4DV4-MiniProject-03
Simple Image Processing and Display Controller

COMPENG 4DV4: Very Large Scale Integration System Design

Deadline

Last day to submit this project is Tuesday April 7 (7/4/2026) by 11:59 p.m

1 Data Preparation

1. Extract assignment:

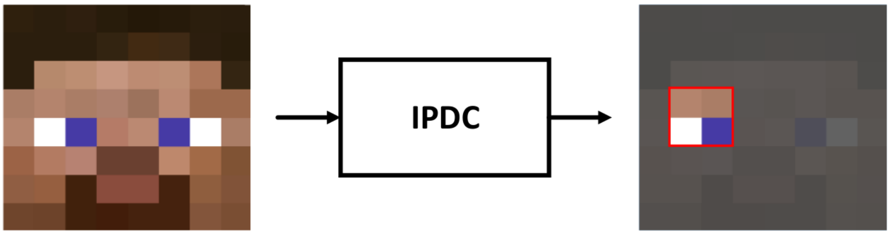
```
tar -xvf 1101_hw3.tar
```

2. Content description:

Folder	File	Description
00_TESTBED	testbed_temp.v	Testbench template
00_TESTBED/PATTERN	Indata*.dat	Input image test patterns
	opmode*.dat	Processing mode test patterns
	golden*.dat	Golden display result patterns
01_RTL	ipdc.v	Design implementation
	rtl_01.f	RTL simulation file list
	01_run	NCVerilog execution script
	99_cleaan_up	Temporary file cleanup script
02_SYN	syn.tcl	Synthesis script
	ipdc_dc.sdc	Timing constraints
	02_run.dc	Design Compiler execution script
03_GATE	rtl_03.f	Gate-level file list
	03_run	Gate-level NCVerilog script
	99_cleaan	Cleanup script
sram_****x8	.v/.db/.lib	SRAM IP and models
top report.txt: Submission report form		

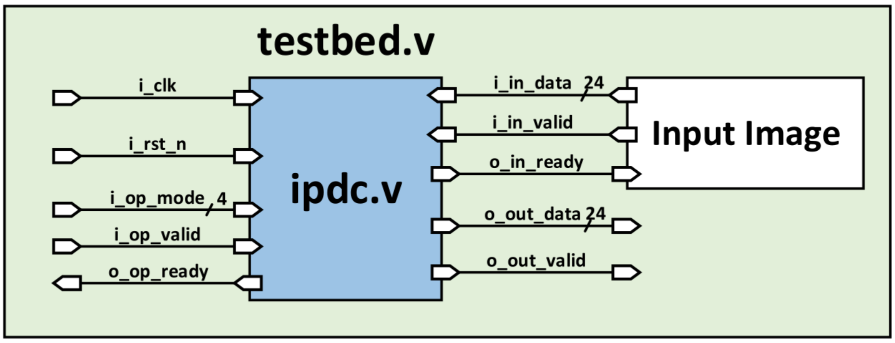
2 Introduction

This project focuses on image display functionality for embedded systems. You will design an Image Processing and Display Controller (IPDC) that supports several pixel manipulation operations. A 16×16 RGB image is first loaded then processed based on operation requests. You are required to do the APR flow after you finish the design.



3 System Architecture

Your IPDC module interfaces with the testbench for loading image data and performing display operations on demand.



4 Specifications

1. Top module name: `ipdc`
2. Port descriptions:

Signal	I/O	Width	Description
`i_clk`	I	1	System clock
`i_rst_n`	I	1	Active-low asynchronous reset
`i_op_valid`	I	1	Valid flag for processing mode
`i_op_mode`	I	4	Operation mode identifier
`o_op_ready`	O	1	Ready flag for next operation command
`i_in_valid`	I	1	Valid flag for input pixel data
`i_in_data`	I	24	RGB input pixel: [23:16] R, [15:8] G, [7:0] B
`o_in_ready`	O	1	Ready flag for receiving next pixel (mode 0 only)
`o_out_valid`	O	1	Output pixel valid
`o_out_data`	O	24	Output pixel RGB/YCbCr coded

3. Input sampling occurs on falling edges.

4. Outputs must be registered at rising edge.

5. All outputs cleared upon reset.

6. `i_op_mode` valid only one cycle along with `i_op_valid`.

7. Prohibit simultaneous assertion of the following signal pairs:

   - `i_in_valid` with `o_op_ready`
   - `i_op_valid` with `o_op_ready`
   - `i_in_valid` with `o_out_valid`
   - `i_op_valid` with `o_out_valid`
   - `o_op_ready` with `o_out_valid`

8. `o_op_ready` must pulse high once per valid operation command response.

9. `o_out_valid` must assert only when output pixel data is correct.

10. At least one SRAM instance must exist in the design.

11. Latency from `i_op_valid` to `o_op_ready` should be less than 1000ns for all operations except mode 0000.

12. Only worst-case library is allowed for synthesis.

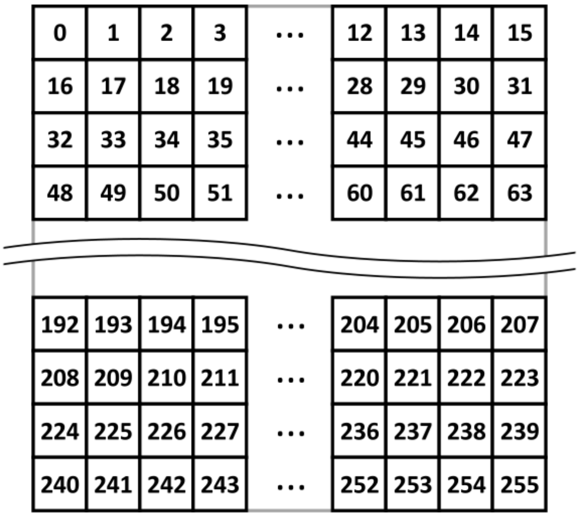
13. No inferred latches permitted.

14. Timing slack must be non-negative and no timing violations allowed in gate-level simulation.

5 Design Description

5.1 Image Input

- Image dimension: 16×16 pixels × 3 channels (RGB)
- Input ordering: raster scan (left-to-right, top-to-bottom)



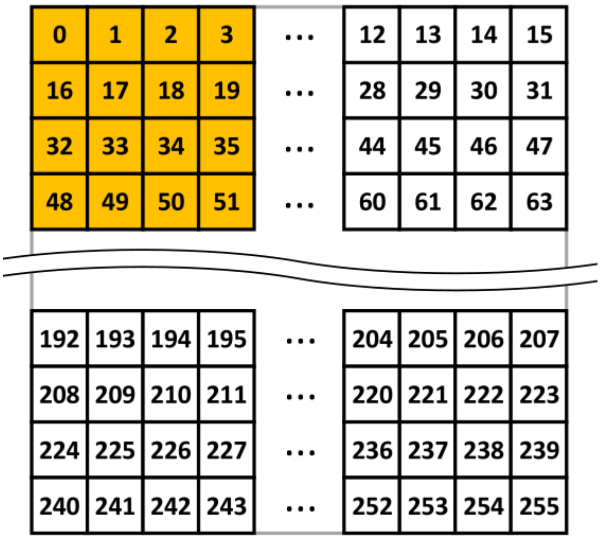
0	1	2	3	⋯	12	13	14	15
16	17	18	19	⋯	28	29	30	31
32	33	34	35	⋯	44	45	46	47
48	49	50	51	⋯	60	61	62	63
192	193	194	195	⋯	204	205	206	207
208	209	210	211	⋯	220	221	222	223
224	225	226	227	⋯	236	237	238	239
240	241	242	243	⋯	252	253	254	255

Figure 1: Raster scan

- Perform 256 input cycles
- Pause input streaming if `o_in_ready = 0`
- Signal readiness using `o_op_ready = 1` once loading completes

5.2 Display Ordering

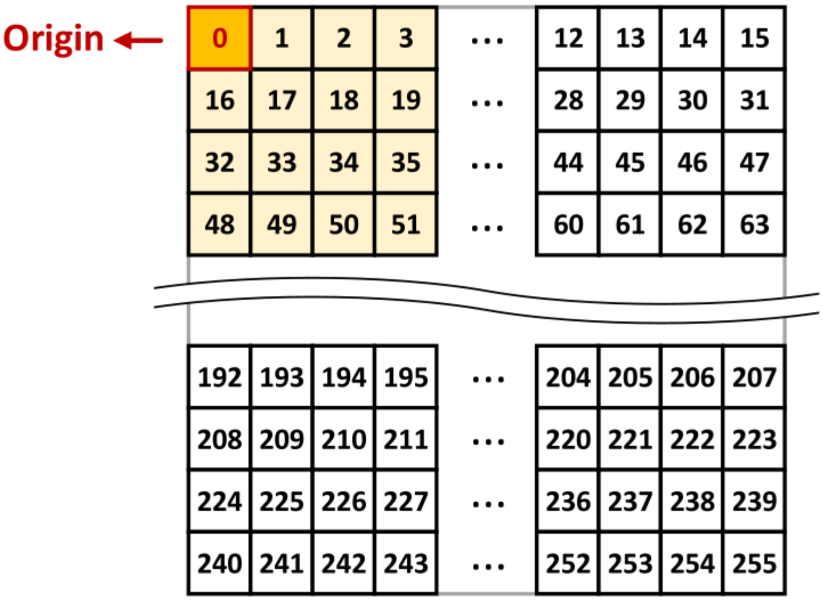
Output is also raster scan ordered.



0	1	2	3	...	12	13	14	15
16	17	18	19	...	28	29	30	31
32	33	34	35	...	44	45	46	47
48	49	50	51	...	60	61	62	63
192	193	194	195	...	204	205	206	207
208	209	210	211	...	220	221	222	223
224	225	226	227	...	236	237	238	239
240	241	242	243	...	252	253	254	255

5.3 Origin Handling

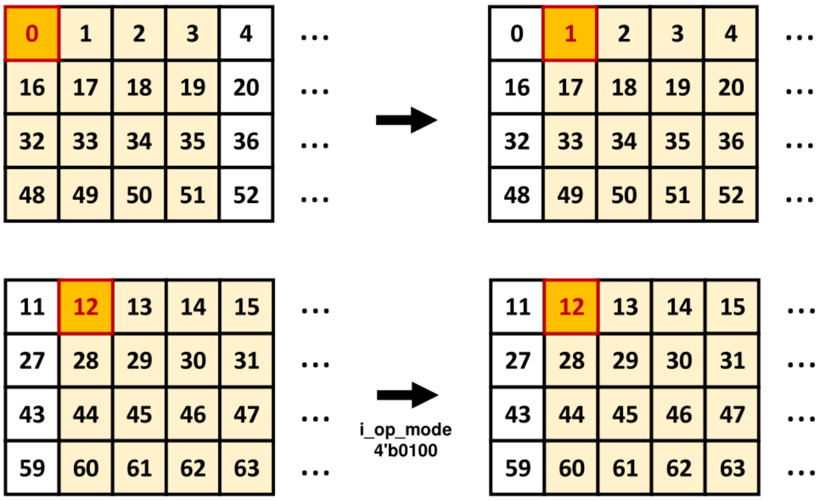
The first displayed pixel is treated as the origin location. Origin position shifts based on operations.



0	1	2	3	...	12	13	14	15
16	17	18	19	...	28	29	30	31
32	33	34	35	...	44	45	46	47
48	49	50	51	...	60	61	62	63
192	193	194	195	...	204	205	206	207
208	209	210	211	...	220	221	222	223
224	225	226	227	...	236	237	238	239
240	241	242	243	...	252	253	254	255

5.4 Origin shifting

EX. Origin right shift (i_op_mode = 4'b0100). If output of display exceeds the image boundary, retain the same origin point.



5.5 Operation Modes

Mode	Meaning	Display Required
0000	Load input image	No
0100	Shift origin right	Yes
0101	Shift origin left	Yes
0110	Shift origin up	Yes
0111	Shift origin down	Yes
1000	Scale down	Yes
1001	Scale up	Yes
1100	Median filtering	Yes
1101	YCbCr output	Yes
1110	Census transform	Yes

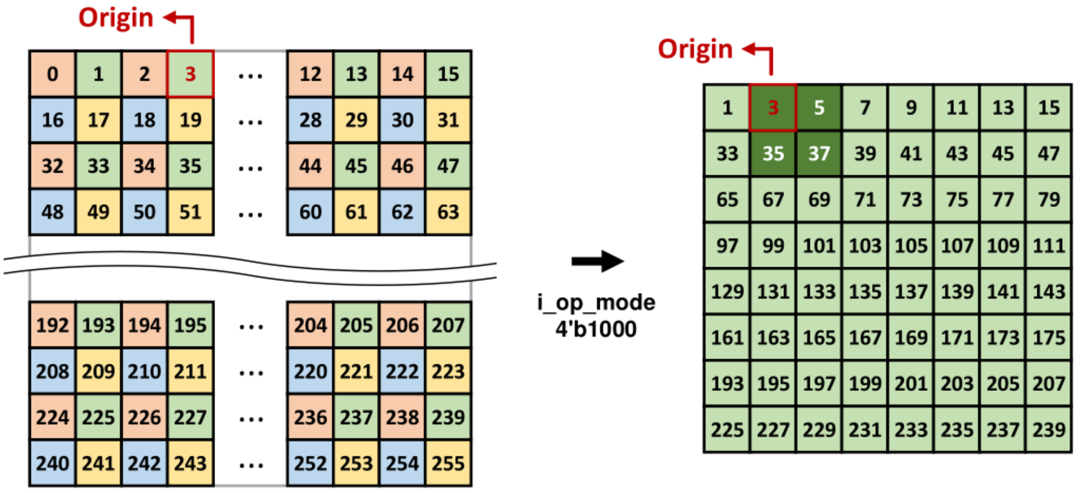
5.6 Image Scaling Rules

Three supported sizes:

Image Size	Display Size
16×16	4×4
8×8	2×2
4×4	1×1

Scale Down

- 16→8→4
- Maintain origin alignment



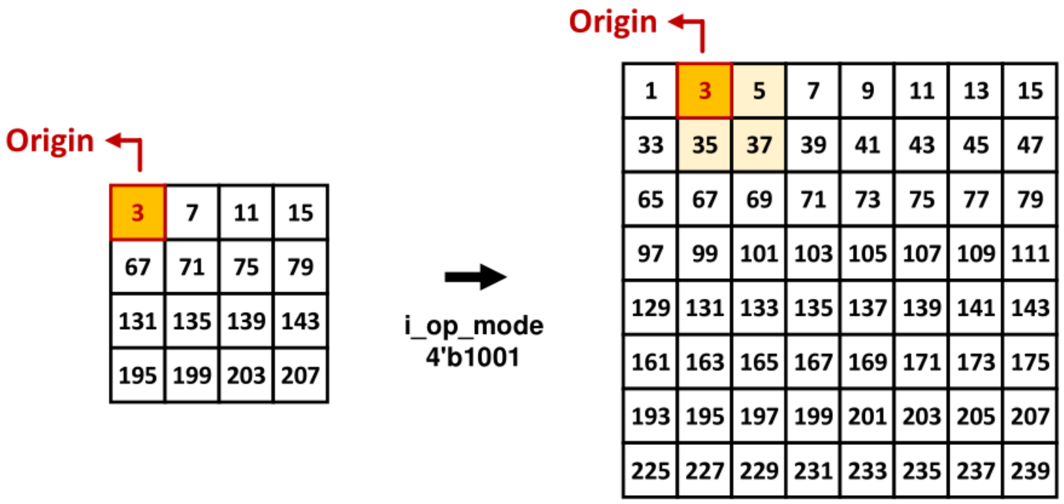
0	1	2	3	…	12	13	14	15
16	17	18	19	…	28	29	30	31
32	33	34	35	…	44	45	46	47
48	49	50	51	…	60	61	62	63
192	193	194	195	…	204	205	206	207
208	209	210	211	…	220	221	222	223
224	225	226	227	…	236	237	238	239
240	241	242	243	…	252	253	254	255

1	3	5	7	9	11	13	15
33	35	37	39	41	43	45	47
65	67	69	71	73	75	77	79
97	99	101	103	105	107	109	111
129	131	133	135	137	139	141	143
161	163	165	167	169	171	173	175
193	195	197	199	201	203	205	207
225	227	229	231	233	235	237	239

Figure 2: Scale Down

Scale Up

- 4→8→16
- Maintain relationship to previously scaled areas
- When display reaches right boundary, do not enlarge further



3	7	11	15
67	71	75	79
131	135	139	143
195	199	203	207

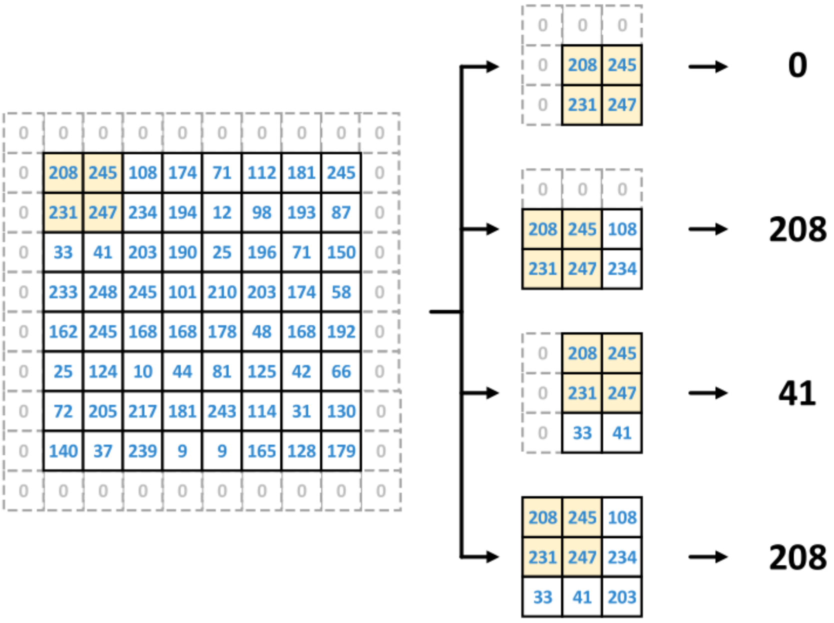
1	3	5	7	9	11	13	15
33	35	37	39	41	43	45	47
65	67	69	71	73	75	77	79
97	99	101	103	105	107	109	111
129	131	133	135	137	139	141	143
161	163	165	167	169	171	173	175
193	195	197	199	201	203	205	207
225	227	229	231	233	235	237	239

Figure 3: Scale Up

5.7 Median Filter

- Apply 3×3 median operator to each pixel in display region
- Zero pad boundary pixels
- Operate per-channel (R, G, B)

- Do not modify original image data



0	0	0	0	0	0	0	0	0	0
0	208	245	108	174	71	112	181	245	0
0	231	247	234	194	12	98	193	87	0
0	33	41	203	190	25	196	71	150	0
0	233	248	245	101	210	203	174	58	0
0	162	245	168	168	178	48	168	192	0
0	25	124	10	44	81	125	42	66	0
0	72	205	217	181	243	114	31	130	0
0	140	37	239	9	9	165	128	179	0
0	0	0	0	0	0	0	0	0	0

0	0	0
0	208	245
0	231	247

→ 0

0	0	0
208	245	108
231	247	234

→ 208

0	208	245
0	231	247
0	33	41

→ 41

208	245	108
231	247	234
33	41	203

→ 208

Figure 4: Median Filter

5.8 YCbCr Output

- Convert display region into YCbCr using:

$$Y = 0.25R + 0.625G$$

$$Cb = -0.125R - 0.25G + 0.5B + 128$$

$$Cr = 0.5R - 0.375G - 0.125B + 128$$

- Apply rounding only at the final stage
- No intermediate truncation

5.9 Census Transform

- Compare each neighborhood pixel to center pixel in a 3×3 region
- Generate 8-bit output per channel (per R/G/B)
- Zero padded edges
- Source image remains unmodified

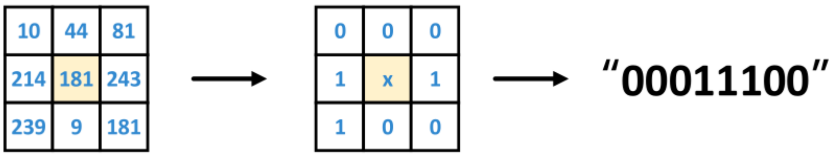
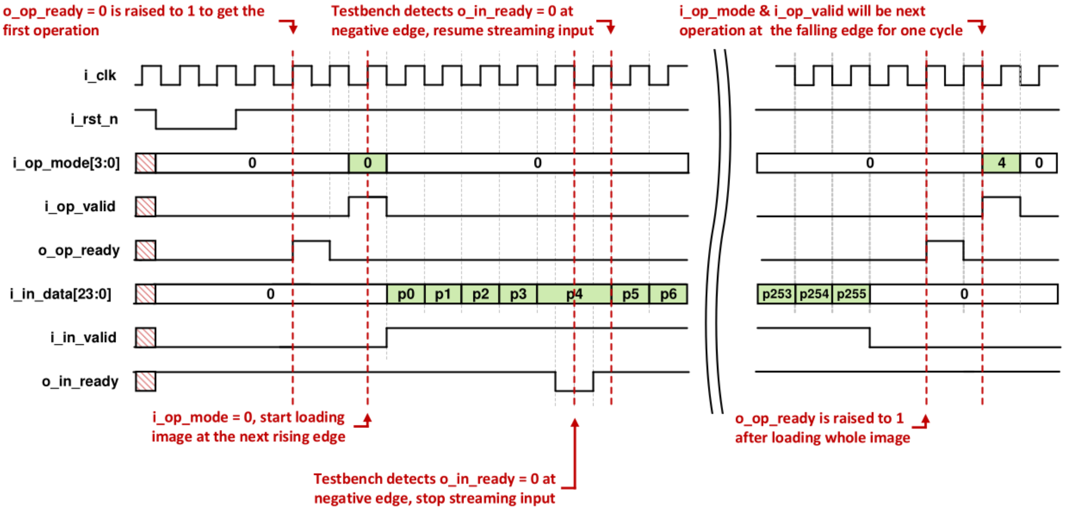


Figure 5: Census Transform

6 Sample Waveform

- Load Image Data (i_op_mode = 0)



o_op_ready = 0 is raised to 1 to get the first operation

Testbench detects o_in_ready = 0 at negative edge, resume streaming input

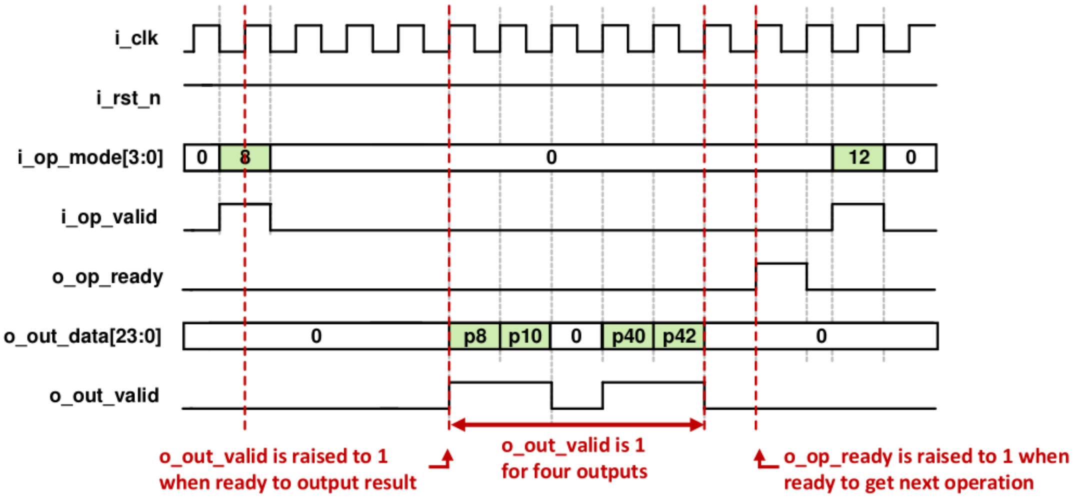
i_op_mode & i_op_valid will be next operation at the falling edge for one cycle

i_op_mode = 0, start loading image at the next rising edge

Testbench detects o_in_ready = 0 at negative edge, stop streaming input

o_op_ready is raised to 1 after loading whole image

- Other operations



o_out_valid is raised to 1 when ready to output result

o_out_valid is 1 for four outputs

o_op_ready is raised to 1 when ready to get next operation

7 Submission Requirements

1. Create folder: `Proj3_studentID` that includes all your design files and a report showing your work with explanations and screenshots.

2. Compress and upload it to its corresponding Dropbox on avenue

8 Grading Policy

1. Simulation must run without errors:

```
ncverilog -f rtl_01.f +notimingchecks +access+r +define+tb0
```

2. Correct execution: 25

Pattern	Description	RTL
tb0	Load + shift	5%
tb1	Load + shift + scale	5%
tb2	Load + shift + filter	5%
tb3	Full functionality	5%
hidden	10 hidden cases	5%

3. APR report: 35%

4. Correctness of mmmc.view setting: 10%

5. Correctness of simulation after APR: 30%