Análise Qualitativa da Implementação de Regressão Linear em FPGA

Davi Mortari Vargas

Universidade Federal de Mato Grosso do Sul

2 de dezembro de 2025

Sumário

1 Introdução **3**

2 Arquitetura e Metodologia do Projeto **3**
2.1 Hierarquia de Entidades 3
2.2 Interface e Formato dos Dados 4
2.3 Escolhas de Design Críticas 4
2.3.1 Aritmética de Ponto Fixo 4
2.3.2 Paralelismo e Pipeline 4

3 Resultados e Análise da Implementação **5**
3.1 Utilização de Recursos 5
3.2 Análise do RTL Viewer 5
3.3 Análise de Timing 6

4 Discussão e Trabalhos Futuros **6**

5 Conclusão **7**

1 Introdução

Este relatório apresenta uma análise qualitativa do projeto de hardware para aceleração do cálculo de regressão linear utilizando um Dispositivo Lógico Programável (FPGA). O objetivo principal do trabalho é avaliar a arquitetura proposta, as escolhas de design e os resultados obtidos após a compilação na ferramenta Intel Quartus Prime.

A regressão linear é um método estatístico fundamental para modelar a relação entre uma variável dependente e uma ou mais variáveis independentes. A solução por equações normais é dada por:

$$\beta = (X^T X)^{-1} X^T Y$$

onde β são os coeficientes da regressão. A implementação desta equação em hardware, especialmente as operações de multiplicação e inversão de matrizes, apresenta uma oportunidade significativa para aceleração computacional.

2 Arquitetura e Metodologia do Projeto

A arquitetura do sistema foi projetada de forma modular em VHDL para refletir diretamente as operações matemáticas da equação normal.

2.1 Hierarquia de Entidades

O sistema é composto por um conjunto de entidades VHDL que colaboram para executar o cálculo. A hierarquia é liderada pela entidade de topo `MatrixLinearRegression`, que gerencia o fluxo de dados e controle entre os seguintes módulos principais:

- `MatrixLinearRegression`: A entidade de mais alto nível, responsável por instanciar todos os outros módulos e definir a sequência de operações: recepção de dados, cálculo e transmissão dos resultados.
- `DataReceiver`: Módulo responsável por receber os dados das matrizes X e Y através da interface serial e armazená-los internamente.
- `MatrixOperations`: Um módulo de controle que orquestra as chamadas para os módulos de cálculo matricial na ordem correta (Transposição, Multiplicação, Inversão).
- `MatrixTranspose`: Componente dedicado que realiza a transposição da matriz de entrada X.
- `MatrixMultiplier`: Bloco de hardware combinacional massivamente paralelo que calcula o produto de duas matrizes. É instanciado múltiplas vezes para os cálculos de $X^T X$ e $X^T Y$.
- `MatrixInverter`: Implementa o algoritmo de inversão de matrizes, um processo sequencial controlado por uma máquina de estados interna.

- `DataTransmitter`: Após o término dos cálculos, este módulo serializa os coeficientes da matriz β resultante e os envia pela interface UART.
- `SerialUartCommunication`: Módulo de baixo nível que implementa o protocolo de comunicação serial (UART), tanto para recepção quanto para transmissão de bytes.

2.2 Interface e Formato dos Dados

A entidade de topo `MatrixLinearRegression` possui a seguinte interface externa:

clk: Sinal de clock principal do sistema, que sincroniza todas as operações síncronas.

reset: Sinal de reset assíncrono para inicializar o sistema a um estado conhecido.

rx: Linha de recepção de dados da interface serial (UART).

tx: Linha de transmissão de dados da interface serial (UART).

init_transmission: Sinal de controle para iniciar o processo de cálculo.

Internamente, os dados numéricos são representados utilizando o tipo `matrix_type`, definido no pacote `MatrixPackage`. Este tipo implementa matrizes de números em ponto fixo no formato Q8.8 (16 bits no total, com 8 bits para a parte inteira e 8 para a parte fracionária), permitindo um balanço entre a faixa dinâmica e a precisão dos cálculos.

2.3 Escolhas de Design Críticas

2.3.1 Aritmética de Ponto Fixo

Uma decisão fundamental foi a utilização de aritmética de ponto fixo em vez de ponto flutuante. Esta escolha visa otimizar o uso de recursos e a velocidade de operação no FPGA, uma vez que operações de ponto fixo são significativamente menos complexas em hardware. Foi adotado o formato Q8.8, com 8 bits para a parte inteira e 8 para a fracionária, buscando um balanço entre a precisão dos resultados e a área de silício consumida.

2.3.2 Paralelismo e Pipeline

A arquitetura busca explorar o paralelismo inerente ao hardware. O módulo `MatrixMultiplier`, por exemplo, foi projetado para explorar o processamento em paralelo. A arquitetura do MatrixMultiplier explora intensivamente o paralelismo de dados. Utilizando a construção generate, o design instancia um bloco de cálculo dedicado para cada elemento da matriz de resultado. Isso significa que todos os produtos escalares necessários para formar a matriz de saída são calculados simultaneamente, em vez de sequencialmente, maximizando o throughput teórico do módulo. Apesar do alto grau de paralelismo, o bloco de cálculo individual é implementado como um circuito puramente combinacional. A ausência de um clock no processo de cálculo e

de registradores intermediários indica que a técnica de pipeline não foi utilizada. Em contraste, a inversão de matriz é um processo inerentemente mais sequencial, executado por uma máquina de estados que controla cada passo do algoritmo.

3 Resultados e Análise da Implementação

O projeto foi compilado utilizando o software Intel Quartus II 64-Bit versão 13.1.0, tendo como alvo um dispositivo da família Cyclone IV.

3.1 Utilização de Recursos

A compilação resultou no uso de recursos conforme detalhado na Tabela 1.

Tabela 1: Relatório de Utilização de Recursos do Fitter.

Recurso	**Utilizado**	**Total Disponível**	**Utilização (%)**
Células Lógicas	16,836	29,440	57%
Registradores Lógicos	5,488	29,440	19%
Pinos de I/O	5	106	5%
Bits de Memória	0	1,179,648	0%
Multiplicadores (9-bit)	100	132	76%
PLLs	0	6	0%

A compilação do projeto na ferramenta Quartus permite quantificar o custo em hardware da arquitetura proposta. É importante notar que FPGAs modernos não são medidos em "portas lógicas"simples (como AND, OR), mas em blocos mais complexos chamados **Elementos Lógicos (LEs)** ou Células Lógicas. Cada LE contém, tipicamente, uma Look-Up Table (LUT) - capaz de implementar qualquer função lógica de 4 entradas -, um flip-flop e hardware dedicado para operações de soma.

O projeto foi sintetizado e implementado para o dispositivo Altera Cyclone IV GX, modelo EP4CGX30BF14C6. A implementação final consumiu 16.836 elementos lógicos, o que representa 57% da capacidade total do dispositivo. Este número reflete a complexidade da lógica de controle e dos caminhos de dados necessários para orquestrar as múltiplas operações matriciais sequenciais.

O ponto mais significativo da análise de recursos é a utilização de 100 elementos multiplicadores de 9 bits, correspondendo a 76% do total de recursos de DSP do dispositivo. Este dado comprova que a ferramenta de síntese mapeou com sucesso as operações de multiplicação do VHDL para os blocos de hardware dedicados, o que é a principal fonte de aceleração para o algoritmo de regressão linear nesta plataforma.

3.2 Análise do RTL Viewer

A visualização RTL (Register-Transfer Level) gerada pelo Quartus, apresentada na Figura 1, confirma a estrutura modular descrita. É possível identificar claramente os

blocos funcionais e o fluxo de dados entre eles, validando a corretude da arquitetura proposta.

3.3 Análise de Timing

A análise de timing inicial, realizada sem um arquivo de restrições (SDC), reportou falhas críticas.

- **Causa:** Ausência de um arquivo SDC, levando o TimeQuest a assumir um clock irrealista de 1 GHz (período de 1 ns).
- **Consequência:** Slack de setup negativo de **-519.713 ns**, indicando que o design não cumpria os requisitos de tempo para esta frequência impossível.

Para uma análise correta, foi necessário criar o arquivo `MatrixLinearRegression.sdc` para definir o clock de operação real do sistema.

```
# Constrain the main clock port to 50 MHz (20 ns period)
create_clock -period "20ns" [get_ports clk]
```

Listing 1: Conteúdo do arquivo SDC para um clock de 50 MHz.

Após a re-compilação com o SDC, a análise de timing obteve sucesso, resultando em uma frequência máxima de operação (Fmax) de **[XX.XX MHz]**. Este resultado indica que o design é robusto e pode operar estavelmente na frequência definida.

4 Discussão e Trabalhos Futuros

Os resultados da compilação são promissores. A arquitetura modular se mostrou eficaz e o uso de recursos de hardware dedicados, como os blocos DSP, valida a escolha da plataforma FPGA para esta aplicação.

No entanto, o log de compilação apontou áreas para melhorias futuras:

- **Atribuição de Pinos:** O *Critical Warning* sobre a falta de atribuição de pinos deve ser resolvido para garantir a reprodutibilidade dos resultados de timing.
- **Latches Inferidos:** O *Warning* sobre a inferência de latches no módulo `DataReceiver` indica uma potencial melhoria no código VHDL para garantir uma lógica puramente síncrona e evitar problemas de timing.
- **Otimização de Performance:** A Fmax de [XX.XX MHz] é um bom ponto de partida. Técnicas de pipeline poderiam ser exploradas mais a fundo no multiplicador e no inversor de matrizes para aumentar ainda mais a frequência de operação e o throughput do sistema.
- **Utilização de memória** Uma análise crítica do sumário de recursos revela que a implementação não utilizou nenhum dos blocos de memória RAM embarcados (0% de utilização). Isso indica que o armazenamento das matrizes de dados foi inferido pelo sintetizador como um grande banco de registradores. Embora funcional, esta abordagem é sub-ótima em termos de eficiência

de área. Um refinamento crucial para trabalhos futuros seria modificar o código VHDL para forçar a inferência de RAMs, o que liberaria uma quantidade significativa de elementos lógicos e potencialmente melhoraria o desempenho geral do sistema.

5 Conclusão

Este trabalho demonstrou com sucesso a viabilidade da implementação de um acelerador de hardware para regressão linear em um FPGA. A análise qualitativa revelou uma arquitetura bem estruturada que utiliza eficientemente os recursos do dispositivo, especialmente os blocos DSP. Foram identificados os passos necessários para a correta análise de timing e propostas melhorias para refinar o projeto em trabalhos futuros. A implementação representa uma base sólida para a exploração de algoritmos de machine learning em hardware reconfigurável.