A brief introduction to Convolution Module in the Project

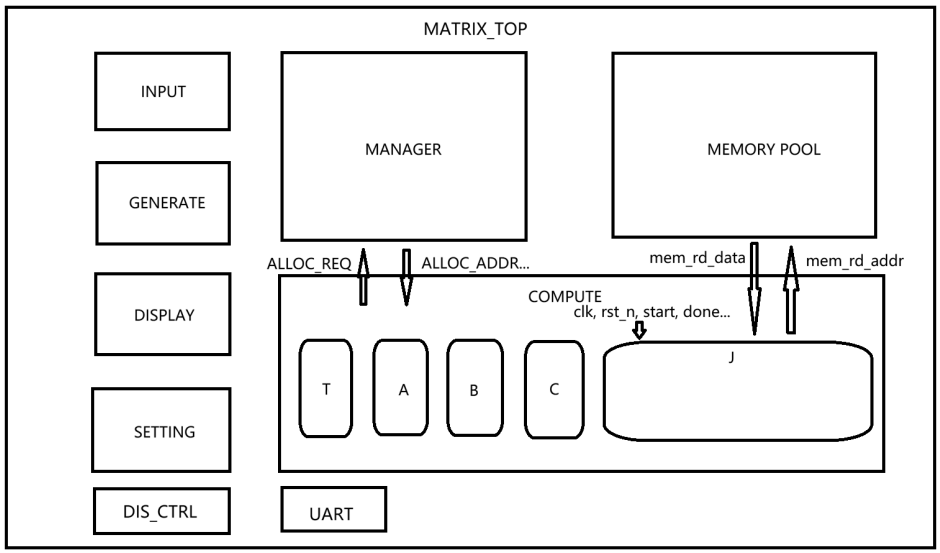
Yanqiao Chen

Department of Computer Science and Engineering, SUSTech

2025.12.21

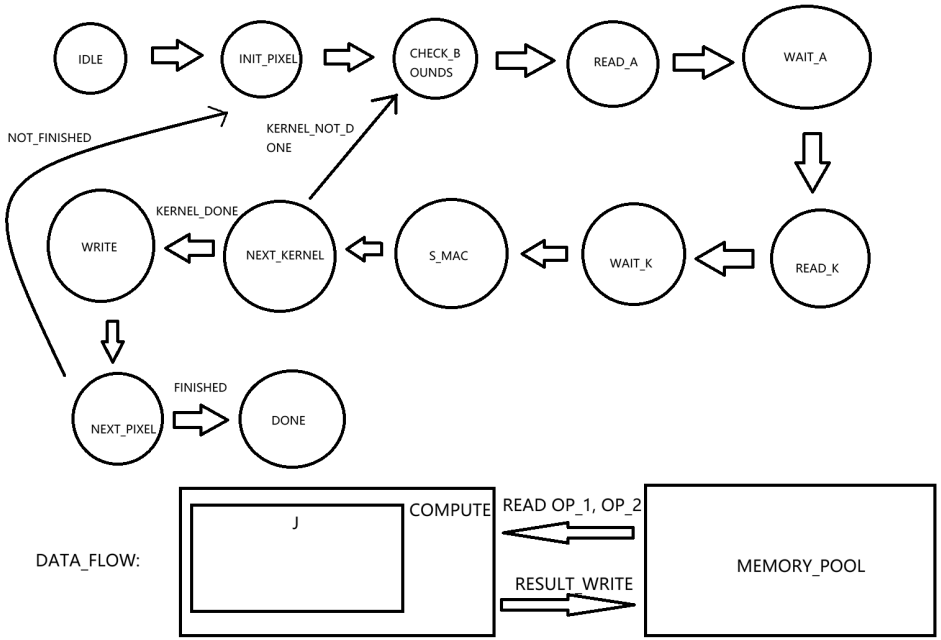
卷积模块位置和接口

卷积模块位于计算模块（Compute Mode）的下一级，负责对输入的数据进行卷积操作。接口包含：clk, rst_n, start, done，dim_m, dim_n, addr_op1, addr_op2, addr_res（来自 Compute Mode），mem_rd_data（来自 Memory Pool），mem_rd_en, mem_rd_addr（读使能与读地址），mem_wr_en, mem_wr_addr, mem_wr_data（写使能、写地址与写数据）。（图中只展示了与卷积模块相关的接口，详见项目文档）



卷积模块的设计与数据流

卷积模块设计为一个有限状态机（FSM），包含 IDLE、INIT_PIXEL、CHECK_BOUND、READ_A、WAIT_A、READ_K、WAIT_K、S_MAC、WRITE、NEXT_PIXEL、DONE 等状态。计算流程为，对于每一个位置，进入卷积核内部的状态循环，累加结果然后写入内存，然后继续计算下一个位置的卷积结果，直到到达矩阵最右下角。
其中，我们通过计算模式用户选择的操作数 1 和操作数 2 的地址为起始，通过内部的偏移量计算出每次需要读取的数据的地址，从 Memory Pool 中读取数据进行计算，最后将结果写回 Memory Pool。



卷积模块的设计与数据流

具体来说，数据流的设计如下：

- 卷积模块读取计算模块所给予的两个操作数的起始地址，分别对应输入矩阵和卷积核矩阵。
- 通过 i, j 两个变量控制读取的位置，通过这两个变量计算块内的偏移，从而计算出每次需要读取的数据的地址。例如，当中心为 $i = 1, j = 1$ 的时候，我们需要计算一个偏移地址。例如，当我们在卷积核最左上角的位置的时候，我们需要读取 $i = 0, j = 0$ 位置的数据进行计算。也就是说，若卷积核内循环变量为 $0 \leq ki, kj \leq 2$，则需要读取的位置为 $(i + ki - 1, j + kj - 1)$。
- 读取的数据通过 mem_rd_data 接口传入卷积模块，进行乘加操作。通过累加器 acc 保存中间结果，然后在 WRITE 状态中通过 mem_wr_data 接口将结果写回 Memory Pool。

理论周期数分析

根据所给矩阵大小 10×12 和卷积核大小 3×3，计算出输出矩阵大小为 8×10。每个输出位置需要进行

$$3 \times 3 = 9$$

次乘加操作，单个位置所需要的周期数为：计算偏移 1 周期，读取矩阵 2 周期，读取卷积核 2 周期，累加 1 周期，因此单个位置总共需要 6 周期。总共循环 9 次，计算得到单个位置需要 54 周期。其中，每次 INIT 需要消耗 1 周期，总共 80 周期，WRITE 需要 1 周期，总共 80 周期，NEXT_PIXEL 需要 1 周期，总共 80 周期。（done 和 start 共两周期，忽略不计）因此计算得到：

$$54 \times 80 + 80 + 80 + 80 = 4560$$

周期。

实际测试时钟周期数

实际经过测试，我们的周期数为 5280 周期，高于理论周期数 4560 周期。这可能是 Vivado 的综合实现导致的额外开销，也有可能是在等待数据的过程中产生的额外周期数。

```
148 142 119 153 149 140 154 150 130 115
139 143 159 137 187 145 127 80 168 155
147 108 107 188 187 146 121 114 158 158
154 143 128 136 178 160 98 135 146 152
160 89 159 168 197 148 123 126 123 131
152 156 141 145 139 192 172 96 138 109
137 156 168 145 127 189 161 117 89 151
176 164 190 108 155 156 178 137 89 87
5280
```

其中最后一行为时钟周期数。

资源使用报告

我们的综合资源使用报告如下：

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)
matrix_calculator_top_opt...	4102	2037	45	11	1382	4102	685	1
bram_mem_inst (bra...	111	0	0	0	50	111	0	1
compute_mode_inst (...	1740	579	19	0	583	1740	334	0
op_add_inst (matrix...	88	65	0	0	39	88	43	0
op_conv_inst (matri...	210	107	0	0	91	210	64	0
op_mul_inst (matrix...	193	70	0	0	80	193	44	0
op_smul_inst (matr...	103	48	0	0	37	103	45	0
op_trans_inst (matr...	142	48	0	0	54	142	38	0
db_back (button_debo...	15	25	0	0	12	15	12	0
db_confirm (button_de...	15	25	0	0	12	15	12	0
disp_ctrl_inst (display...	11	22	0	0	16	11	3	0
display_mode_inst (di...	55	54	0	0	27	55	17	0
generate_mode_inst (...	221	155	0	0	96	221	67	0
input_mode_inst (inpu...	375	148	0	0	159	375	79	0
matrix_mgr_inst (matri...	1225	849	22	11	490	1225	18	0
rng_inst (lfsr_rng)	3	16	0	0	7	3	1	0
setting_mode_inst (se...	166	63	4	0	62	166	41	0
uart_inst (uart_module)	150	65	0	0	82	150	34	0

由上可见，我们的卷积模块使用 LUT 共 210 个，slice Register 共 107 个，LUT 主要用于逻辑实现。

遇到的问题和解决方案

实际开发过程中，我们会遇到一些问题，例如如何存储计算结果、如何读取数据等。针对这些问题，我们的核心解决方案就是将结果写入内存，以方便后续的展示。我们通过地址-数据映射的方式，通过起始地址-块内偏移的方式，计算出每一次需要读取数据和写入数据的位置，以免读取数据发生错读和错写的问题。

同时，我们在实际开发过程中，在 COMPUTE MODE 中为卷积模块定制了一个计算流程：首先选择需要卷积的矩阵，然后自动给出所有的 3×3 卷积核，随后开始计算。但是在开发过程中，由于计算模式的状态接近 100 个，我们会遇到由于数位截断导致状态重合的问题（这是由于早期开发状态的位宽较小的原因，例如 5‘b100111 和 5’b000111 重合）。我们通过仔细检查状态机的状态定义，增加状态位宽，最终解决了该问题。