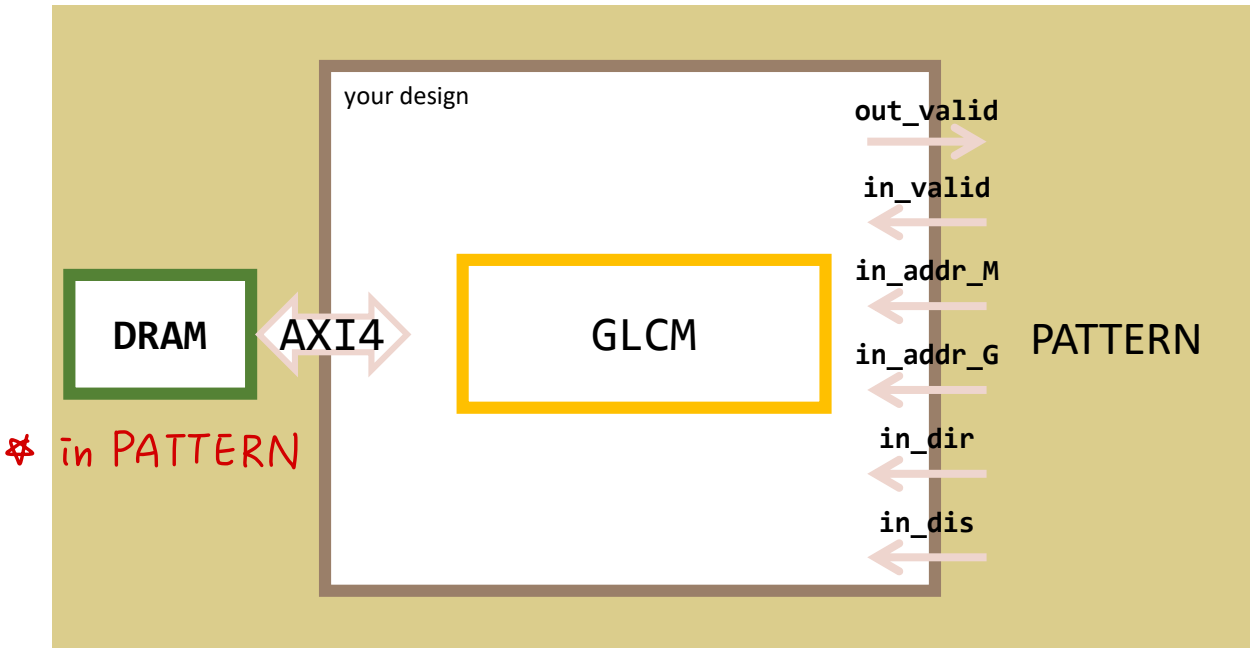
Midterm project 2023 spring

ICLAB

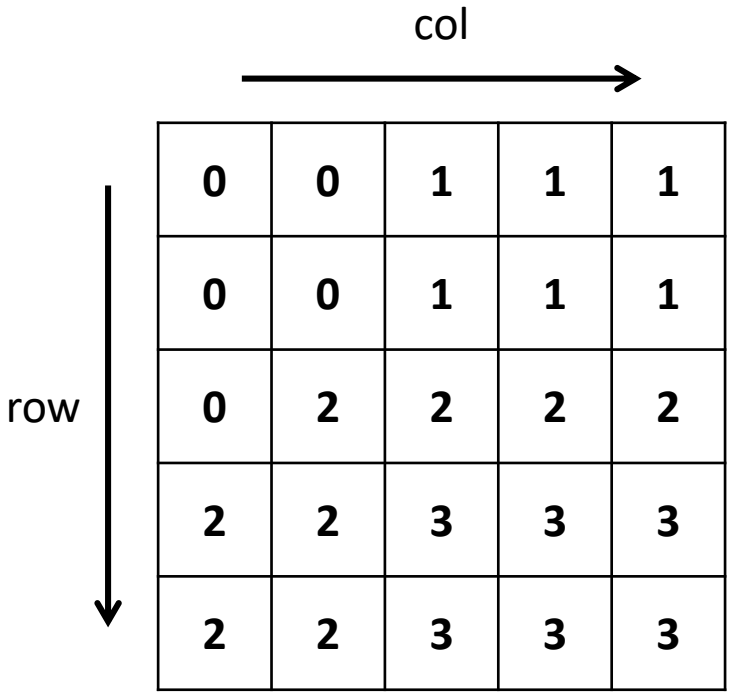
TA: Hsi-Hao Huang

Overall system block



- PATTERN will communicate with design through IO ports.
- You will get the address of DRAM from the pattern. And you need to use it to get the data of input matrix from the DRAM.
  - For Input matrix: address range: 1000~1fff
  - For GLCM matrix: address range: 2000~2fff
- After you completed the calculation, the result should be written back to DRAM at the address of GLCM.
- When out_valid is high, pattern will check check the GLCM value in DRAM.

Input matrix:



Offset:

(0,1)

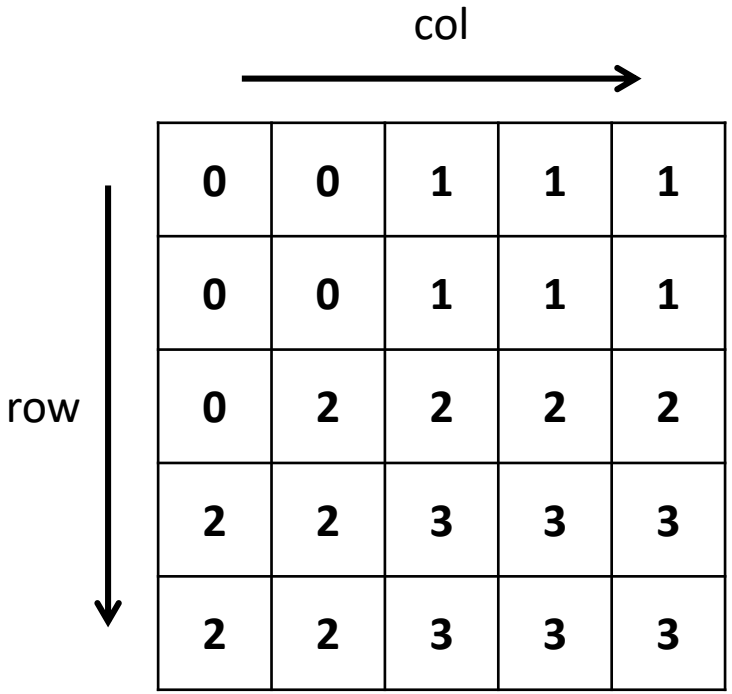
offset 後的救值

GLCM matrix:

原始救值

	C. 0	C. 1	C. 2	C. 3
Ref. 0	2	2	1	0
Ref. 1	0	4	0	0
Ref. 2	0	0	5	2
Ref. 3	0	0	0	4

Input matrix:



0	0	1	1	1
0	0	1	1	1
0	2	2	2	2
2	2	3	3	3
2	2	3	3	3

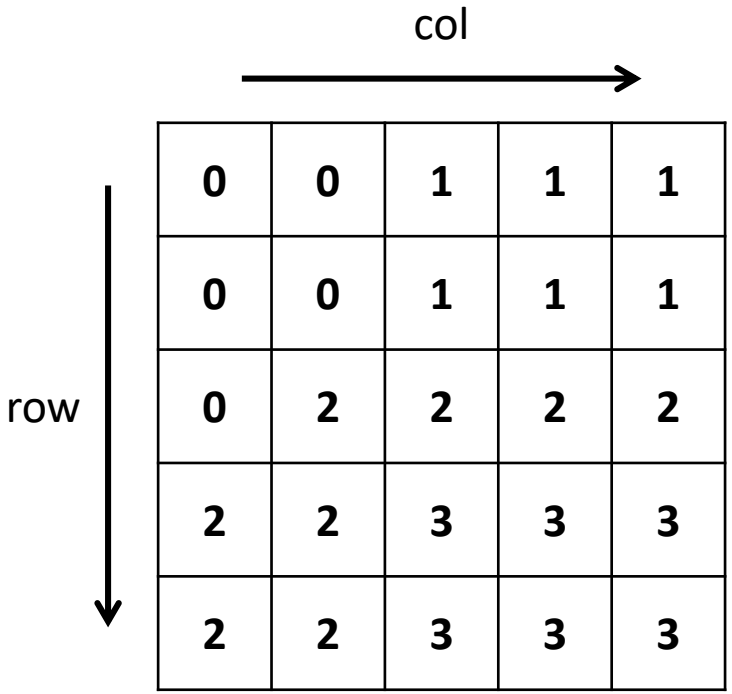
Offset:

(1,0)

GLCM matrix:

	C. 0	C. 1	C. 2	C. 3
Ref. 0	3	0	2	0
Ref. 1	0	3	3	0
Ref. 2	0	0	3	3
Ref. 3	0	0	0	3

Input matrix:



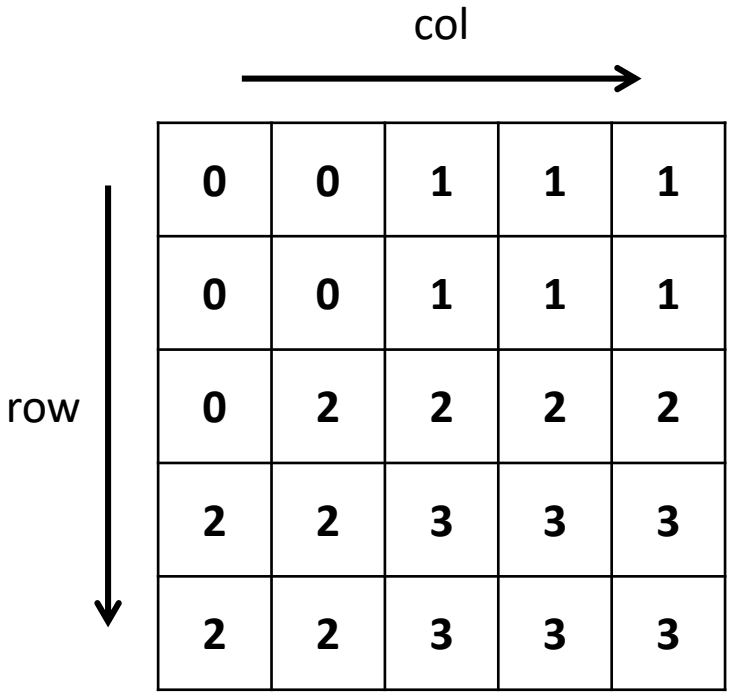
Offset:

(1,1)

GLCM matrix:

	C. 0	C. 1	C. 2	C. 3
Ref. 0	1	1	3	0
Ref. 1	0	2	2	0
Ref. 2	0	0	1	4
Ref. 3	0	0	0	2

Input matrix:



0	0	1	1	1
0	0	1	1	1
0	2	2	2	2
2	2	3	3	3
2	2	3	3	3

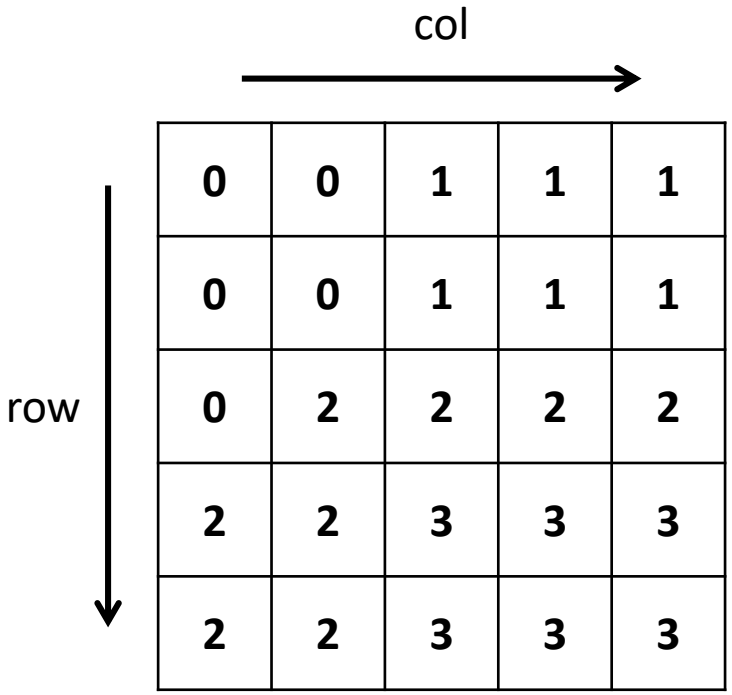
Offset:

(2,2)

GLCM matrix:

	C. 0	C. 1	C. 2	C. 3
Ref. 0	0	0	2	3
Ref. 1	0	0	1	1
Ref. 2	0	0	0	2
Ref. 3	0	0	0	0

Input matrix:



0	0	1	1	1
0	0	1	1	1
0	2	2	2	2
2	2	3	3	3
2	2	3	3	3

Offset:

(2,0)

GLCM matrix:

	C. 0	C. 1	C. 2	C. 3
Ref. 0	1	0	4	0
Ref. 1	0	0	3	3
Ref. 2	0	0	1	3
Ref. 3	0	0	0	0

DRAM map

- DRAM supports both read and write mode, so you need to connect it with AXI-4 read and write related signals.

```
module PATTERN #(parameter ID_WIDTH=4, DATA_WIDTH=32, ADDR_WIDTH=32)(
// axi write address channel
input wire [ID_WIDTH-1:0]        awid_s_inf;
input wire [ADDR_WIDTH-1:0]    awaddr_s_inf;
input wire [2:0]             awsize_s_inf;
input wire [1:0]            awburst_s_inf;
input wire [3:0]              awlen_s_inf;
input wire                  awvalid_s_inf;
output wire                 awready_s_inf;
// axi write data channel
input wire [DATA_WIDTH-1:0]     wdata_s_inf;
input wire                   wlast_s_inf;
input wire                  wvalid_s_inf;
output wire                 wready_s_inf;
// axi write response channel
output wire [ID_WIDTH-1:0]         bid_s_inf;
output wire [1:0]             bresp_s_inf;
output wire                   bvalid_s_inf;
input wire                    bready_s_inf;
// ----------------------------
// axi read address channel
input wire [ID_WIDTH-1:0]       arid_s_inf;
input wire [ADDR_WIDTH-1:0]   araddr_s_inf;
input wire [3:0]             arlen_s_inf;
input wire [2:0]            arsize_s_inf;
input wire [1:0]           arburst_s_inf;
input wire                 arvalid_s_inf;
output wire                arready_s_inf;
// ----------------------------
// axi read data channel
output wire [ID_WIDTH-1:0]         rid_s_inf;
output wire [DATA_WIDTH-1:0]     rdata_s_inf;
output wire [1:0]              rresp_s_inf;
output wire                    rlast_s_inf;
output wire                   rvalid_s_inf;
input wire                    rready_s_inf;
// ----------------------------
```

From: 0x0000
To : 0x0FFF
Kernel Not Accessible

From: 0x1000
To : 0x1FFF
Input matrix: read only

From: 0x2000
To : 0x2FFF
GLCM: read and write

→ **Read & Write**

.dat file example

```
@1000
c f 15 0
@1004
3 1b 3 7
@1008
9 13 15 12
@100c
4 17 6 18
@1010
18 c 1a 1
@1014
```

@1000

C	f	15	0
[7:0]	[15:8]	[23:16]	[31:24]

Address 1000 : data = C
Address 1001 : data = f
Address 1002 : data = 15
Address 1003 : data = 0

GLCM: 8 bit for each data
Input matrix: 8 bit for each data

DRAM note

- You may modify the following part in:
  ../00_TESTBED/pseudo_DRAM.v

```
parameter DRAM_p_r = "../00_TESTBED/DRAM/pseudo_DRAM.dat";

MAX_WAIT_READY_CYCLE=300;

`ifdef FUNC
`define LAT_MAX 20
`define LAT_MIN 1
`endif
`ifdef PERF
`define LAT_MAX 20
`define LAT_MIN 1
`endif
```