Regressão Linear

Projeto - MC613

Rodrigo Seiji Piubeli Hirao - 186837 - rodrigo.seiji.hirao@gmail.com
Luiz Eduardo Araujo Zucchi - 183006 - luiz.e.a.zucchi@gmail.com

20 de Junho de 2018

Conteúdo

1 Teoria **4**

2 Descrição **5**
2.1 Diagrama de Blocos 5
2.2 Implementação 5
2.2.1 Pontos Fixos 5
2.2.2 Memória 6
2.2.3 Máquina de Estados 6
2.2.4 Teclado 7
2.2.5 Monitor 7

3 Resultado **9**
3.1 O que faltou 11

4 Conclusão **11**

5 Bibliografia **11**

Lista de Figuras

1 Diagrama de Blocos 5
2 Máquina de Estados - Regressão Linear 6
3 Máquina de Estados - Monitor 8
4 Waveform - baixa precisão 9
5 Monitor 9
6 FPGA 10

github: https://github.com/Rep-MFAB/mc613

Resumo

Será implementado um algoritmo de regressão linear para machine learning, usando um teclado como entrada, para fazer o treinamento do machine learning, um monitor como saída, para mostrar o atual custo da função e o resultado da função treinada.

1 Teoria

No projeto, foi desenvolvido um circuito que realiza o algoritmo de regressão linear, que é um algoritmo comumente aplicado a Machine Learning. A motivação para tal circuito é mostrar que é possível, e prático, a manufatura de circuitos que executam esse algoritmo, sem a necessidade de implementá los através de uma linguagem de programação que irá exigir mais recursos em termos de hardware para operar.

A regressão linear é um método simples, que consiste em achar um mínimo (ou máximo) local de uma função, através do método gradiente e de uma função custo que utiliza do método dos mínimos quadrados.

Para realizar essa tarefa, iremos separar o processo em duas etapas que irão consistir em:

- passar dados com suas respectivas respostas esperadas e com isso calcular uma função que será otimizada a medida que adicionamos mais dados,
- Desenhar um gráfico da funçã gerada no monitor.

Para achar a função em questão, partiremos da função hipótese:

$$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i$$

Onde x_i são as entradas, n é o número de parâmetros e θ_i são os argumentos que queremos encontrar

A função de custo, que deve ser minimizada, se baseia no método dos mínimos quadrados, sendo a função:

$$J(\theta) = \frac{1}{2n} \sum_{i=i}^{n} (h_\theta(x_i) - y_i)^2$$

Onde y_i é o resultado esperado para x_i

O método gradiente consiste em iterar sobre:

$$\theta_i := \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$$

Onde α é a taxa de aprendizado. Vale ressaltar que um α muito grande pode causar uma alta imprecisão, pois acabamos sempre pulando o mínimo, e um α muito pequeno pode gerar um tempo de execução muito longo, existem heurísticas para determinar α, porém elas não serão abordadas

2 Descrição

O sistema em si (sem contar os periféricos), é formado pelo circuito que realiza os cálculos relacionados a regressão linear [linear regression.vhd], pela memória [ram.vhd] e pela máquina de estados [linear regression fsm.vhd] que controla a regressão linear. Conectados a esse sistema, estão os periféricos que são um teclado numérico para a entrada de dados e um monitor que mostra os dados sendo digitados, os pontos no gráfico e por fim a reta que vem da regressão linear.

2.1 Diagrama de Blocos

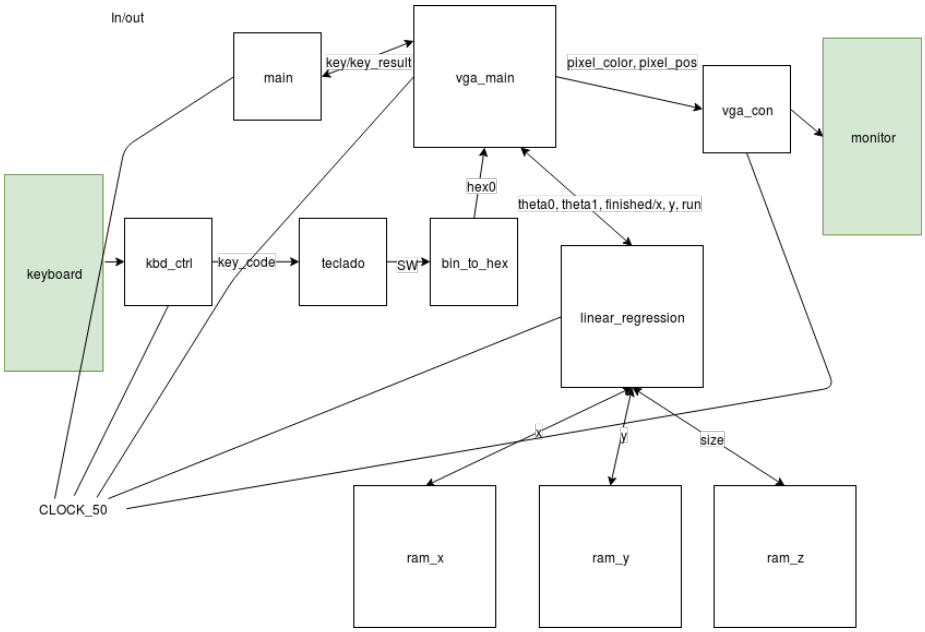

Figura 1: Diagrama de Blocos

2.2 Implementação

A implementação de [linear regression.vhd] consiste em realizar as fórmulas referidas na parte teórica.

2.2.1 Pontos Fixos

Para uma regressão linear precisa, é necessário a utilização de casas decimais nas contas, para evitar a necessidade de implementar representação por ponto flutuante, que é complexa, utilizamos um ponto fixo: Um vetor, guardando a parte decimal numa metade, e a parte fracionária em outra. Tal escolha facilitou as operações, e reduziu pouco a eficiência.

2.2.2 Memória

O algoritmo usa 3 memórias de 256 palavras, com 8 bytes cada, a motivação para isso, é que não há necessidade de um conjunto muito vasto de dados, porém, com 8 bytes, é possível alcançar uma boa precisão para cálculos. As memórias são usadas para armazenar:

1. os valores em um vetor de entradas
2. os valores em um vetor de saída
3. outros dados (como tamanho do vetor)

2.2.3 Máquina de Estados

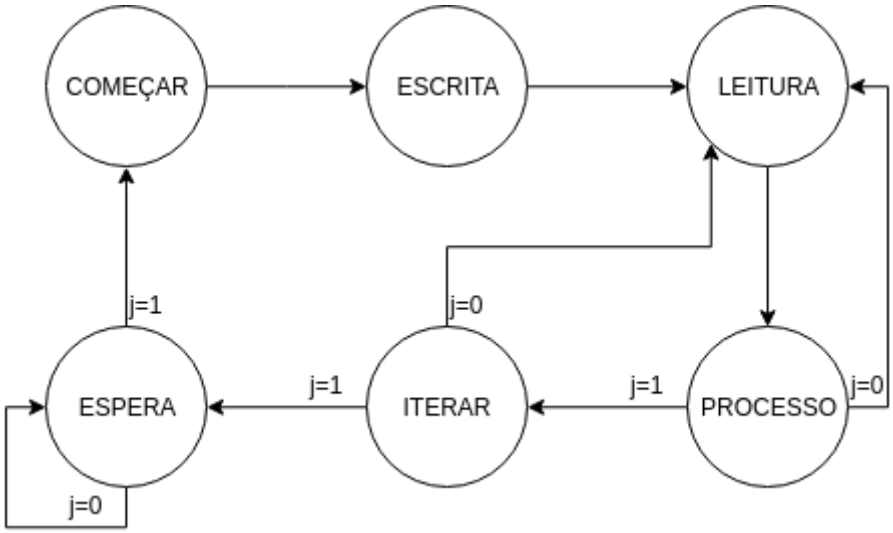

Figura 2: Máquina de Estados - Regressão Linear

A máquina de estados que controla a regressão linear está sincronizada com a borda de descida do [CLOCK 50], ao contrário do resto dos componentes. Essa escolha foi feita para poder resolver algumas instabilidades que ocorriam ao mudar as flags ao mesmo tempo que o estado, além de melhorar a performance do algoritmo. Vale notar que a máquina é sensível ao [CLOCK 50] apenas quando uma flag [run] estiver alta.

A máquina começa no estado [COMEÇAR] que serve apenas marcar um início, indo direto para [ESCRITA], onde é incrementado o tamanho do vetor, e escrito a nova entrada e saída da função na memória. Logo após, é inciado o estado de [LEITURA], onde ele lê o primeiro valor do vetor, e inicia o [PROCESSO] para colocar na somatória. Assim a máquina de estados fica iterando sobre [LEITURA] e [PROCESSO] até que uma flag seja ativada, simbolizando que todo o vetor foi lido. Após a somatória estar feito, o estado [ITERAR] se encarrega de calcular os θ's e reiniciar

o processo de iteração desde [LEITURA] por mais algumas vezes. Após o fim da iteração, a flag será ativada e o último estado [ESPERA] será ativado, ativando a flag de que o algoritmo acabou e esperando ser desligado. Assim, só voltando para [COMEÇAR] quando o [run] voltar a ser ativado.

2.2.4 Teclado

O teclado seguiu o que foi visto em sala, ou seja usa [kbd ctrl.vhd] como máquina de estados para controlar a ordem em que as teclas são processadas. Basicamente ao apertar uma tecla, recebemos o código relativo a tecla e convertemos para o código em binário, se a tecla foi um número, o código será o número escrito em binário, se for enter, backspace ou vírgula, teremos um número em binário entre 10 e 12.

2.2.5 Monitor

A placa usa do componente [vgacon.vhd] para conectar com o monitor, e inicializa ela usando o [monitor mem.mif]

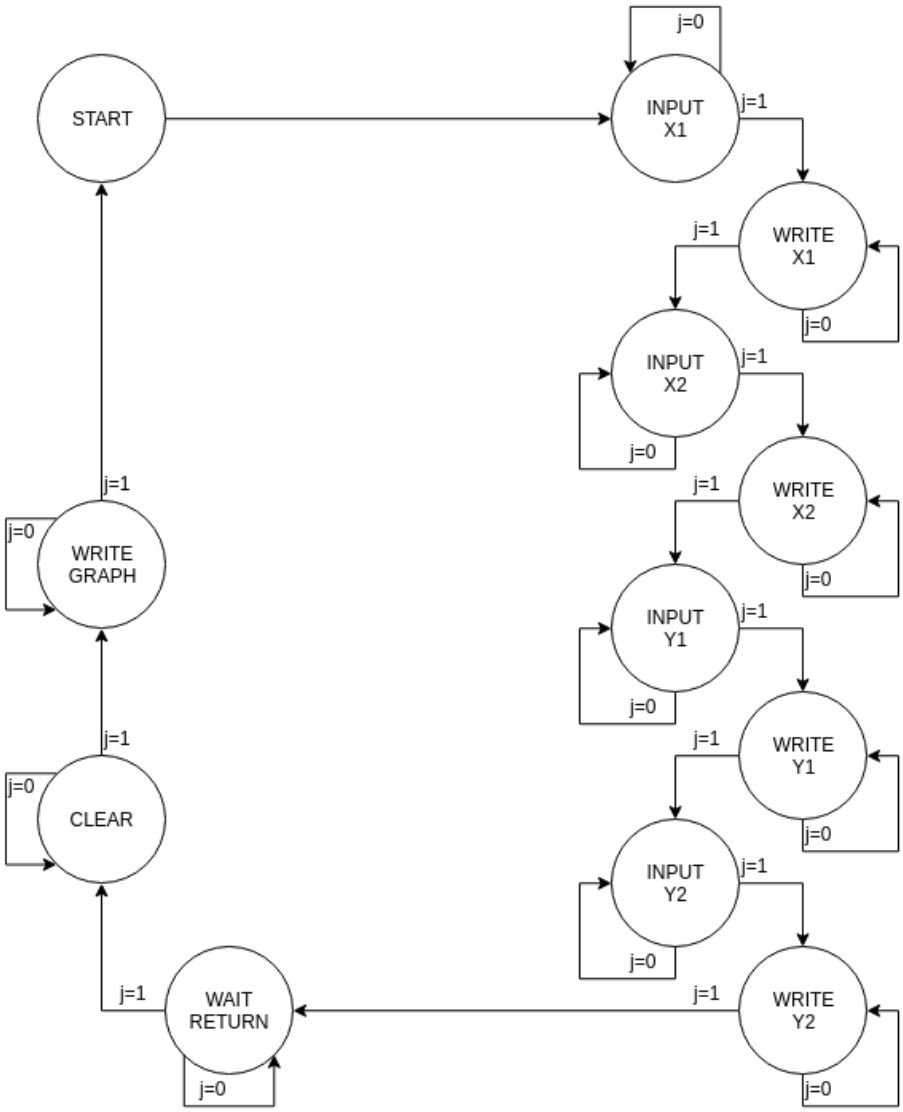

Figura 3: Máquina de Estados - Monitor

O monitor usa uma máquina de estados [vga main fsm.vhd] que está sincronizado com a borda de descida de um clock de 10Hz. Pois 50MHz era muito rápido para a transição de estados.

A máquina começa no estado [START], assim como o [COMEÇAR] da máquina de estados da regressão linear, passando para o [INPUTX1], que espera o input numérico do teclado para armazenar o valor e ligar a flag, passando para o estado [WRITEX1], que desenha o número na tela, na posição correta, os próximos [INPUT]'s e [WRITE]'s fazem a mesma

coisa, mas para variáveis e posições diferentes.

$$x = 10 * X1 + X2$$
$$y = 10 * Y1 + Y2$$

Assim que escritos X1, X2, Y1 e Y2, o estado [WAIT RETURN] irá esperar o usuário pressionar Enter, para limpara a tela em [CLEAR] e desenhar o gráfico em [WRITE GRAPH], iterando 100 vezes incrementando o valor de x, e calculando o valor de y.

$$x = x + 1$$
$$y = \theta_0 + \theta_1 * x$$

3 Resultado

O algoritmo foi testado com $\alpha = 4200$ e precisão de inteiro, o que resultou na seguinte waveform.

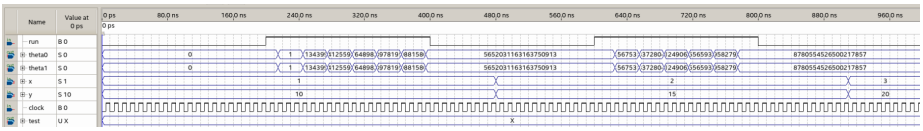

Figura 4: Waveform - baixa precisão

Que nunca se aproxima do valor correto por sua baixa precisão e alta taxa de atualização.

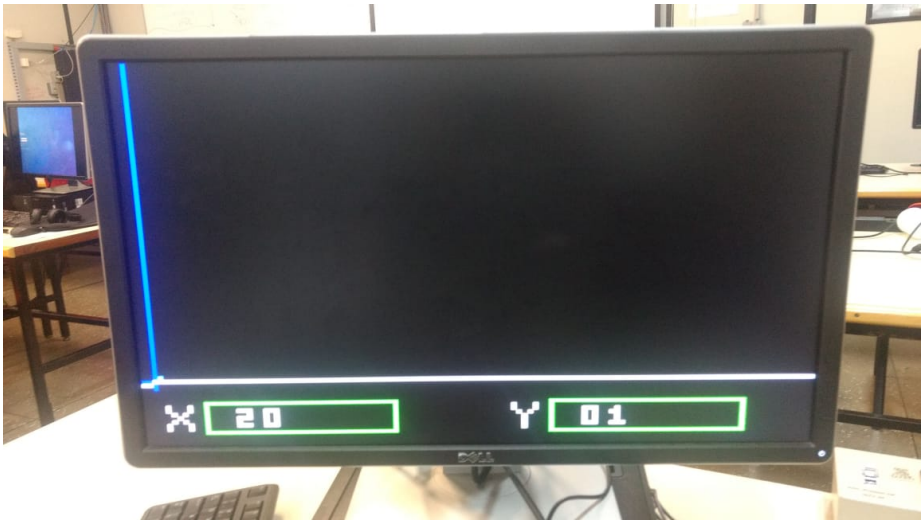

Figura 5: Monitor

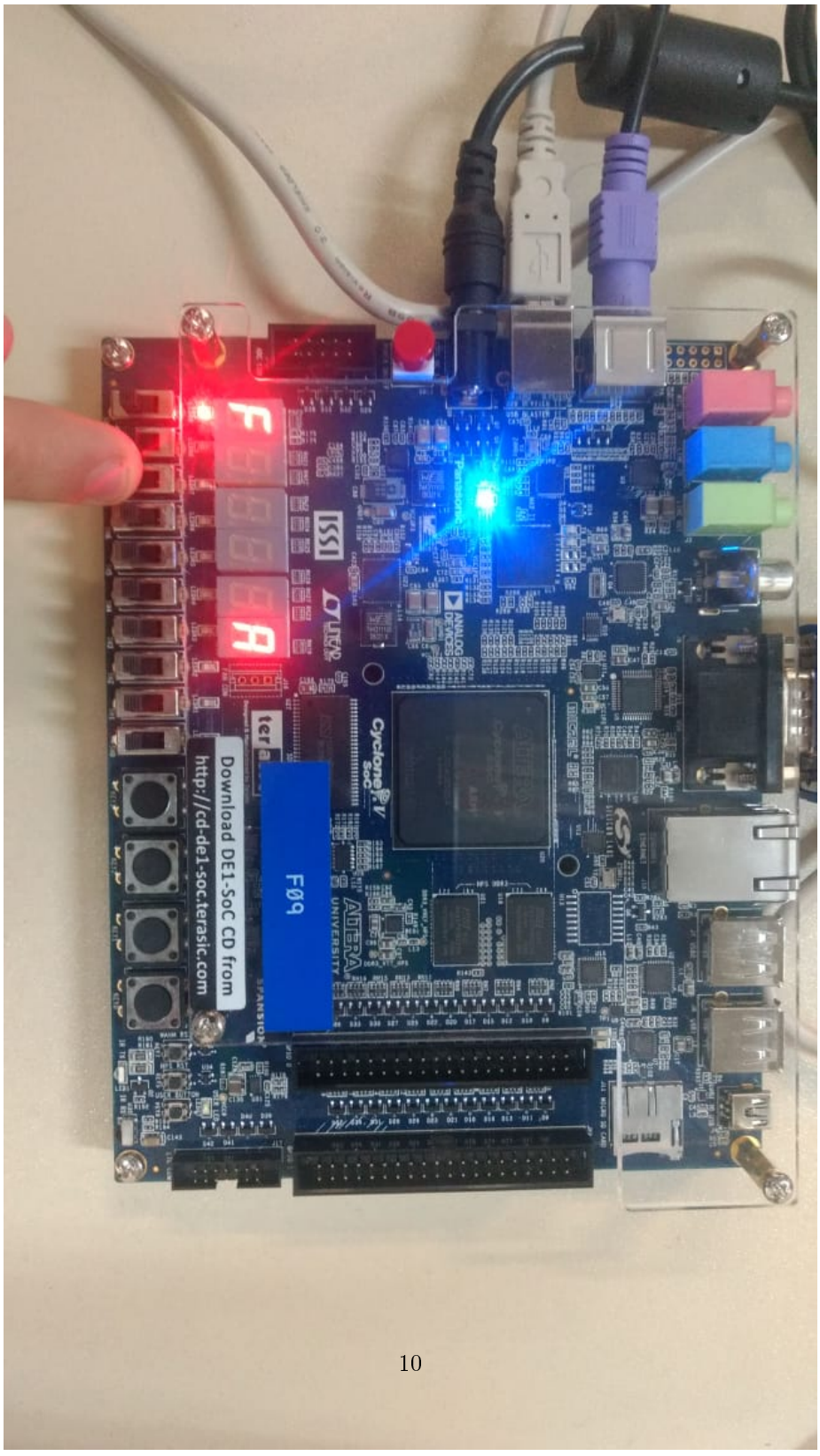

Figura 6: FPGA

Então foi testado no monitor, com precisão de ponto fixo de 32 casas. Mas a precisão fez com que todos os valores virassem 0

3.1 O que faltou

O algoritmo, embora pronto, ainda não foi completamente implementado no montitor.

Os seguintes bugs foram encontrados:

- Limpar a tela, não funciona
- Desenho do gráfico sai dos limites

Os seguintes itens poderiam ainda ser implementados:

- Desenhar os pontos do vetor na tela
- Função de apagar dígito (Backspace)
- Função de mudar de campo (Tab)
- Suporte para input de números negativos
- Suporte para input de números decimais
- Aumentar a resolução

4 Conclusão

A conclusão obtida após a conclusão do projeto é que é possível implementar um metodo de machine learning em uma placa fpga, através da linguagem vhdl. Como placas fpga geralmente tem baixo consumo e custo menor, seria possível considerar a implementação deste e de outros algoritmos em fpga's para economizar nos custos desse tipo de projeto, que normalmente são em hardware para aceleração de treinamento Além disso, notamos a diferença na eficiência da representação por pono flutuante e a representação utilizada por nós. Com algumas otimizações e revisões, acreditamos que tornariamos a implementação de regressão linear numa fpga algo com utilidade real.

5 Bibliografia

1. Yan, Xin (2009), Linear Regression Analysis: Theory and Computing.
2. Nievergelt, Yves (1994). "Total Least Squares: State-of-the-Art Regression in Numerical Analysis"