**Relatório de Desenvolvimento da NPU**

O desenvolvimento da Neural Processing Unit (NPU), em Verilog, foi dividido na implementação dos módulos da NPU Core e os módulos Top Level, como exibido abaixo:

**Módulos NPU Core:**

1. Input Buffer
2. MAC Module
3. ReLu
4. PISO Out
5. PISO Deb
6. Auto Comparator
7. FIFO
8. MUX

**Módulos Top Level:**

1. NPU Core
2. FSM
3. SSFR
4. **Input Buffer**

O módulo Input Buffer foi desenvolvido com o objetivo de sincronizar e controlar o fluxo dos dados de entrada (DA, DB, DC e DD), cada um com largura de 8 bits. Para isso, são utilizados flip-flops do tipo D, que atuam como um buffer de entrada. O controle do armazenamento é realizado por meio dos sinais EN\_BUF\_IN (habilitação) e CLR\_BUF\_IN (limpeza). O clock externo (CLKEXT) é utilizado para sincronizar a operação dos registradores. Quando CLR\_BUF\_IN está em nível lógico baixo, todas as saídas (QA, QB, QC e QD) são zeradas; caso contrário, e , se EN\_BUF\_IN estiver ativo, os dados de entrada são transferidos para as respectivas saídas.

1. **MAC Module**

O módulo Multiply-and-Accumulate (MAC) consiste em um multiplicador de 8 bits, um somador de complemento de 2 de 16 bits com controle de saturação (2CASC), um multiplexador (MUX) e um flip-flop para armazenar o valor anterior, os quais foram implementados nesta ordem na construção do código.

Referente a adição, as entradas são estendidas com a duplicação do bit mais significativo (MSB), permitindo o controle de saturação, o qual possui a seguinte lógica: o resultado é 0x7FFF, em caso de estouro positivo; e 0x8000, em caso de estouro negativo.

O MUX seleciona o resultado do 2CASC, caso o valor de RST\_MAC seja zero, caso contrário, é selecionado um valor de bias pré-carregado. Esse valor é armazenado em um registrador implementado com flip-flop D, o qual é sincronizado com o clock externo (CLKEXT) e sua operação é habilitada pelo sinal EN\_MAC.

1. **ReLu**

O módulo ReLu implementa a função Rectified Linear Unit com opção de bypass, sua principal característica é zerar valores negativos caso o bypass não esteja ativo . Ele possui as seguintes entradas: Data\_Reg[15:0]: entrada de dados (16 bits); En\_ReLU: enable do ReLU; En\_MAC\_ReLU: enable do MAC para ReLU; BYPASS\_ReLU: bypass do módulo; RST\_ReLU: reset assíncrono (ativo em alto); CLK: clock do sistema. E uma única saída: ReLU\_OUT[15:0]: resultado do ReLU (16 bits).

O módulo implementa a seguinte lógica:

Se RST\_ReLU = 1: ReLU\_OUT = 0x0000

Se BYPASS\_ReLU = 1: ReLU\_OUT = Data\_Reg

Se BYPASS\_ReLU = 0: ReLU\_OUT = (Data\_Reg < 0) ? 0 : Data\_Reg

1. **PISO OUT**

O módulo Parallel-Input Serial-Output (PISO) Out possui a função de converter as duas saídas paralelas dos módulos MAC, de 16 bits cada, em quatro blocos sequenciais de 8 bits compatíveis com a porta de saída D\_OUT.

Para isso, utiliza-se um registrador interno de 32 bits que, quando habilitado por EN\_PISO\_OUT, pode operar em dois modos a depende do valor da variável SHIFT\_OUT. Se ela for 0, os dados de mac0\_out e mac1\_out são armazenados em shift\_reg. Caso contrário, ocorre o deslocamento serial, assim, os 8 bits mais significativos são enviados para D\_OUT e o registrador é deslocado à esquerda a cada ciclo de clock, controlado por CLKEXT.

Além disso, o sinal CLR\_PISO\_OUT permite a limpeza do registrador e da saída, garantindo inicialização e reinício corretos no processo de conversão.

1. **PISO DEB**

O módulo PISO DEB também realiza uma conversão paralelo para serial, mas ele foi construído com a finalidade de ser um sistema de depuração. Ele recebe os sinais internos (SSFR, CON\_SIG, MAC1, MAC2, QA, QB, QC e QD) e os transmite em sequência através da porta de saída D\_OUT.

Esses dados são armazenados em um vetor interno (dbg\_bytes) e controlados pelos sinais EN\_PISO\_DEB (habilita a transmissão), CLR\_PISO\_DEB (zera os registradores) e SHIFT\_DEB (seleciona o modo de operação: 0 para captura paralela e 1 para transmissão serial). Durante o funcionamento, o módulo armazena os sinais internos e, em seguida, os disponibiliza byte a byte a cada ciclo de clock.

1. **Auto Comparator**

O módulo Auto Comparator compara duas entradas de 16 bits e retorna o maior valor entre elas. Para operar, ele trabalha com as seguintes entradas: In\_Read[15:0]: primeira entrada para comparação; In\_COMP[15:0]: segunda entrada para comparação; RST\_COMP: reset assíncrono (ativo em alto); EN\_COMP: enable do comparador; e CLK: clock do sistema. Por fim, o resultado da comparação é armazenado na variável Output.

Abaixo está descrito sua lógica funcional:

Se RST\_COMP == 1 → saída é zerada imediatamente (reset assíncrono).

Senão, na borda de subida do clock:

Se EN\_COMP == 1 → saída recebe o maior valor entre In\_Read e In\_COMP.

Se EN\_COMP == 0 → saída recebe 0x0000.

1. **FIFO**

O módulo FIFO tem o objetivo de armazenar os resultados do PISO\_OUT para que o usuário possa ler depois, sem perder dados após um ciclo. Ele segue o modelo de projeto síncrono, ou seja, possui um único clock para leitura e escrita e ambas as operações podem ocorrer no mesmo ciclo.

O módulo possui os seguintes sinais de controle: EN\_FIFO: habilita o funcionamento do FIFO (codificado em SSFR); RST\_FIFO: reseta ponteiros de leitura e escrita (codificado em SSFR); WR\_EN: habilita escrita: gerado apenas pelo FSM (modo automático); RD\_EN: habilita leitura: acessível ao sistema mestre.

Além disso, há regras de intertravamento, como descrito abaixo:

* Nunca escrever com FULL = 1;
* Nunca ler com EMPTY = 1;
* Segurança garantida pelo bloqueio automático via FULL/EMPTY.

1. **MUX**

O módulo MUX é um bloco de saída que conecta os diferentes módulos funcionais da NPU a um único barramento de saída. Ele implementa um multiplexador de 5 entradas de 8 bits que seleciona qual dado será enviado para a saída D\_OUT, de acordo com o valor do sinal de controle SEL\_OUT. Sua operação está descrita a seguir:

Entrada de dados:

* fifo\_data: saída do módulo FIFO (memória de fila)
* piso\_data: saída do módulo PISO (Parallel-In Serial-Out)
* cmp\_data: saída do comparador
* relu\_data: saída da função de ativação ReLU
* mac\_data: saída da unidade MAC (Multiply and Accumulate)

Sinal de seleção:

* 000: seleciona fifo\_data
* 001: seleciona piso\_data
* 010: seleciona cmp\_data
* 011: seleciona relu\_data
* 100: seleciona mac\_data
* outros valores recebem 0x00 (valor nulo)

1. **NPU Core**
2. **FSM**

O módulo FSM foi desenvolvido para atuar como unidade de controle principal do sistema, coordenando a operação sequencial entre os blocos Input Buffer, MAC, ReLU, FIFO e PISO. Sua implementação foi realizada por meio de uma máquina de estados finita (FSM) síncrona ao clock externo (CLKEXT), utilizando registradores do tipo D para armazenar o estado atual e a contagem de ciclos.

Seu funcionamento é dividido em sete estados: IDLE, LOAD\_INPUT, COMPUTE, RELU\_STAGE, WRITE\_FIFO, OUTPUT\_SHIFT e FINISH. No estado LOAD\_INPUT, os dados de entrada são carregados para o Input Buffer. Em COMPUTE, o módulo MAC é habilitado para realizar operações de multiplicação e acumulação durante um número configurado de ciclos. Em RELU\_STAGE, a saída do MAC passa pelo ReLU, que aplica a função de ativação. Em WRITE\_FIFO, os resultados processados são armazenados no FIFO em formato de bytes. Posteriormente, em OUTPUT\_SHIFT, o módulo PISO é habilitado para converter os resultados em série, disponibilizando-os na saída D\_OUT. Finalmente, no estado FINISH, o processamento é concluído e o sistema retorna ao estado IDLE.

1. **SSFR**