e +100ºC, foram:

Fmax	Restricted Fmax	Clock Name	Note
391.08 MHz	391.08 MHz	clock	

Figura 16: Frequência Máxima para temperatura de -40ºC

Fmax	Restricted Fmax	Clock Name	Note
437.83 MHz	437.83 MHz	clock	

Figura 17: Frequência Máxima para temperatura de +100ºC

O tempo crítico é calculado como sendo o inverso da frequência máxima. Para efeitos de referência do cálculo do tempo crítico da aplicação, utilizaremos a menor frequência máxima encontrada, portanto, 391.08 MHz. Logo, temos que o tempo crítico será

$$\frac{1}{Fmax} = \frac{1}{391.08MHz} = 2.56\,ns.$$

3 SIMULAÇÃO E RESULTADOS

Com o objetivo de verificar se a implementação foi realizada com sucesso, foram feitos testes para cada componente da aplicação e utilizado o conjunto de dados da tabela 1

name row	heart beat	blood volume pulse	galvanic skin response	respiration	label
sample_i_10	10	10	10	10	0
sample_i_20	20	20	20	20	0
sample_i_30	30	30	30	30	0
sample_i_40	40	40	40	40	1
sample_i_50	50	50	50	50	1
sample_i_60	60	60	60	60	1
sample_i_70	70	70	70	70	2
sample_i_80	80	80	80	80	2
sample_i_90	90	90	90	90	3
sample_i_100	100	100	100	100	3

Tabela 1: dataset

3.1 Simulação do módulo de distância Manhattan

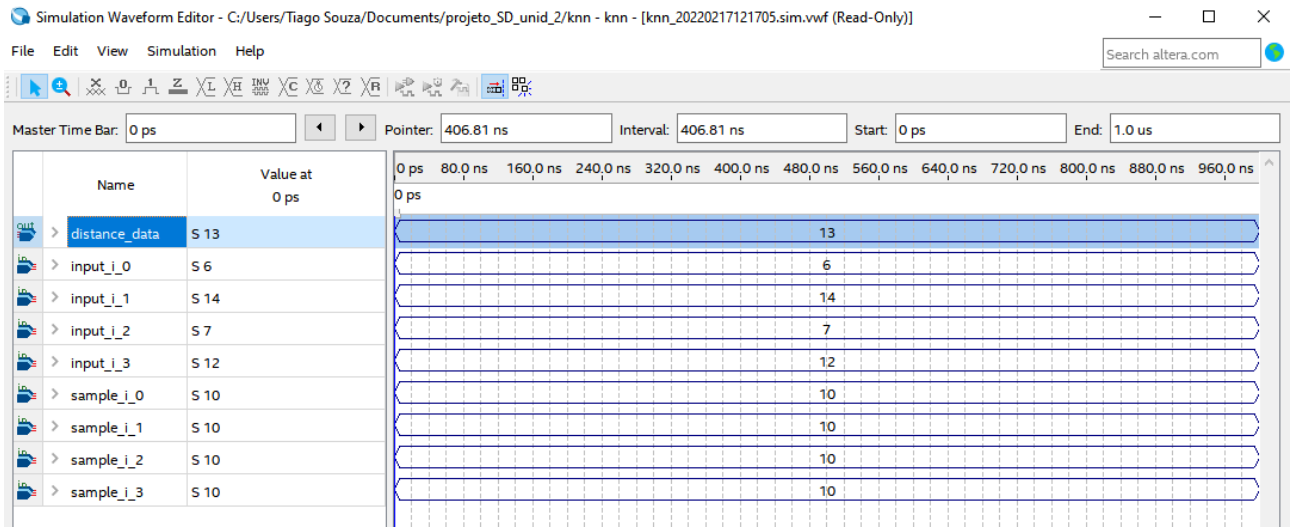

Figura 18: Simulação do módulo de distância Manhattan

Sabendo que a distância Manhattan é a soma dos modulo das diferenças então:

$$|6-10|+|14-10|+|7-10|+|12-10|=13 \tag{3}$$

3.2 Simulação do módulo que calcula as 5 melhores amostras

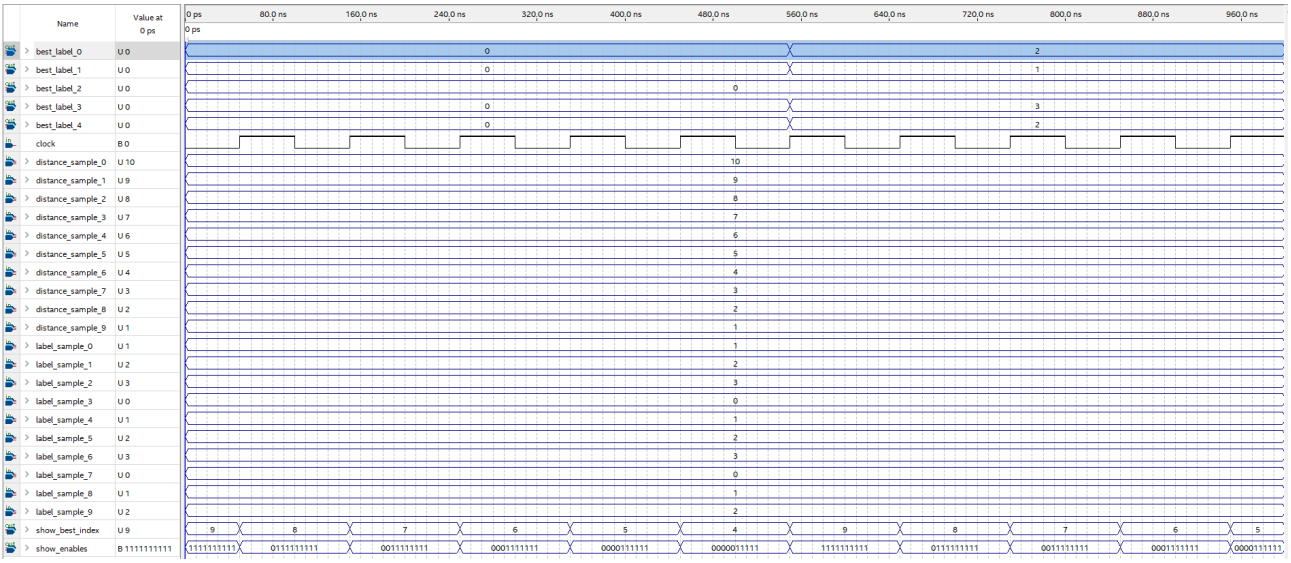

Figura 19: Simulação do módulo que calcula as 5 melhores amostras

nesse teste observamos que em toda na sexta subida de clock, os valores são atualizados, os melhores rótulos são que possuem as menos distancias e o vetor de habilitados é alterado a cada clock.

3.3 Simulação KNN - scary

Lembrando que na nossa implementação, foi utilizado conjunto de dados da tabela 1

portanto para uma entrada com os vetores de valores [10,10,10,10], a sequência das melhores amostras seria sample_i_10 ,sample_i_20,sample_i_30,sample_i_40, sample_i_50, cujo os seus respectivos labels seriam [0,0,0,1,1] o que é exatamente o observado. Sabendo que scary é o label 0 e que amusing é label 1, então o *duty cycle* seria 60% para scary e 40% para amusing o que foi exatamente o observado.

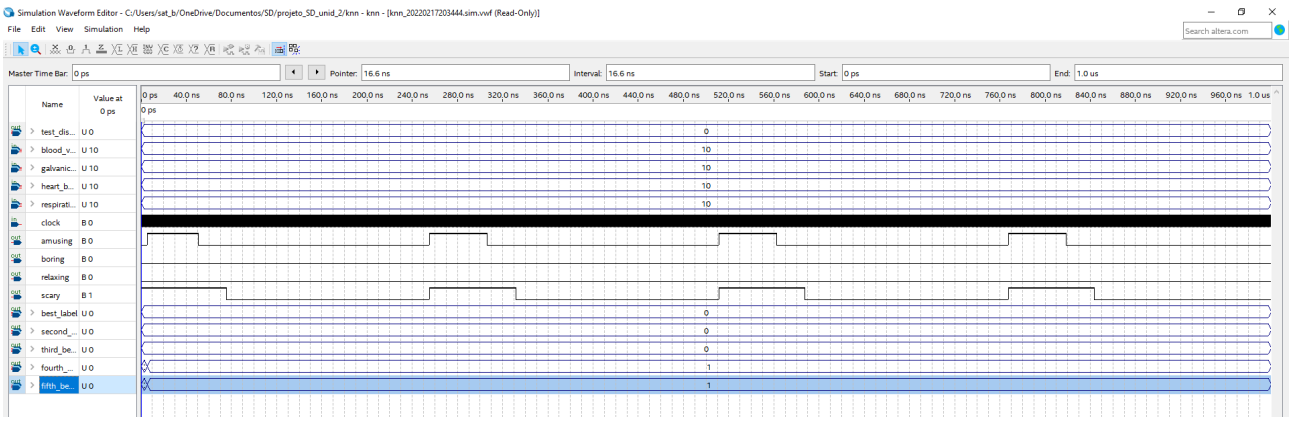

Figura 20: Simulação KNN - scary

3.4 Simulação KNN - boring

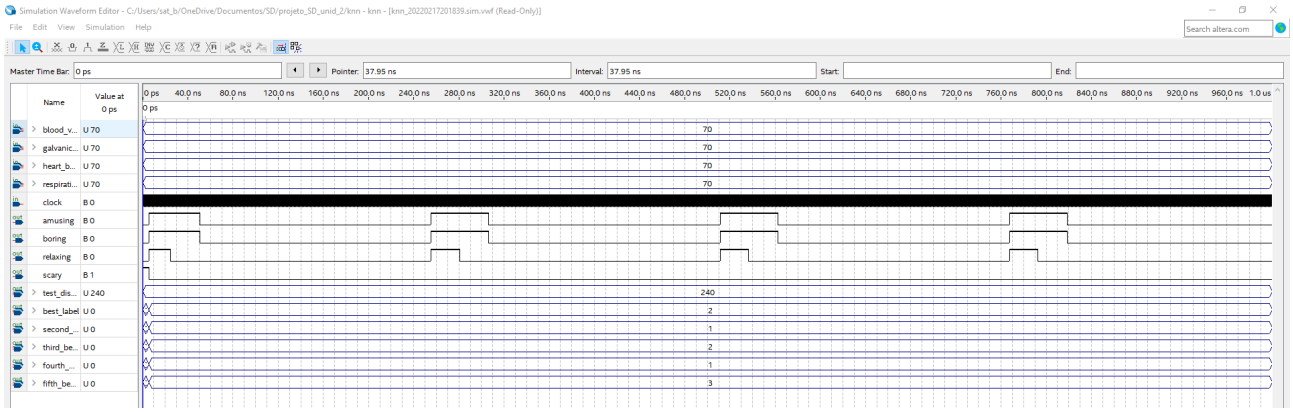

Figura 21: Simulação KNN - boring

Sabendo que boring é o label 2, que os sample_i_70 e sample_i_80 possuem esse label então para testar, foi colocado como entrada o vetor [70,70,70,70] por tanto as melhores amostras seriam sample_i_70, sample_i_80, sample_i_60, sample_i_50, sample_i_90, que possuem os respectivos labels 2, 1, 2, 1, 3 o que foi exatamente o observado.

3.5 Simulação KNN - amusing

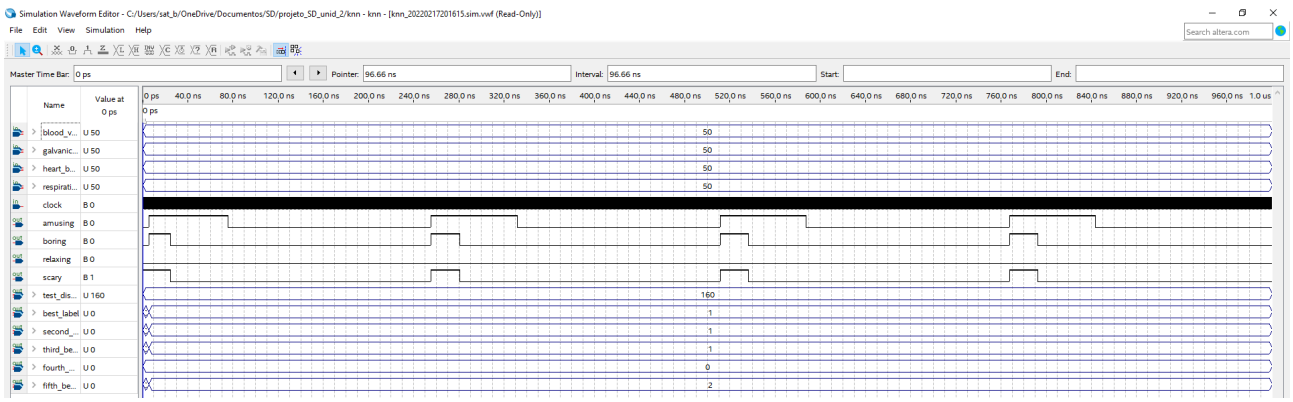

Figura 22: Simulação KNN - amusing

sabendo que amusing é o label de valor 1 e que as amostras: sample_i_40 ,sample_i_50, sample_i_60 possuem esse rótulo foi testado um vetor de entrada de valor [50,50,50,50] cuja as melhores amostram são: sample_i_50, sample_i_40, sample_i_60, sample_i_30, sample_i_70 que possuem os respectivos labels 1, 1, 1, 0, 2, o que foi exatamente o observado.

3.6 Simulação KNN - relaxing

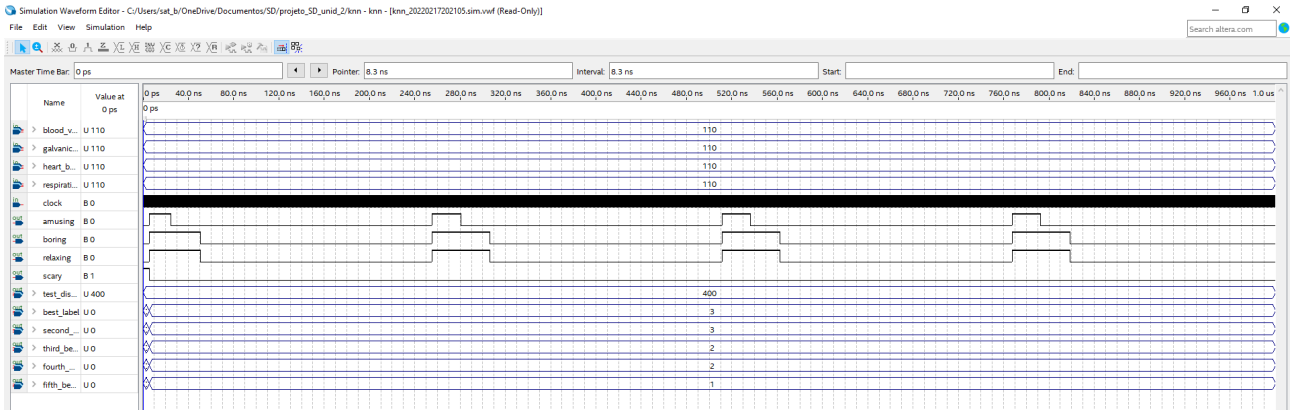

Figura 23: Simulação KNN - relaxing

sabendo que relaxing é o label de valor 3 e que as amostras: sample_i_100, sample_i_90 possuem esse label, então foi testado um vetor de entrada: [110, 110, 110, 110], cuja as melhores amostras seriam: sample_i_100, sample_i_90,sample_i_80, sample_i_70, sample_i_60, que cujo os respectivos labels são 3, 3, 2, 2, 1, o que foi exatamente o observado.