RESEARCH PROJECT

UNDER DR. SPARSH MITTAL, IITR

Image Processing in MATLAB

- We have used this cameraman grayscale image of size 256x256
- This image is used as reference for filtering with different filters and data-types(SP, HP, Double)
- Approximation on values of kernel and image matrix done using commands in MATLAB

Image Processing in MATLAB

- This is the resultant image , kernel and original image are approximated using CHOP() in MATLAB
- chop(X, n, unit) rounds the elements of X to n significant figures whose digits (mantissa) are exactly divisible by unit.
- Here X is the input matrix (image), n=1, unit=5

Image Processing in MATLAB

- This is the output when n=1, unit=4
- In both the results , the output image size is 252x252, and the datatype is single precision.
- Half precion shows minor decrement in quality.

Image Processing using OpenCV

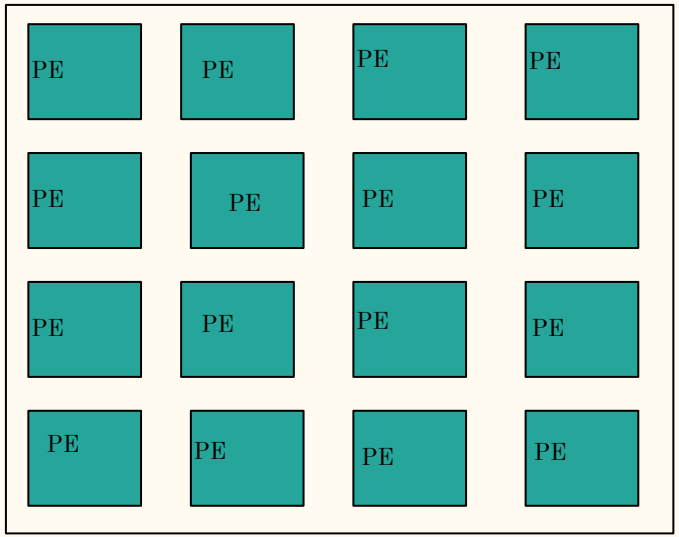
- This filtering is done using 7x7 kernel.
- Datatype is single precision, HP shows minor decrement in quality.

Image Processing using OpenCV

- This filtering is done using 5x5 kernel.
- This has been done using single-precision, half precision shows minor decrement in quality.
- Convolutions have been performed using direct commands in OpenCV, GeMM based convolution also performed in MATLAB.

Verilog designs of Systolic array

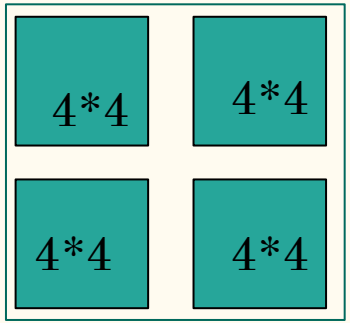
- We created Verilog designs for 4*4, 8*8 and 16*16 systolic arrays along with their testbenches to understand their working and synthesize them on tools like Yosys and Synopsys in order to evaluate and compare their area, power consumption and latency of the designs.
- We began with the basic building block of a processing element in a module, then used it to form a 4*4 array of PEs and thus implement the systolic array.
- In 8*8 array designs, we instantiated the existing 4*4 systolic array 4 times and created its design. Similarly, we created the 16*16 systolic array.



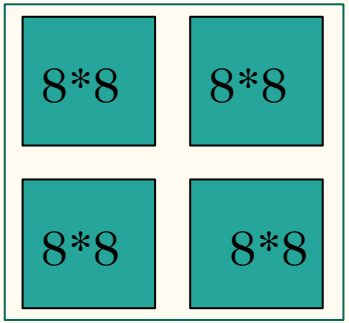

```verilog
module block(inp_north, inp_west, clk, rst, outp_south, outp_east, result);
    input [31:0] inp_north, inp_west;
    output reg [31:0] outp_south, outp_east;
    input clk, rst;
    output reg [63:0] result;
    wire [63:0] multi;
    always @(posedge rst or posedge clk) begin
        if(rst) begin
            result <= 0;
            outp_east <= 0;
            outp_south <= 0;
        end
        else begin
            result <= result + multi;
            outp_east <= inp_west;
            outp_south <= inp_north;
        end
    end
    assign multi = inp_north*inp_west;
endmodule
```

4*4 SYSTOLIC ARRAY DESIGN WITH CODE OF A PE

DESIGN IDEA FOR HIGHER ARRAYS IN VERILOG



8*8 DESIGN IDEA



16*16 DESIGN IDEA

Area Calculations using Yosys

```
2.24.5. Finished fast OPT passes.

2.25. Executing HIERARCHY pass (managing design hierarchy).

2.25.1. Analyzing design hierarchy..
Top module:  \systolic_array

2.25.2. Analyzing design hierarchy..
Top module:  \systolic_array
Removing unused module '\block'.
Removed 1 unused modules.

2.26. Printing statistics.

=== systolic_array ===

   Number of wires:                 22
   Number of wire bits:            276
   Number of public wires:          12
   Number of public wire bits:     263
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 18
     $_ANDNOT_                       5
     $_DFF_PP0_                      5
     $_MUX_                          1
     $_NAND_                         1
     $_NOT_                          1
     $_ORNOT_                        3
     $_XOR_                          2

2.27. Executing CHECK pass (checking for obvious problems).
checking module systolic_array..
found and reported 0 problems.

yosys>
```

- We formed the structural netlist of the created verilog designs using Yosys and Apio and estimated the area using size of standard cells.

- The shown results are for 4*4 systolic array, providing information on the physical statistics of design.

Statistics for 8*8 and 16*16 systolic array

```
Finding unused cells or wires in module \block..
Finding unused cells or wires in module \sysarray8..
Finding unused cells or wires in module \systolic_array..
Removed 0 unused cells and 608 unused wires.
<suppressed ~2 debug messages>

2.24.5. Finished fast OPT passes.

2.25. Executing HIERARCHY pass (managing design hierarchy).

2.25.1. Analyzing design hierarchy..
Top module:  \sysarray8

2.25.2. Analyzing design hierarchy..
Top module:  \sysarray8
Removing unused module '\block'.
Removing unused module '\systolic_array'.
Removed 2 unused modules.

2.26. Printing statistics.

=== sysarray8 ===

   Number of wires:                 23
   Number of wire bits:            519
   Number of public wires:          23
   Number of public wire bits:     519
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  0

2.27. Executing CHECK pass (checking for obvious problems).
checking module sysarray8..
found and reported 0 problems.

yosys>
```

```
Finding unused cells or wires in module \sysarray8..
Finding unused cells or wires in module \systolic_array..
Removed 0 unused cells and 608 unused wires.
<suppressed ~2 debug messages>

2.24.5. Finished fast OPT passes.

2.25. Executing HIERARCHY pass (managing design hierarchy).

2.25.1. Analyzing design hierarchy..
Top module:  \sysarray16

2.25.2. Analyzing design hierarchy..
Top module:  \sysarray16
Removing unused module '\block'.
Removing unused module '\sysarray8'.
Removing unused module '\systolic_array'.
Removed 3 unused modules.

2.26. Printing statistics.

=== sysarray16 ===

   Number of wires:                 39
   Number of wire bits:           1031
   Number of public wires:          39
   Number of public wire bits:    1031
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  0

2.27. Executing CHECK pass (checking for obvious problems).
checking module sysarray16..
found and reported 0 problems.

yosys>
```

What's next?

- We will further calculate the power consumption and latency of these designs using tools like Synopsys design compiler.
- We aim to optimize the current verilog designs in terms of area, power and latency
- We will further extend the array upto 128*128 and then look at the trends for the same.
- We may also test it for sparse and pruned matrices, and even replace the exact MAC operation in a PE by approximate operation and then check the trends.

Thank you!