INF01113
Organização De Computadores B

Segundo Trabalho

Benchmarks de Organizações de Computadores

Bruno Alexandre Hofstetter Bourscheid (00550177)
Fernando Longhi de Andrade (00580366)
Luiz Augusto Ponzoni Schmidt (00580108)
Miguel Dutra Fontes Guerra (00342573)
Pedro Lubaszewski Lima (00341810)

Turma B

Enunciado do Problema

Objetivo

Projetar e descrever em VHDL um circuito que multiplique duas matrizes, some com uma terceira matriz e filtre a matriz resultado de acordo com a definição do cálculo abaixo. Além disso, comparar a construção manual com a solução HLS do Xilinx Vitis HLS.

Enunciado do Problema

Objetivo

Projetar e descrever em VHDL um circuito que multiplique duas matrizes, some com uma terceira matriz e filtre a matriz resultado de acordo com a definição do cálculo abaixo. Além disso, comparar a construção manual com a solução HLS do Xilinx Vitis HLS.

Função da Saída do Sistema (R_{2x2})

Dadas as matrizes A_{2x2}, B_{2x2} e C_{2x2} tais que a_{ij}, b_{ij}, $c_{ij} \in \mathbb{N}$ e $a_{ij} < 255$, $b_{ij} < 255$, $c_{ij} < 65535$, $\forall i, j \in \{1, 2\}$, e $F(M_{2x2}) = Q_{2x2}$ tal que

$$q_{ij} = \begin{cases} m_{ij} & \text{se } 0 < m_{ij} \leqslant 128 \\ 128 & \text{se } m_{ij} > 128 \end{cases}, \forall i, j \in \{1, 2\}, \text{ então}$$

$$R = F[(A \times B) + C]$$

Fluxograma ASM

Fluxograma ASM

Datapath

Datapath

FSM

FSM

Propostas de Testes

Para testar a solução, foram criadas as seguintes entradas:

- $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$;
- $B = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$;
- $B_{NULL} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$;
- $C = \begin{bmatrix} 121 & 10 \\ 50 & 3 \end{bmatrix}$.

Propostas de Testes

Para testar a solução, foram criadas as seguintes entradas:

- $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$;
- $B = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$;
- $B_{NULL} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$;
- $C = \begin{bmatrix} 121 & 10 \\ 50 & 3 \end{bmatrix}$.

Esperando as seguintes saídas:

- $R_1 = \begin{bmatrix} 128 & 15 \\ 70 & 16 \end{bmatrix}$, com A, B e C como entradas;
- $R_2 = \begin{bmatrix} 121 & 10 \\ 50 & 3 \end{bmatrix}$, com A, B_{NULL} e C como entradas.

Simulação para Validação de R_1

Simulação para Validação de R_1

Como a saída $doutr = [128, 15, 70, 16]$, no modo da memória *read first*, então foi validado que $R_1 = \begin{bmatrix} 128 & 15 \\ 70 & 16 \end{bmatrix}$.

Simulação para Validação de R_2

Simulação para Validação de R_2

Como a saída $doutr = [121, 10, 50, 3]$, no modo da memória *read first*, então foi validado que $R_2 = \begin{bmatrix} 121 & 10 \\ 50 & 3 \end{bmatrix}$.

Algoritmo em C

Algoritmo em C

Síntese da Solução Básica com HLS

Como solução básica, não se utilizou **nenhuma otimização**.

Síntese da Solução Básica com HLS

Como solução básica, não se utilizou **nenhuma otimização**.

Interface da Solução Básica com HLS

Interface da Solução Básica com HLS

Percebe-se um total de **12 IOBs**, totalizando **68 *bits*** de interface de dados.

Síntese da Primeira Solução Otimizada com HLS

Como primeira solução otimizada, utilizou-se `loop unroll` em cada um dos *loops* da aplicação, buscando aumentar o paralelismo da solução.

Síntese da Primeira Solução Otimizada com HLS

Como primeira solução otimizada, utilizou-se `loop unroll` em cada um dos *loops* da aplicação, buscando aumentar o paralelismo da solução.

Interface da Primeira Solução Otimizada com HLS

Interface da Primeira Solução Otimizada com HLS

Percebe-se um total de **16 IOBs**, totalizando **96 bits** de interface de dados.

Síntese da Segunda Solução Otimizada com HLS

Além das otimizações da versão anterior (`loop unroll`), acrescentou-se **`array partitioning`** com dimensão zero em todas as entradas e saídas do sistema. Isso foi feito em busca do maior nível de paralelismo possível para este circuito.

Síntese da Segunda Solução Otimizada com HLS

Além das otimizações da versão anterior (`loop unroll`), acrescentou-se **`array partitioning`** com dimensão zero em todas as entradas e saídas do sistema. Isso foi feito em busca do maior nível de paralelismo possível para este circuito.

Interface da Segunda Solução Otimizada com HLS

Interface da Segunda Solução Otimizada com HLS

Percebe-se um total de **16 IOBs**, totalizando **160 bits** de interface de dados.

Comparação do PC-PO com HLS

Ao construir a solução partindo do **Fluxograma ASM**, da **Parte Operativa** e da **Parte de Controle**, obteve-se um circuito com:

- 174 LUTs;
- 43 flip flops;
- 0 DSPs;
- 4 BRAMs;
- Latência de 76 ciclos de *clock*;
- Interface de dados com 4 IOBs, 40 bits.

Comparação do PC-PO com HLS

Ao construir a solução partindo do **Fluxograma ASM**, da **Parte Operativa** e da **Parte de Controle**, obteve-se um circuito com:

- 174 LUTs;
- 43 flip flops;
- 0 DSPs;
- 4 BRAMs;
- Latência de 76 ciclos de *clock*;
- Interface de dados com 4 IOBs, 40 bits.

Através da melhor solução com **HLS**, obteve-se os seguintes resultados:

- 405 LUTs;
- 36 flip flops;
- 4 DSPs;
- 0 BRAMs;
- Latência de 3 ciclos de *clock*;
- Interface de dados com 16 IOBs, 160 bits.

Comparação do PC-PO com HLS

Ao construir a solução partindo do **Fluxograma ASM**, da **Parte Operativa** e da **Parte de Controle**, obteve-se um circuito com:

- 174 LUTs;
- 43 flip flops;
- 0 DSPs;
- 4 BRAMs;
- Latência de 76 ciclos de *clock*;
- Interface de dados com 4 IOBs, 40 bits.

Através da melhor solução com **HLS**, obteve-se os seguintes resultados:

- 405 LUTs;
- 36 flip flops;
- 4 DSPs;
- 0 BRAMs;
- Latência de 3 ciclos de *clock*;
- Interface de dados com 16 IOBs, 160 bits.

De forma geral, a solução HLS tem um **paralelismo** melhor que a PC-PO. No entanto, a solução PC-PO apresenta uma **melhor utilização de recursos** (LUTs em particular) e uma **interface mais compacta**.

Obrigado pela atenção!
Alguma dúvida?