Instituto Superior Técnico

PROJETO DE SISTEMAS DIGITAIS
MEEC 2018-2019

3º Projeto

Algoritmo de classificação k-Nearest Neighbours

Alunos:
João Pedro Cardoso, 84096
Micaela Moraes Serôdio, 84139
Docente:
Horácio Neto

Conteúdo

1 Introdução **1**

2 Unidades de Dados **1**
2.1 Unidade de Dados 1 1
2.2 Unidade de Dados 2 2
2.3 Decisão 3

3 Unidade de Controlo **3**

4 Memória **5**

5 Otimização do Design **6**

6 Recursos Usados, Restrições de Tempo, Latência,Frequência Máxima **7**
6.1 Design Não Otimizado 7
6.2 Design Otimizado 7

7 Conclusão **8**

1 Introdução

O objetivo deste trabalho é desenhar uma unidade de processamento para classificar as espécies de uma flor de íris, a partir de 4 medições, utilizando o algoritmo de classificação (kNN) k-Nearest Neighbor e utilizando a distância Euclideana como métrica. O segundo objetivo deste trabalho consiste na otimização da unidade de processamento projetada.

2 Unidades de Dados

2.1 Unidade de Dados 1

A unidade de processamento de dados representada na Fig. 1, é constituída por 4 subtratores, 4 multiplicadores e 3 somadores instanciados. Nesta unidade utiliza-se pipeline através de registos auxiliares, tal como se pode verificar na Fig. 1 de forma a aumentar a performance do circuito.

Os inputs recebidos nesta unidade são os seguintes:

- B_in: 64 bits, que corresponde a uma palavra da base de dados proveniente da memória. Esta palavra é dividida em 4 palavras de 16 bits B0, B1, B2;
- A_in: 64 bits, que corresponde à instância da flor a ser classificada. Esta instância é dividida em 4 palavras de 16 bits A1, A2, A3 e A4 que correspodem às 4 medidas da flor;
- valid: 1 bit, que funciona como flag que indica à unidade de dados 1 que o output da memória é válido;
- load: 2 bits, em que o bit mais significativo indica se a instância introduzida é igual à instância introduzida anteriormente. Caso este bit esteja a 1 os dados são processados e guardados nos respectivos registos, caso contrário nada é escrito nos registos e o output desta unidade mantém-se inalterado;
- clk: ciclo de relógio do circuito;
- rst: reset do circuito, que indica se os dados devem ser reiniciados;

Quanto aos outputs desta unidade:

- C: 32 bits (Q6.26), que corresponde ao resultado das operações efetuadas;
- valid_result: 1 bit, que funciona como uma flag que indica se o resultado desta unidade é válido.

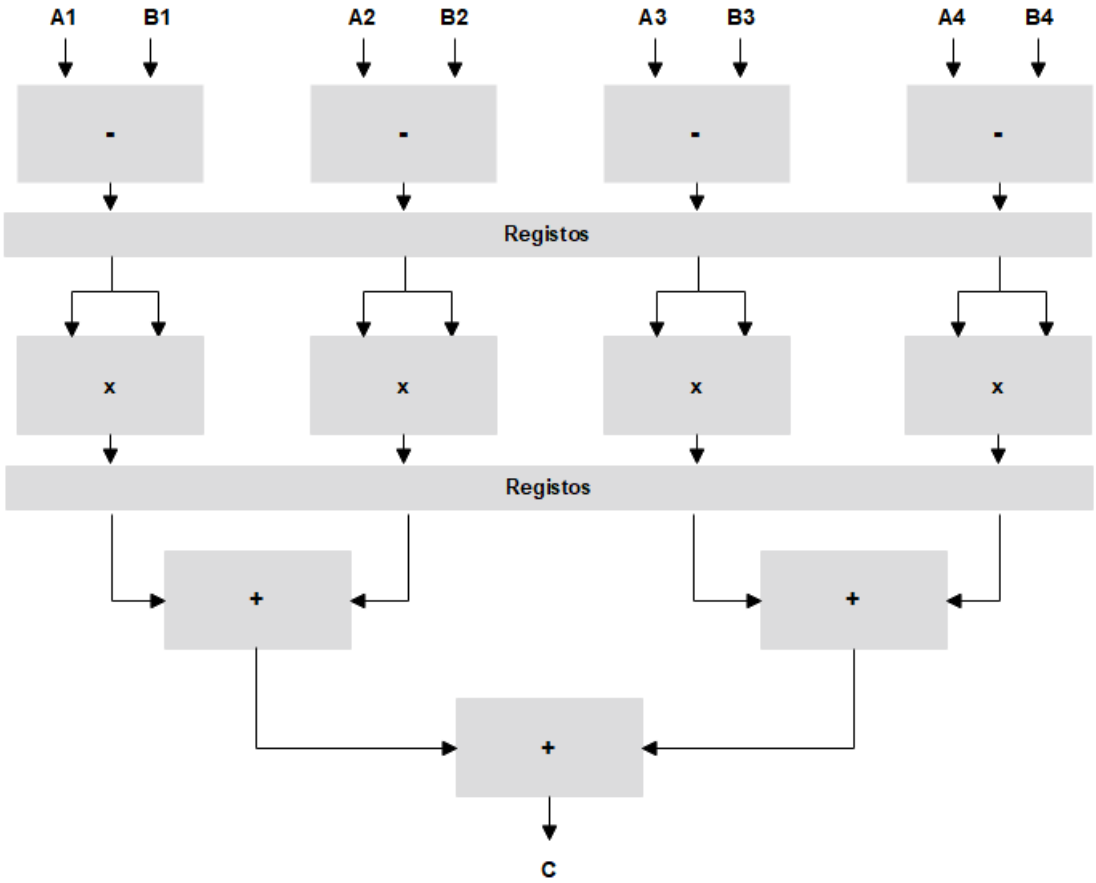

Figura 1: Fluxo na Unidade de Dados 1.

2.2 Unidade de Dados 2

Após o calculo do quadrado da distância eucladiana, é preciso comparar esse valor com os 5 valores mais baixos que estão guardados em um registo. Após comparar com os 5 valores em paralelo, cria 5 bits para fazer load aos registos. Este algoritmo chama-se insert sort.

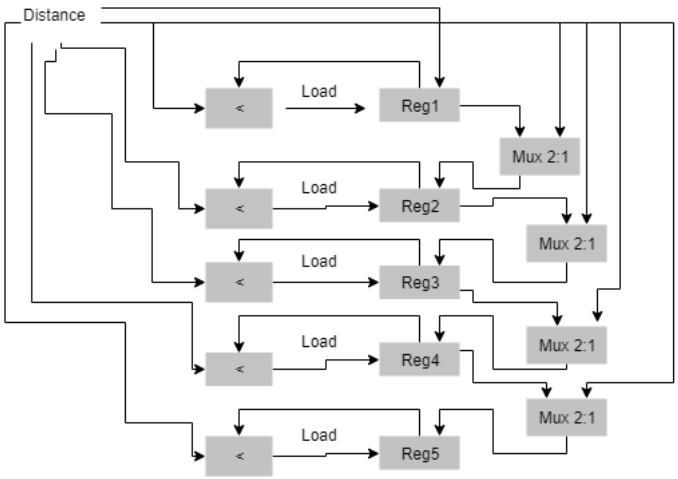

Figura 2: Unidade de Dados 2: Guarda os 5 valores mais baixos enquanto os dados que chegam são válidos.

2.3 Decisão

A acompanhar os dados vindos da memória, vem um bit que diz se os valores são válidos ou não.Quando os valores deixam de ser válidos(negative edge) e é recebido após os andares de pipeline. É calculado o número de cada classe presente nos registos. A classe com maior valor é o Output deste bloco e do Sistema.

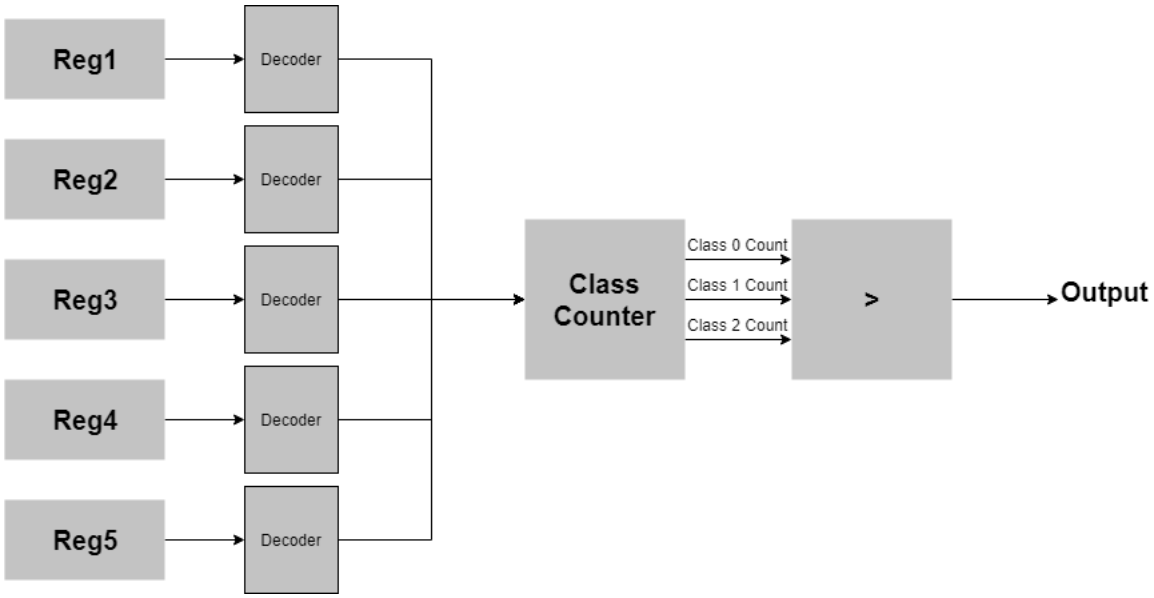

Figura 3: Cálculo do Output do sistema

3 Unidade de Controlo

A unidade de controlo representada pelo diagrama de estados da Fig. 4, é constituída por quatro estados: o estado inicial ($s\_initial$), o estado de execução ($s\_new\_instance$) e o estado final (s_end).

Esta unidade é controlada por 5 inputs: o sinal de ciclo de relógio (clk), o sinal de reset (rst), o sinal de partida($init$), o sinal de option ($option$) que controla a entrada no estado de execução em conjunto con o sinal de partida, o sinal de termino ($done$), a instância a ser classificada ($new\_instance$) e o k (k) que será apenas transmitido para a unidade de dados 2 (datapath2.vhd).

Os outputs desta unidade são: os enables do registos da unidade de dados 1 e 2 ($load(0)$ para o datapath2 e $load(1)$ para o datapath1), o k_out que vai diretamente para a unidade de dados 2, a instância a ser classificada que é apenas transmitida do input para o output desta unidade,de modo a que se o utilizador alterar a flor de íris a ser classificada a meio do processo de classificação de uma flor, essa classificação não será alterada. E, ainda, uma flag que indica à unidade de processamento da FPGA que a classificação da flor de íris foi encontrada e que está preparada para ser apresentada.

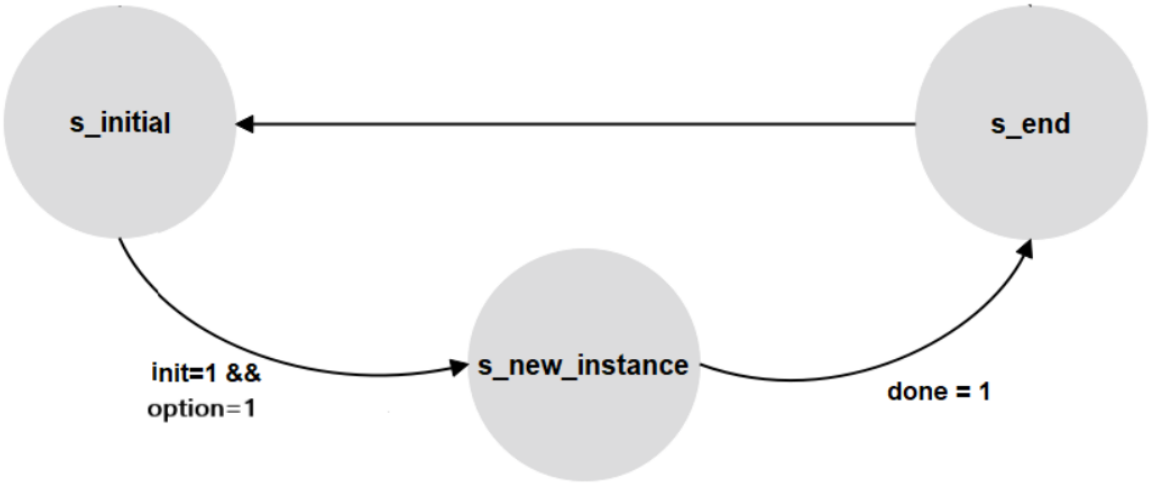

Figura 4: Diagrama de estados.

Tabela 1: Descrição dos estados.

ESTADOS	DESCRIÇÃO
s_initial	Estado em que se recebe os inputs, se inicializa os outputs e se aguarda, por init =1 && rst =0 && option=1.
s_new_instance	A flor de íris é processada, efetuando-se todas as operações do datapath1 e datapath2 necessárias. Neste estado a classificação da flor é "encontrada".
s_end	Estado em que a classificação da flor de íris para o respectivo k é obtida e é válida, podendo ser então apresentada no display de 7 segmentos da FPGA.

4 Memória

As instâncias de treino são guardadas numa single port ROM. Quando está no estado "s_new_instance", um contador começa a gerar endereços e o valor de válido passa a 1. Quando o gerador de endereços chega ao fim, o válido passa a 0. Para o primeiro design, a memória tem 108 palavras de 64 bits (16*4), ou seja, uma instância completa.As Classes estão noutra ROM com 108 palavras de 2 bits.

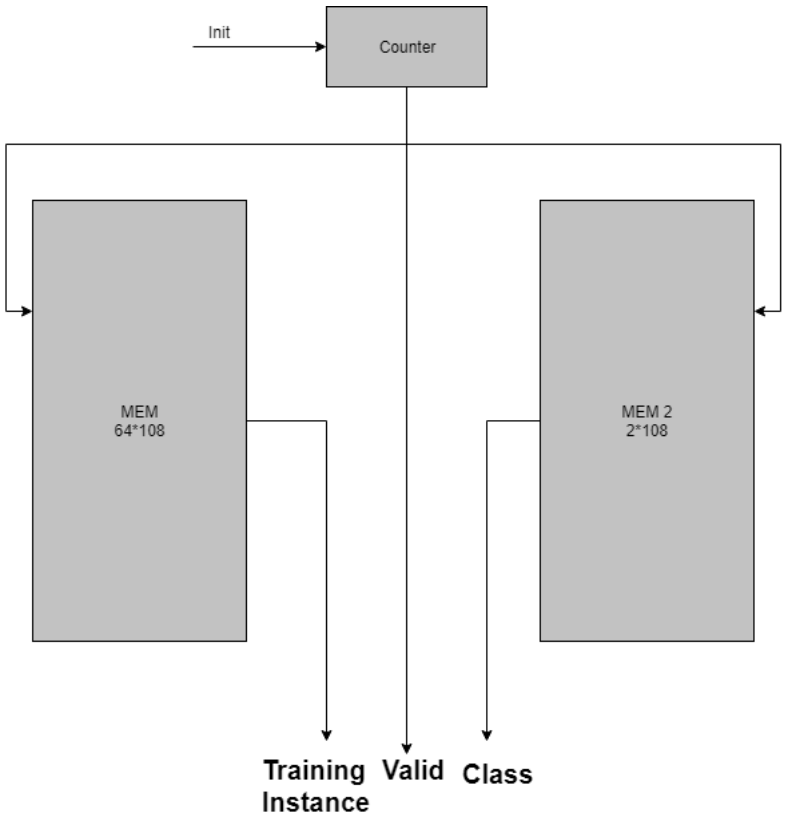

Figura 5: Unidade de Memória com as instâncias de treino

5 Otimização do Design

Para otimizar a performance, foi analisada o bottleneck do sistema, que neste caso era a bandwith da memória. Ao aumentar a memoria para 6 palavras de 1152, pode-se calcular 18 instâncias completas ao mesmo tempo. Para acomodar esta mudança, na unidade de dados 1, multiplica-se por 18. Para a unidade de dados 2, já é menos trivial.Para ter 5 registos finais em que tem os dados minimos, faz-se 18 unidades de dados 2, que trata cada resultado de cada unidade anterior e uma nova unidade de processamento "compare"que em 10 registos,faz output dos 5 de valor mais baixo como está na figura seguinte.

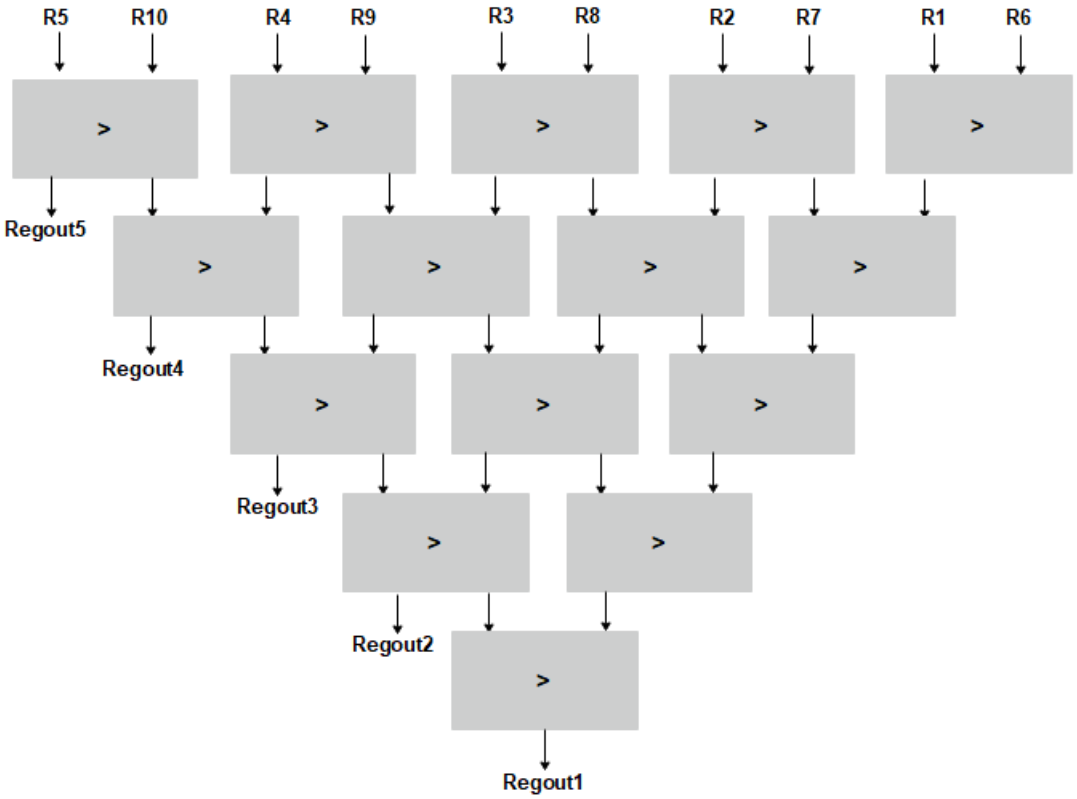

Figura 6: Unidade de processamento de dados "compare"

Depois é analisado em árvore os dados para obter os 5 mais próximos de 0.

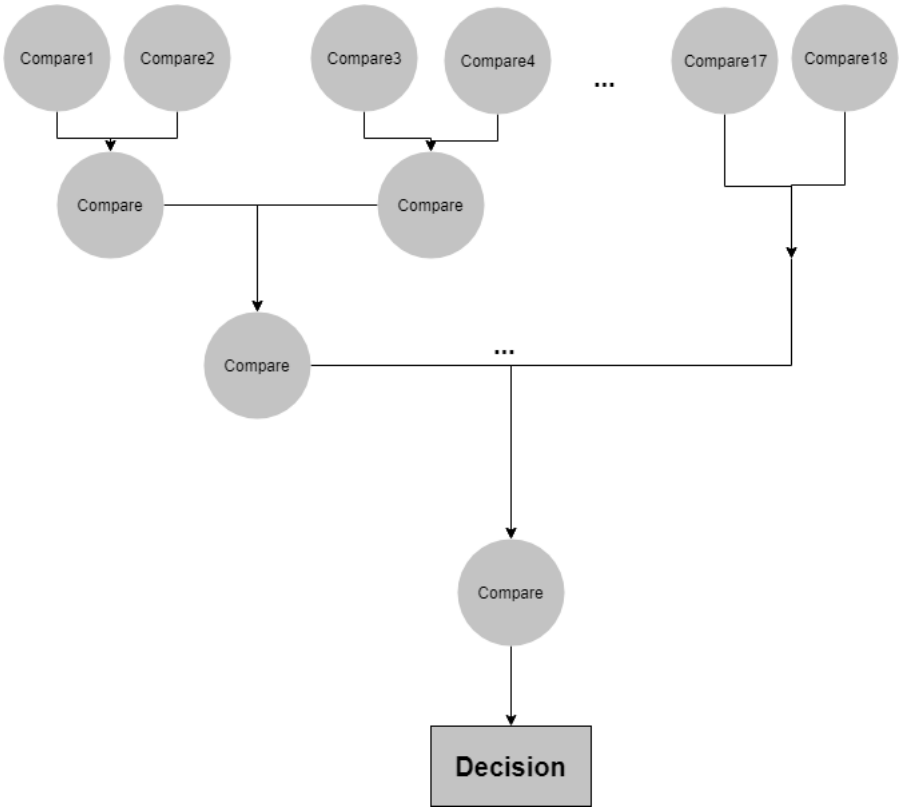

Figura 7: Processamento de dados em árvore para chegar aos 5 registos de valor mais baixo

6 Recursos Usados, Restrições de Tempo, Latência,Frequência Máxima

6.1 Design Não Otimizado

Através da análise do ficheiro "Utilization Summary"do projeto não-otimizado verificou-se que foram usados os seguintes recursos: 410 LUT's de 20800 disponíveis (1.97%); 350 FF's de 41600 (0.84%); 4 DSP de 90(4.44%); 50 IO's de 106(47.17%); 3 BUFG de 32(9.38%); 6 LUTRAM de 9600(0.06%); 2 BRAM de 50(4.00%). Para se definir as "Timming Contraints"criou-se um ciclo de relógio de 10 ns, de modo a que fosse possível efetuar o caminho crítico num ciclo de relógio sem otimizar o clock para o máximo possivel.

Quanto ao ficheiro "Timming Summary"verificou-se que todas as "timming contraints"foram atingidas, sendo que WNS foi de 3.457 ns, WHS 0.108 ns e WPWS 5 ns.

A frequência máxima é dada pelo inverso do período mínimo, que neste caso corresponde ao clock definido - Worst Negative Slack. O período mínimo é 10 ns-WNS, isto é, 6.543 ns. Logo, a frequência máxima é de, aproximadamente, 153 MHz.

6.2 Design Otimizado

Através da análise do ficheiro "Utilization Summary"do projeto otimizado verificou-se que foram usados os seguintes recursos: 17335 LUT's de 20800 disponíveis (83.34%); 19262 FF's de 41600 (46.30%); 72 DSP de 90(80.00%); 73 IO's de 106(68.87%); 1 BUFG de 32(3.13%); 2 LUTRAM de 9600(0.02%); 16 BRAM de 50(32.00%). Para se definir as "Timming Contraints"criou-se

um ciclo de relógio de 6.5 ns, de modo a que fosse possível efetuar o caminho crítico (compare) num ciclo de relógio e que o Worst Negative Slack fosse o menor possível de forma a otimizar a latência do circuito.

Quanto ao ficheiro "Timming Summary"verificou-se que todas as "timming contraints"foram atingidas, sendo que WNS foi de 0.247 ns, 0.027 WHS ns e WPWS 4.5 ns.

A frequência máxima é dada pelo inverso do período mínimo, que neste caso corresponde ao clock definido - Worst Negative Slack. O período mínimo é 6.5 ns-WNS, isto é, 6.253 ns. Logo, a frequência máxima é de, aproximadamente, 159 MHz teórico e 153.84 implementado na placa(período de 6.5 ns).

A latência do circuito, ou seja, o tempo preciso para ter um output á saída é de 14 Ciclos*6.5ns = 91 ns como pode-se ver na seguinte figura.

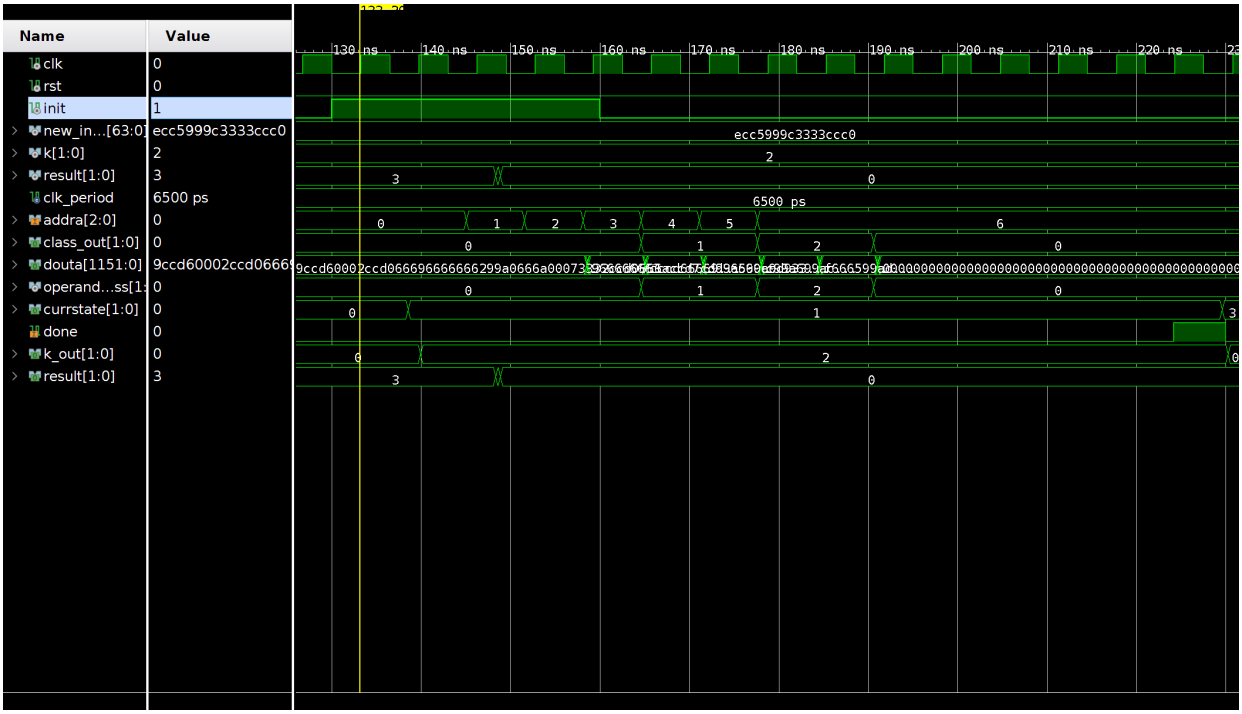

Figura 8: Simulação Pós Implementaton. Quando done=1, quer dizer que acabou de analizar todos os dados

7 Conclusão

Em suma, verifica-se que ambas as implementações, otimizadas e não-otimizadas, efetuam as operações desejadas, uma vez que apresentam as classificações das flores de íris corretamente.

As decisões de otimização do circuito foram tomadas com o objetivo de diminuir a latência e, consequentemente, melhorar a performance do circuito.

Analisando os recursos utilizados bem como os recurso temporais utilizados em ambas as implementações (secção 6), concluiu-se que, de facto, houve uma melhoria na performance do circuito com as otimizações da secção "Otimização do Design".

Concluindo, os objetivos foram atingidos, tanto no que diz respeito ao design como à simulação temporal pós-implementação, que descreve o comportamento mais próximo do comportamento real do circuito implementado numa FPGA e o próprio comportamento na FPGA.