Somador Vetorial Próximo à Memória para SoCs: Projeto, Implementação e Avaliação em RTL e HLS

Eduardo Henrique Basilio de Carvalho
Renan Neves da Silva

Universidade Federal de Minas Gerais
Departamento de Engenharia Eletrônica

9 de agosto de 2024

UFMG

Sumário

1 Introdução
- Contexto
- Limitação
- Objetivo

2 Metodologia
- Referência
- RTL
- HLS
- Avaliação

3 Resultados

4 Discussão e trabalho futuro

5 Referências

UFMG

Introdução

Contexto

A Arquitetura de Memória em Sistemas Embarcados

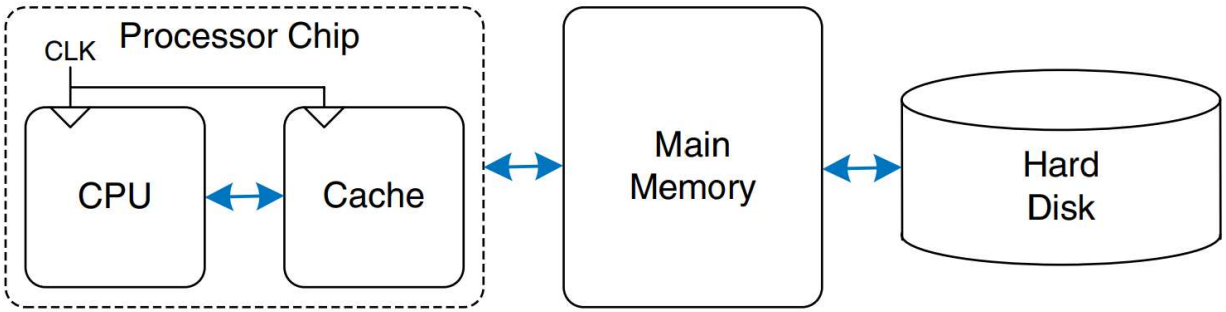

Figura: Exemplo de hierarquia de memória tradicional [1]

UFMG

Contexto

A Arquitetura de Memória em Sistemas **não** Embarcados

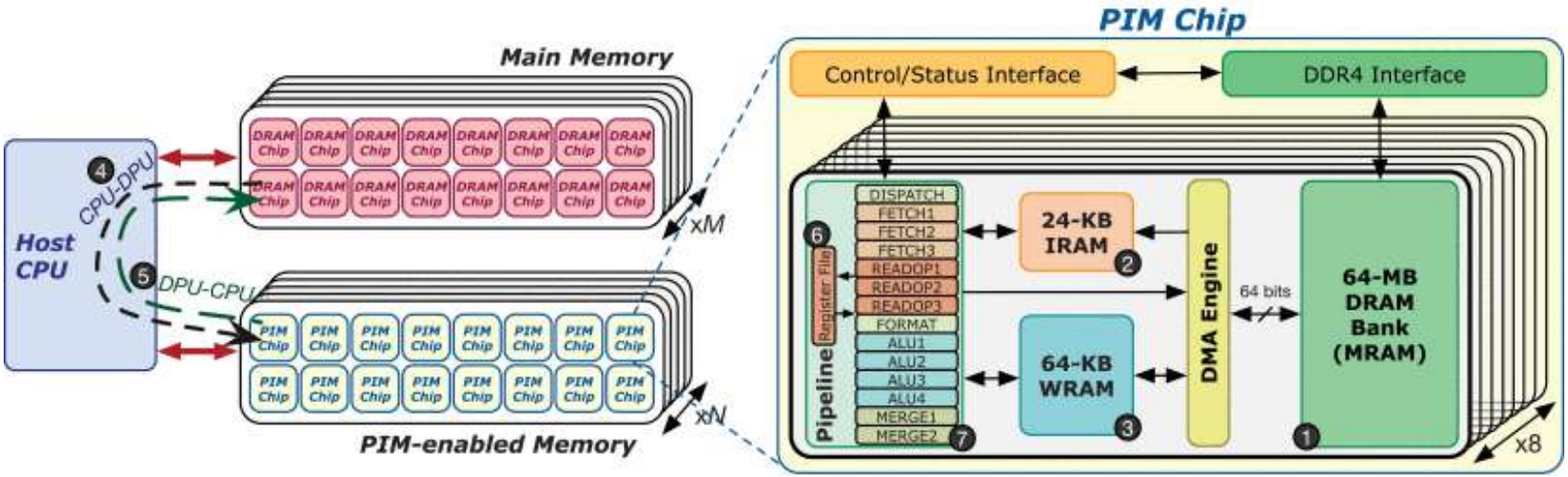

Figura: Exemplo de hierarquia de memória baseada em PIM [2]

UFMG

Limitação

Sistemas embarcados têm menos recursos

Card/SoC	U55C [3]	Z-7020 [4]
Max on-chip RAM	16 GB	4.9 MB
LUTs	1304 k	53.2 k
FFs	2607 k	106.4 k
DSPs	9024	220

UFMG

Objetivo

Implementar um somador em hardware próximo à memória capaz de reduzir a latência e o consumo de energia em soma de vetores.

Metodologia

Referência

Operação tomada como referência

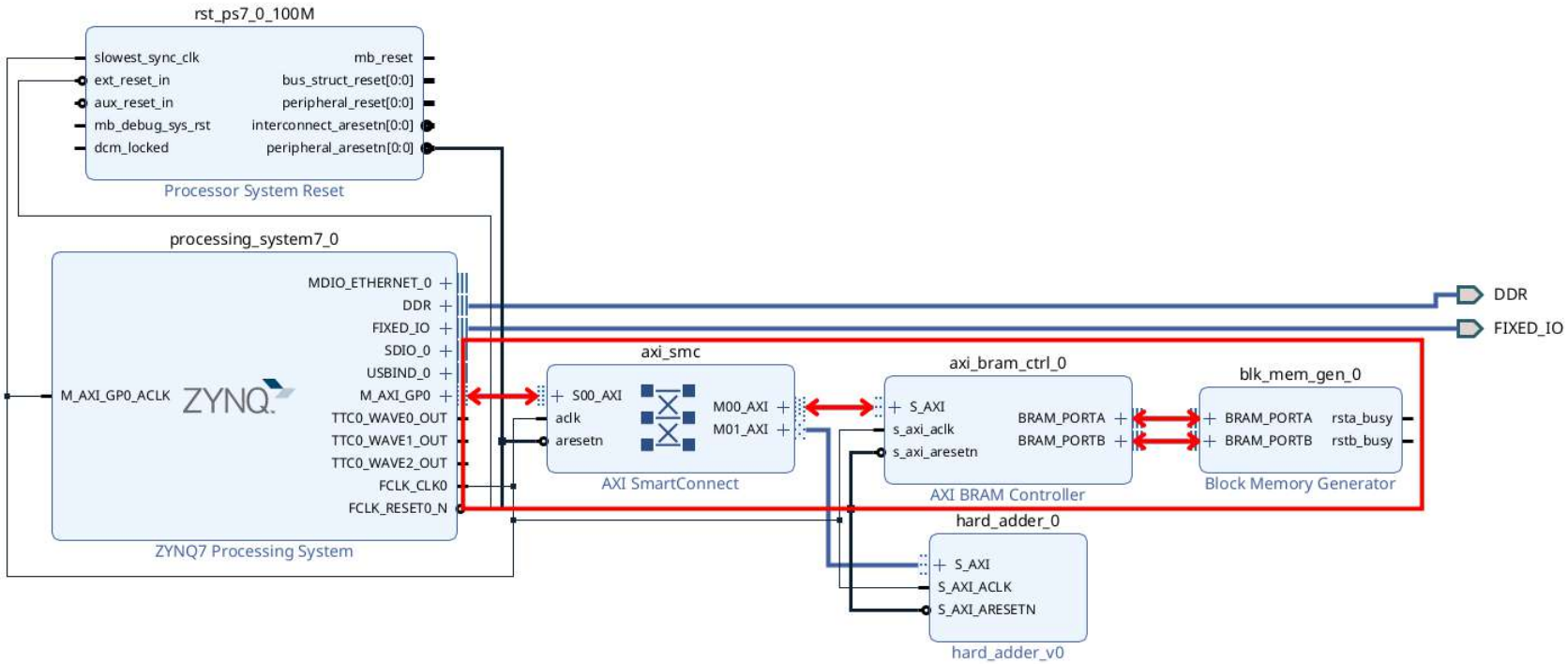

Figura: Diagrama de blocos do sistema, destacando o caminho padrão de dados

Referência

Parâmetros avaliados

- Latência: tempo entre início e fim de uma operação de soma;
- Uso de recursos: Recursos do PL empregados na operação.

UFMG

O IP

- AXI4-Lite;
- Envolve, com elementos lógicos, a BRAM sobre a qual opera;
- Opera assincronamente com o processador.

RTL

O IP

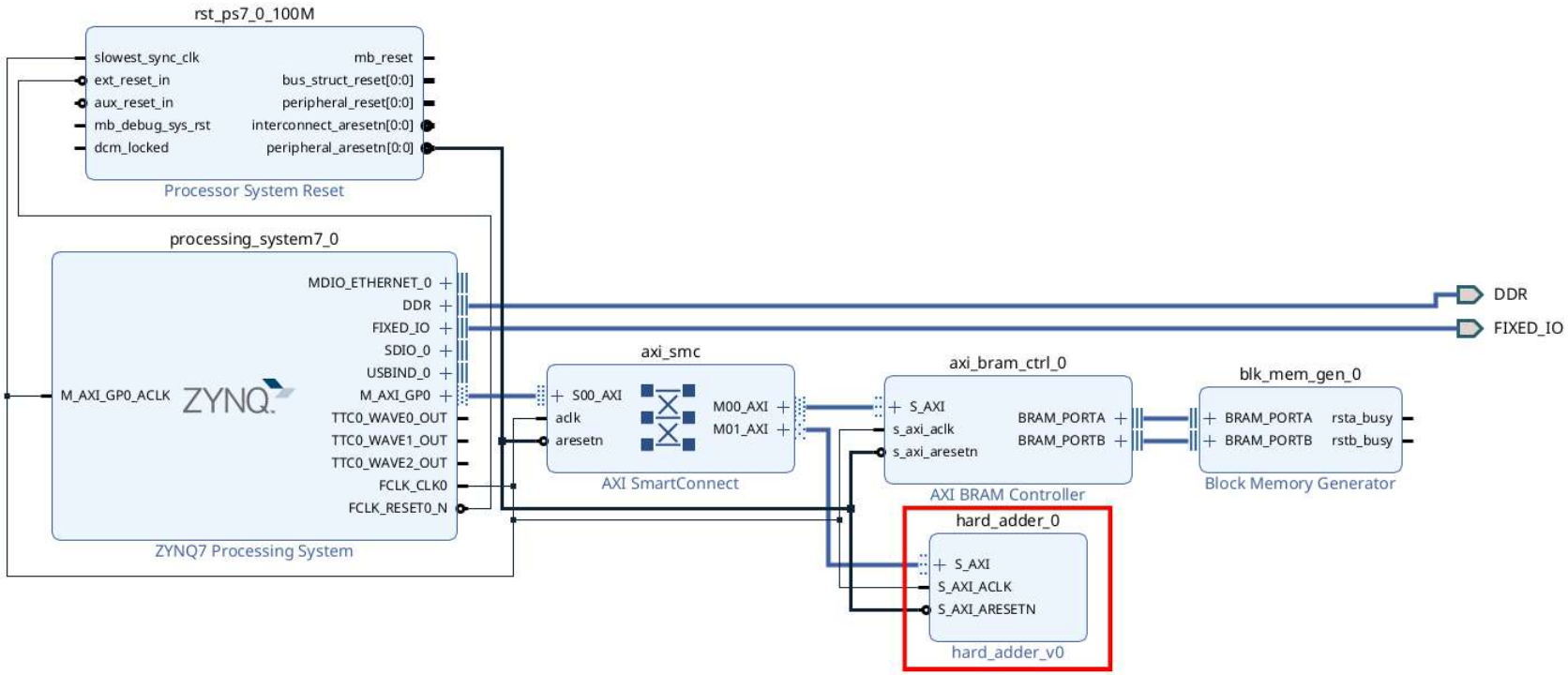

Figura: Diagrama de blocos do sistema, destacando o IP

RTL

O IP

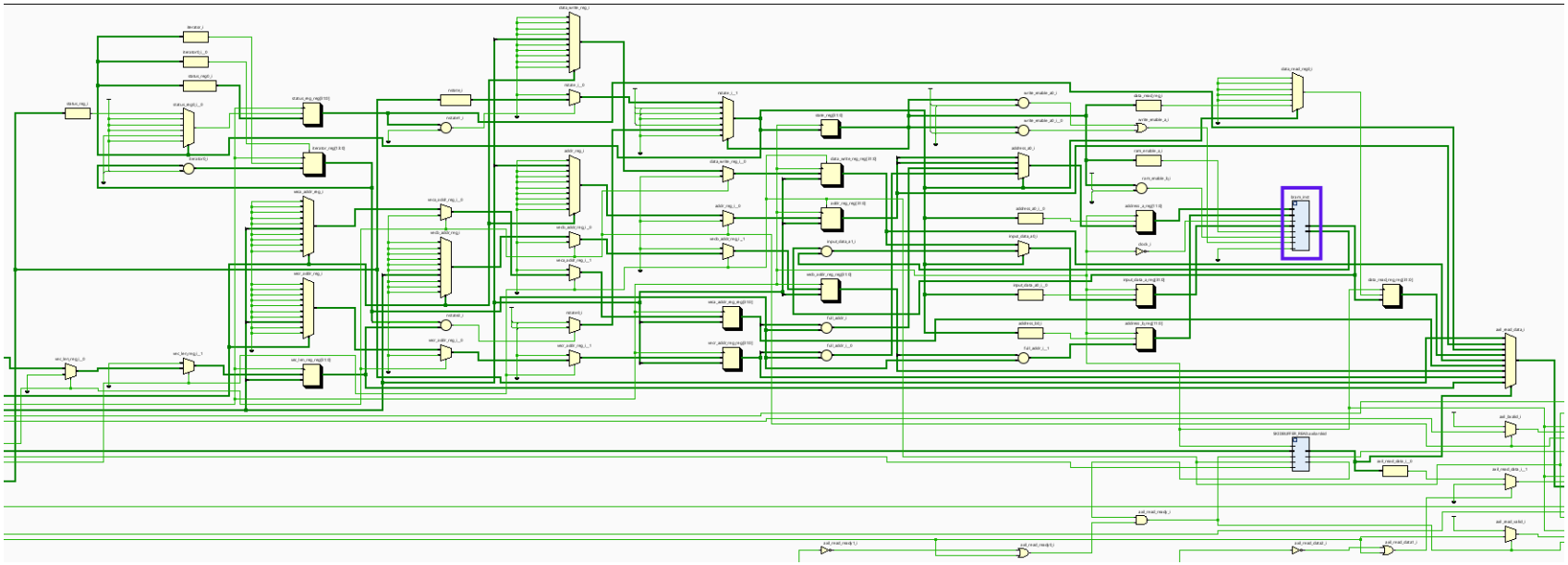

Figura: Esquemático parcial do IP, destacando a BRAM

A descrição em alto nível

```
if (!rst_n) {
    awready = false;
    wready = false;
    bvalid = false;
    arready = false;
    rvalid = false;
    compute_done = false;
    compute_idx = 0;
 } else {
    if (awvalid && !awready) {
        awready = true;
    } else {
        awready = false;
    }
    if (wvalid && !wready) {
        wready = true;
        if (awaddr[ADDR_WIDTH-1] == 0) {
            bram_a[awaddr.range(ADDR_WIDTH-2, 0)] = wdata;
        } else {
            bram_b[awaddr.range(ADDR_WIDTH-2, 0)] = wdata;
        }
    } else {
        wready = false;
    }
```

Código: Função em C++ a ser sintetizada

A descrição em alto nível

```
         if (wready && wvalid && !bvalid) {
            bvalid = true;
        } else if (bvalid && bready) {
            bvalid = false;
        }
        if (arvalid && !aready) {
            aready = true;
            if (compute_done) {
                rdata = bram_res[araddr.range(ADDR_WIDTH-2, 0)];
            } else {
                rdata = 0;
            }
        } else {
            aready = false;
        }
        if (aready && arvalid && !rvalid) {
            rvalid = true;
        } else if (rvalid && rready) {
            rvalid = false;
        }
        if (!compute_done) {
            bram_res[compute_idx] = bram_a[compute_idx] + bram_b[compute_idx];
            compute_idx++;
            if (compute_idx == MEM_SIZE) {
                compute_done = true;
            }
        }
    }
}
```

O IP

- AXI4-Lite;
- Cria três blocos de BRAM dentro do ip;
- Opera assincronamente com o processador com protocolo de handshaking.

HLS

O IP

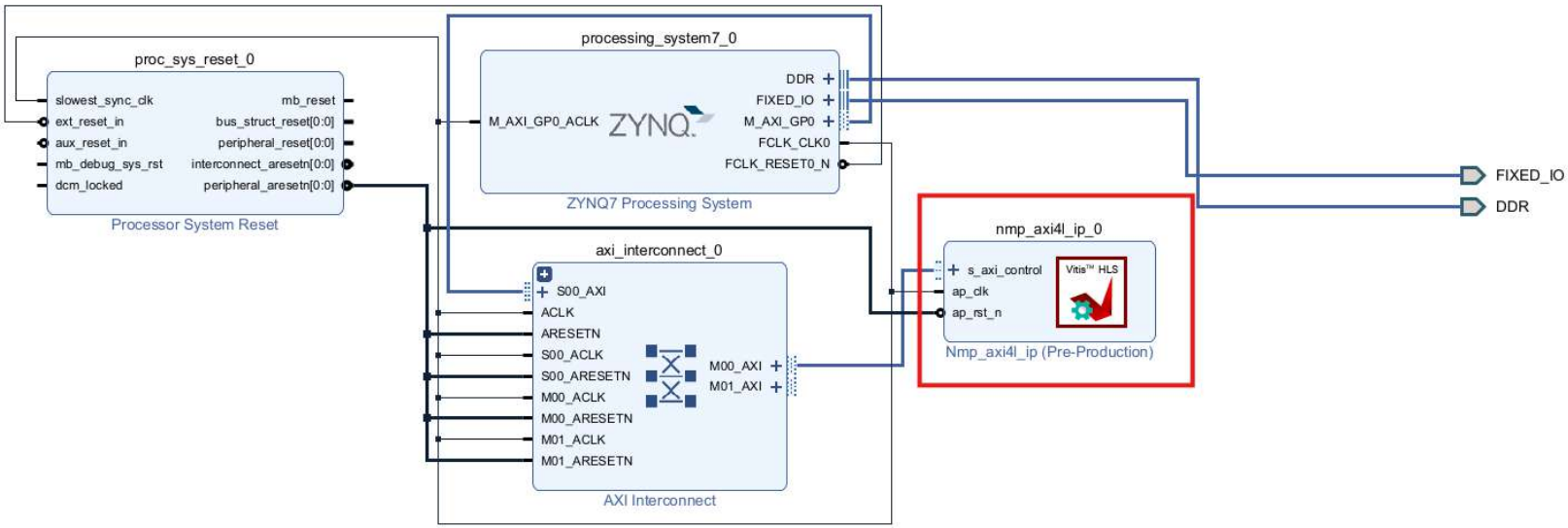

Figura: Diagrama de blocos do sistema, destacando o IP

HLS

O IP

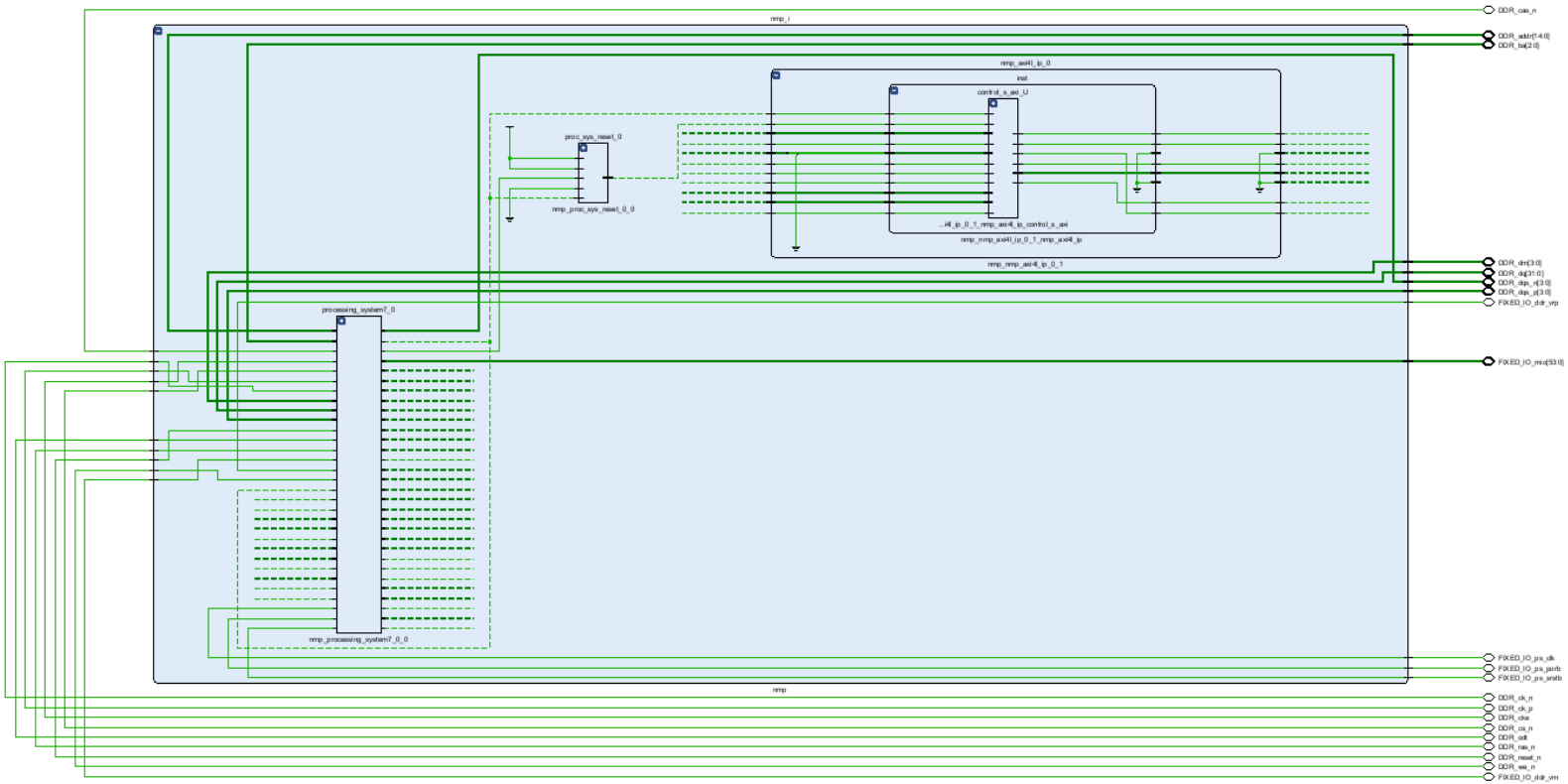

Figura: Esquematico simplificado do sistema

Validação do funcionamento

- *N* somas de vetores com posição, tamanho e conteúdo aleatórios;
- Resultados comparados com resultados da soma entre os mesmos valores na CPU.

Validação do funcionamento

```
unsigned random_tests() {
    for (int i = 0; i < test_qtty; i ++) {
        len   = rand();
        addra = rand() + offset_a;
        addrb = rand() + offset_b;
        addrr = rand() + offset_r;

        randomize_srcs_content_and_write_to_mem(addra, addrb, srca, srcb, len);
        compute_reference(srca, srcb, srcr, len);

        add_vectors(addra, addrb, addrr, len);

        for (u32 j = 0; j < len; j += 4 ) {
            data_read = read_from(addrr + j);

            if (!(data_read == srcr[j])) {
                error_count ++;
            }
        }
    }
    return error_count;
}
```

Código: Função executada no host para validar o somador

UFMG

Avaliação

Latência

Para avaliar a latência do sistema, foram feitas somas de vetores com 2^n palavras : $n = 2, 3, ..., 10$

- Soma em software de vetores armazenados na memória do PS (-O0);
- Soma em software de vetores armazenados na memória do PS (-O3);
- Soma em software de vetores armazenados na memória da PL;
- Soma em hardware de vetores armazenados na memória da PL.

UFMG

Avaliação

Latência

```
void timing_tests() {
    for (u32 i = 2; i < addr_bits; i ++) {
        len = 2 ** i;

        capture_start_t();
        add_vectors(addra, addrb, addrr, len);
        capture_end_t();

        time_diff = endt - startt;
        print(len, time_diff);
    }
}
```

Código: Função executada no host para capturar a latência do somador

Avaliação

Disparo do Acelerador

```
void nmp_add_vectors(XNMP nmp, u32 addra, u32 addrb, u32 addrr, u32 len) {
    Xil_Out32(nmp.base_addr + veca_addr_reg, addra);
    Xil_Out32(nmp.base_addr + vecb_addr_reg, addrb);
    Xil_Out32(nmp.base_addr + vecr_addr_reg, addrr);
    Xil_Out32(nmp.base_addr + vec_len_reg, len);
    Xil_Out32(nmp.base_addr + op_reg, add_op);

    nmp_wait_for_completion(nmp);
    nmp_set_wait(nmp);
}
```

Código: Função executada no host para disparar e aguardar o somador

UFMG

Recursos

Para avaliar o uso de recursos, o relatório de utilização da ferramenta Vivado foi considerado.

Resultados

Validação do funcionamento

Aprovado em todos os testes de funcionamento.

Latência

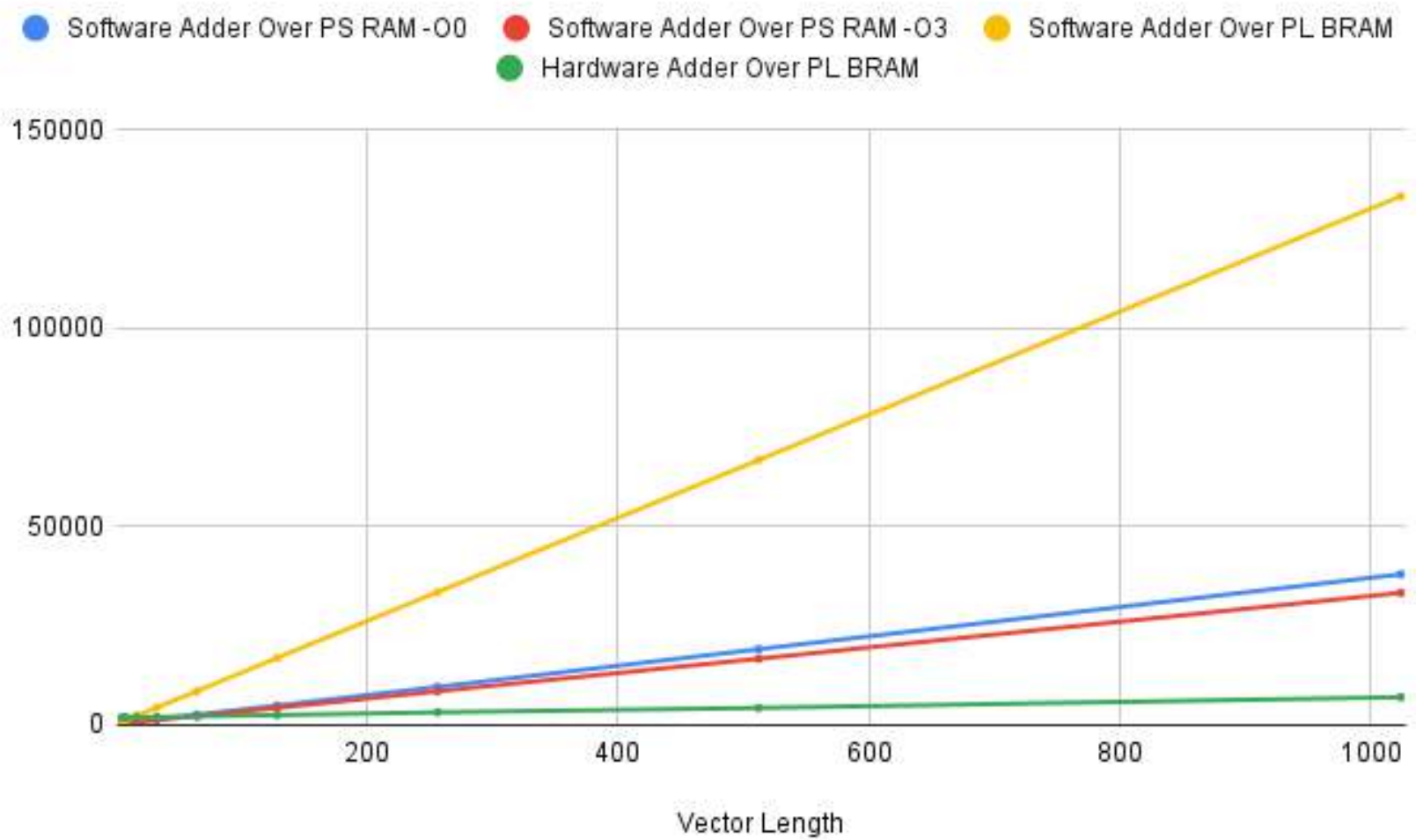

Figura: Latência [ns] x Tamanho [palavras] para as quatro operações avaliadas

Latência

Para todas as curvas, a descrição por $y = ax + b$ tem $R \approx 1$. Tomando esta, temos

Operação	1	2	3
a_i/a_0	26	7.4	6.4

tal que,
0: Soma em hardware de vetores armazenados na memória da PL;
1: Soma em software de vetores armazenados na memória da PL;
2: Soma em software de vetores armazenados na memória do PS (-O0);
3: Soma em software de vetores armazenados na memória do PS (-O3).

UFMG

Recursos

Resources *	Design Total	IP	% used for IP
Slice LUTs	980	318	32
Slice Registers	1192	430	36
Slice	442	147	33
LUT as Logic	974	318	33
LUT as Memory	6	0	0
Block RAM	6	4	67
BUFGCTRL	7	0	0

* O projeto atual do IP exige síntese e implementação altamente orientadas a atender os critérios de temporização, evitando a otimização da área.

UFMG

Discussão e trabalho futuro

Conclusões

O design proposto oferece escalabilidade superior às operações tradicionais, tendo aumento na latência por aumento no comprimento dos vetores inferior ao destas. Para vetores curtos, entretanto, as operações de preparação e disparo do acelerador levam a uma latência elevada. A troca entre consumo de recursos e redução de latência deve ser avaliada conforme aplicação, visto que esta não é desprezível nem impeditiva.

UFMG

Próximos passos

- Aprimorar validação:
  - mais métricas;
  - métricas mais relevantes;
  - foco em consumo de energia.
- Acelerar outras operações:
  - Processamento de grafos;
  - Armazenamento e processamento de matrizes esparsas.

UFMG

Referências

Referências


David Money Harris and Sarah L. Harris.
Digital Design and Computer Architecture.
Morgan Kaufmann, 1 edition, 2007.


Juan Gómez-Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu.
Benchmarking a new paradigm: Experimental analysis and characterization of a real processing-in-memory system.
IEEE Access, 10:52565–52608, 2022.


AMD.
Alveo u55c data center accelerator cards data sheet.
https://www.amd.com/pt/products/accelerators/alveo/u55c.html, 2021.


Xilinx.
Zynq-7000 soc data sheet.
https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf, 2021.

UFMG


Multumesc
Ua Tsaug Rau Koj
Dank Je
Chokrane
Kiitos
Mamana
Mochchakkeram
Maake
Kiitos
CHOKRANE
Juspaxar
Nirringrazzjak
Terma Kasih
Matondo
CHOKRANE
Merci
Kiitos
Thank You
Dankon
Obrigado
Asante
Obrigado
Obrigado
Salamat
Vinaka
Grazie
Maake
Terma Kasih
THANK YOU
Dank Je
Kiitos
Merci
OBRIGADO
Nirringrazzjak
Mochchakkeram
Multumesc
Obrigado
Ua Tsaug Rau Koj
Thank You
Chokrane
Maake
Matur Nuwun
Nirringrazzjak
Spasibo
Kiitos
Dank Je
Arigato
Welalin
Kiitos Multumesc
Spasibo
Cam on ban
Kiitos
Raibh Maith Agat
Merci
Chokrane
Grazie
Mochchakkeram
Juspaxar
Nirringrazzjak
Raibh Maith Agat
Obrigado
Maake
Mochchakkeram
Mochchakkeram
Asante
Obrigado
Salamat
Obrigado