Microprocessor Applications
(Term Project 2)

KU KONKUK UNIVERSITY

TP (2)

2025. 6. 16.

Students
202110410 조민우
202110365 김상만

Outline

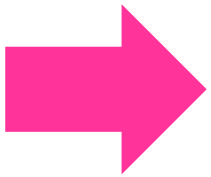
- 64pt FFT Accelerator with Multiple Cores
  - Result of TP_2
  - Code Review
  - How to Debug
  - Ablation Study
  - Roofline Model
  - How to improve
  - Summary

Result of TP_2

KU KONKUK UNIVERSITY

- **Accuracy & Performance**

```
Output count has reached the total number of samples.
Validating bit-accurate output...
    - Total mismatches: 0 / 192
Checking SRAM occupancy...
    - SRAM 2: occupancy = 32
    - SRAM 3: occupancy = 32
Test passed. Ending simulation.
$finish called from file "../../tb/tb_mem_IO.v", line 132.
$finish at simulation time                6634000
Simulation complete, time is 6634000 ps.
```



- **Accuracy**

  - Total mismatches : 0/192

- **Performance**

  - Simulation Time : 6,634,000[ps]

- **Memory Capacity**

  - Used Memory: SRAM 0: 32(128B)
  - Used Memory: SRAM 1: 32(128B)

Code Review Overview

KU KONKUK UNIVERSITY

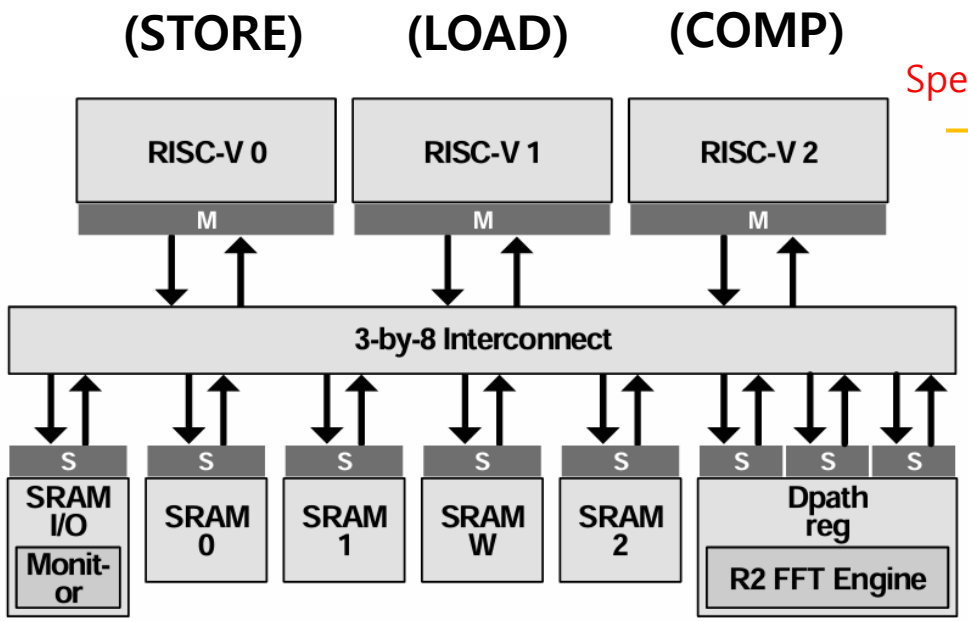
- **Implement POV**
  - Pipeline
  - Effective Memory Capacity
  - Synchronizing with dummy

- **Code Review**

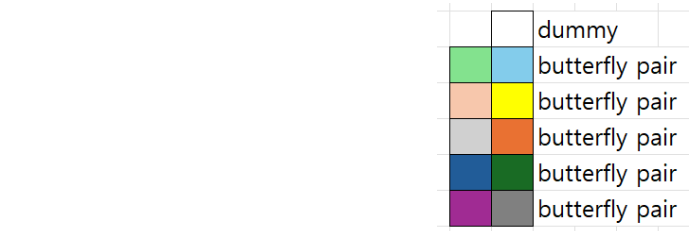
Code Review (1/7)

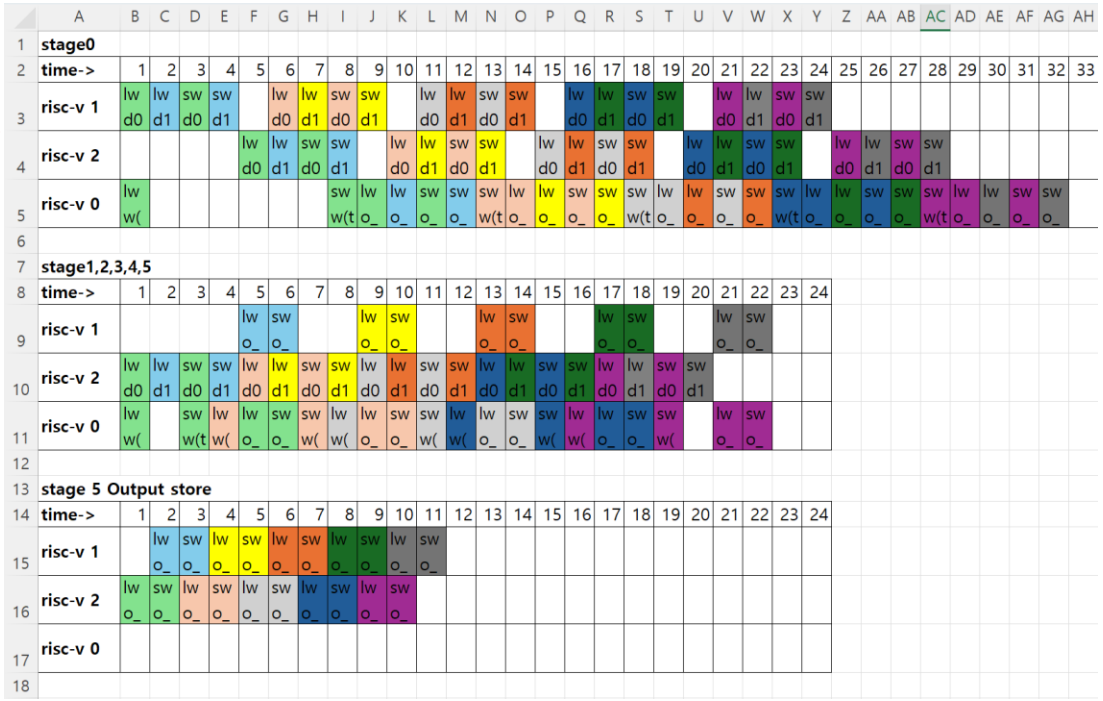
- **Implement POV**
  - Pipelining with 3 cores (HOW?)

    - Distribute the roles among the cores



Specify Cores roles





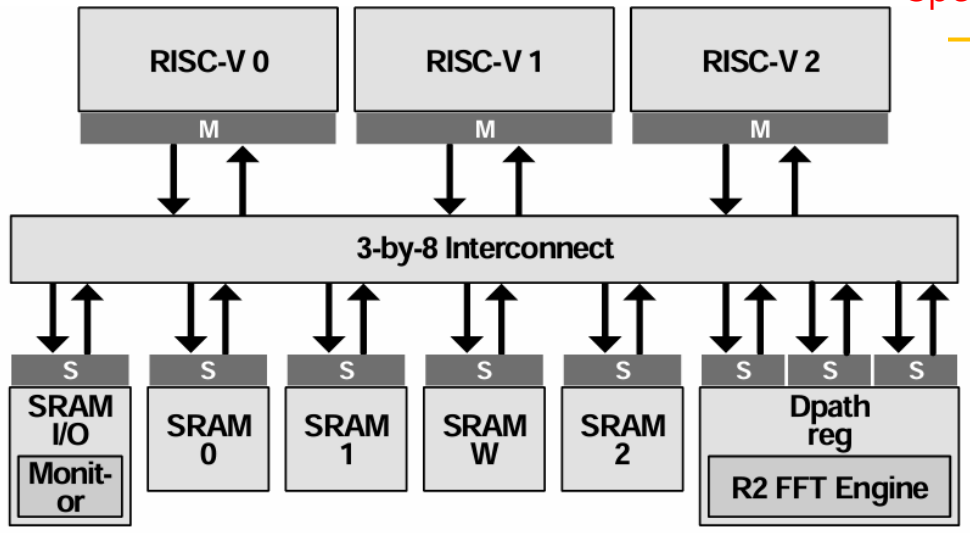
Code Review (2/7)

KU KONKUK UNIVERSITY

- **Implement POV-cont'd**
  - Pipelining with 3 cores (HOW?)

    - Distribute the roles among the cores

STAGE 0



Specify Cores roles

Load Twiddle Factor & use 6 dummy codes before starts

RISC-V 0

```
# Load twiddle factors for all stages (32 values)
# Butterfly 0-31: Load W[0] for stage 0 (all W[0])

# Butterfly 0
sw t4, 0(s6)      # W[0](t4->dpath2)
lw t5, 4(s4)      # o_d0(dpath0->t5)
lw t6, 4(s5)      # o_d0(dpath1->t6)
sw t5, 0(s1)      # o_d0(t5->SRAM_0)
sw t6, 0(s2)      # o_d0(t6->SRAM_1)
```

RISC-V 1

```
# Butterfly 0
lw      t0, 0(s0)       # d0(SRAM I/O->t0)
lw      t1, 128(s0)     # d1(SRAM I/O->t1)
sw      t0, 0(s1)       # d0(t0->SRAM_0)
sw      t1, 0(s2)       # d1(t1->SRAM_1)
addi    x0, x0, 1       # dummy
```

RISC-V 2

```
# Butterfly 0
lw      t2, 0(s1)       # d0(SRAM_0->t2)
lw      t3, 0(s2)       # d1(SRAM_1->t3)
sw      t2, 0(s4)       # d0(t2->dpath0)
sw      t3, 0(s5)       # d1(t3->dpath1)
addi    x0,     x0,     0x0000      # dummy
addi    x0,     x0,     0x0000      # dummy
```

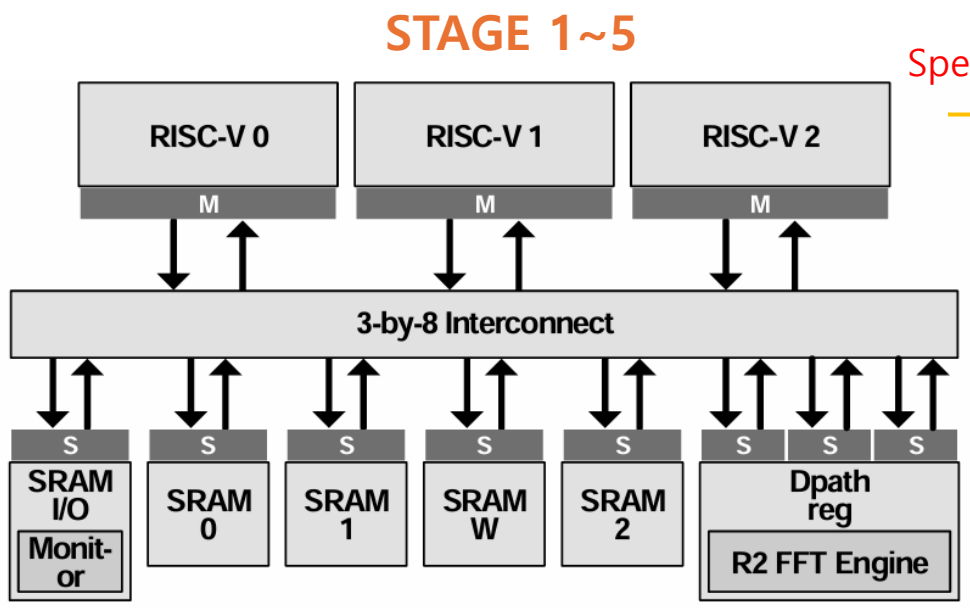
Use 6 dummy codes before starts

Code Review (3/7)

KU KONKUK UNIVERSITY

- **Implement POV-cont'd**
  - Pipelining with 3 cores (HOW?)

    - Distribute the roles among the cores



Specify Cores roles

RISC-V 0

```
# Butterfly 0
lw t4, 0(s3)       # W[0](SRAM_W->t4)
addi    x0, x0,  1       # dummy
sw t4, 0(s6)       # W[0](t4->dpath2)
lw t4, 64(s3)      # W[16](SRAM_W->t4)_next twiddle factor
lw t5, 4(s4)       # o_d0(dpath0->t5)
sw t5, 0(s1)       # o_d0(t5->SRAM_0)

# Butterfly 1
sw t4, 0(s6)       # W[16](t4->dpath2)
lw t4, 0(s3)       # W[0](SRAM_W->t4)_next twiddle factor
lw t5, 4(s4)       # o_d0(dpath0->t5)
sw t5, 0(s2)       # o_d0(t5->SRAM_1)
```

RISC-V 1

```
# Butterfly operations (32 total)

# Butterfly 0
lw      t6, 4(s5)        # o_d1(dpath1->t6)
sw      t6, 4(s1)        # o_d1(t6->SRAM_1)
addi    x0, x0,  1       # dummy
```

RISC-V 2

```
# Butterfly 0
lw t2, 0(s1)       # d0(SRAM_0->t2)
lw t3, 4(s1)       # d1(SRAM_0->t3)
sw t2, 0(s4)       # d0(t2->dpath0)
sw t3, 0(s5)       # d1(t3->dpath1)
```

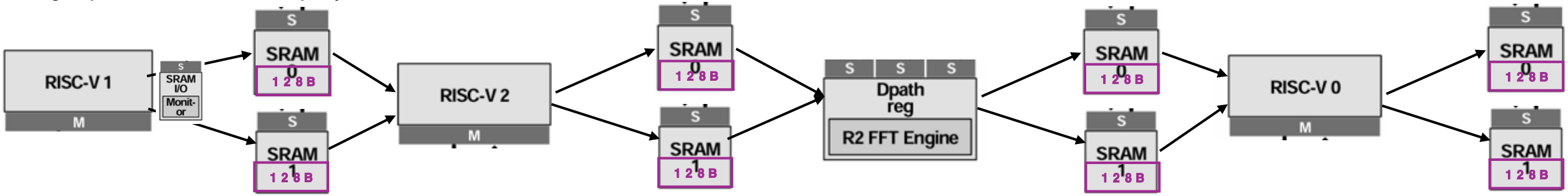
Code Review (4/7)

KU KONKUK UNIVERSITY

- **Implement POV-cont'd**
  - Effective Memory Capacity

    - Memory is reused after use

(The "reg" expression was omitted for simplicity)




```
lw      t0, 0(s0)       # d0(SRAM I/O->t0)
lw      t1, 128(s0)     # d1(SRAM I/O->t1)
sw      t0, 0(s1)       # d0(t0->SRAM_0)
sw      t1, 0(s2)       # d1(t1->SRAM_1)
addi    x0, x0, 1       # dummy
```

<Store Input to SRAM 0, 1>

```
lw      t2, 0(s1)       # d0(SRAM_0->t2)
lw      t3, 0(s2)       # d1(SRAM_1->t3)
sw      t2, 0(s4)       # d0(t2->dpath0)
sw      t3, 0(s5)       # d1(t3->dpath1)
addi    x0,     x0,     0x0000      # dummy
addi    x0,     x0,     0x0000      # dummy
```

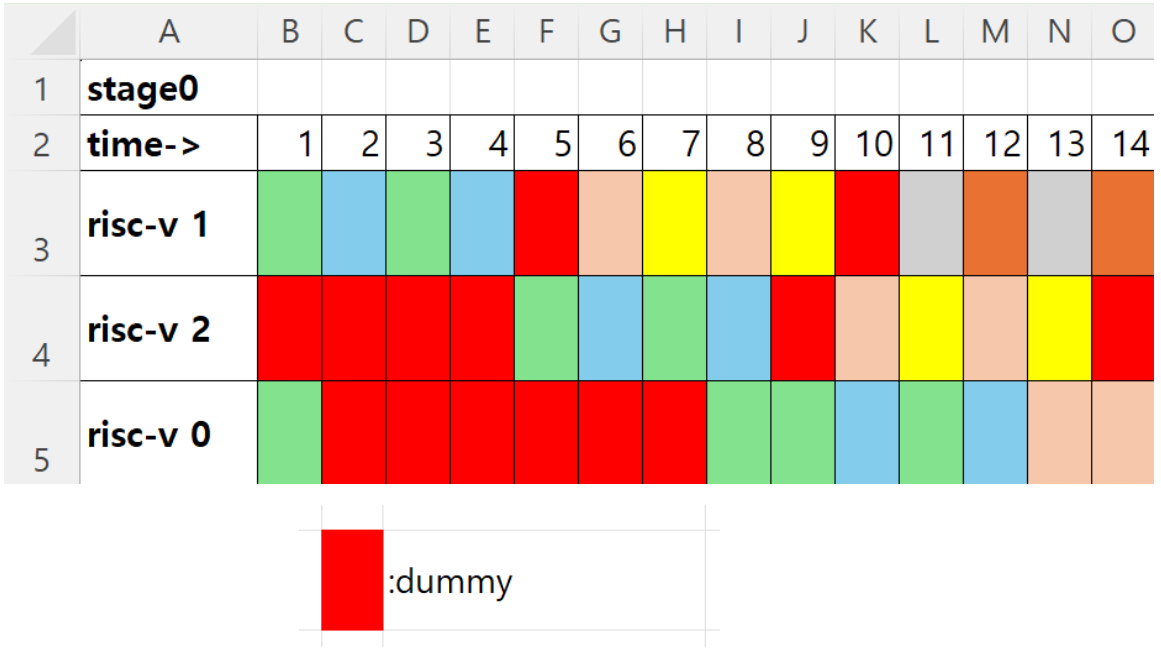
<Store dpath output to SRAM 0, 1>

```
sw t4, 0(s6)        # W[0](t4->dpath2)
lw t5, 4(s4)        # o_d0(dpath0->t5)
lw t6, 4(s5)        # o_d0(dpath1->t6)
sw t5, 0(s1)        # o_d0(t5->SRAM_0)
sw t6, 0(s2)        # o_d0(t6->SRAM_1)
```

<Store dpath output to SRAM 0, 1>

Code Review (6/7)

- **Implement POV-cont'd**
  - Synchronizing with dummy
    - Dummy codes were inserted during IDLE state when a core had no tasks to perform



Synchronizing with other cores!!

```
lw      t4, 0(s3)       # W[0](SRAM_W->t4)

addi    x0, x0,  1      # dummy
addi    x0, x0,  1      # dummy
addi    x0, x0,  1      # dummy
addi    x0, x0,  1      # dummy
addi    x0, x0,  1      # dummy
addi    x0, x0,  1      # dummy

# Load twiddle factors for all stages (32 values)
# Butterfly 0-31: Load W[0] for stage 0 (all W[0])

# Butterfly 0
sw t4, 0(s6)        # W[0](t4->dpath2)
lw t5, 4(s4)        # o_d0(dpath0->t5)
lw t6, 4(s5)        # o_d0(dpath1->t6)
sw t5, 0(s1)        # o_d0(t5->SRAM_0)
sw t6, 0(s2)        # o_d0(t6->SRAM_1)
```

[Dummy codes in stage 0 (RISC-V 0)]

Code Review (7/7)

KU KONKUK UNIVERSITY

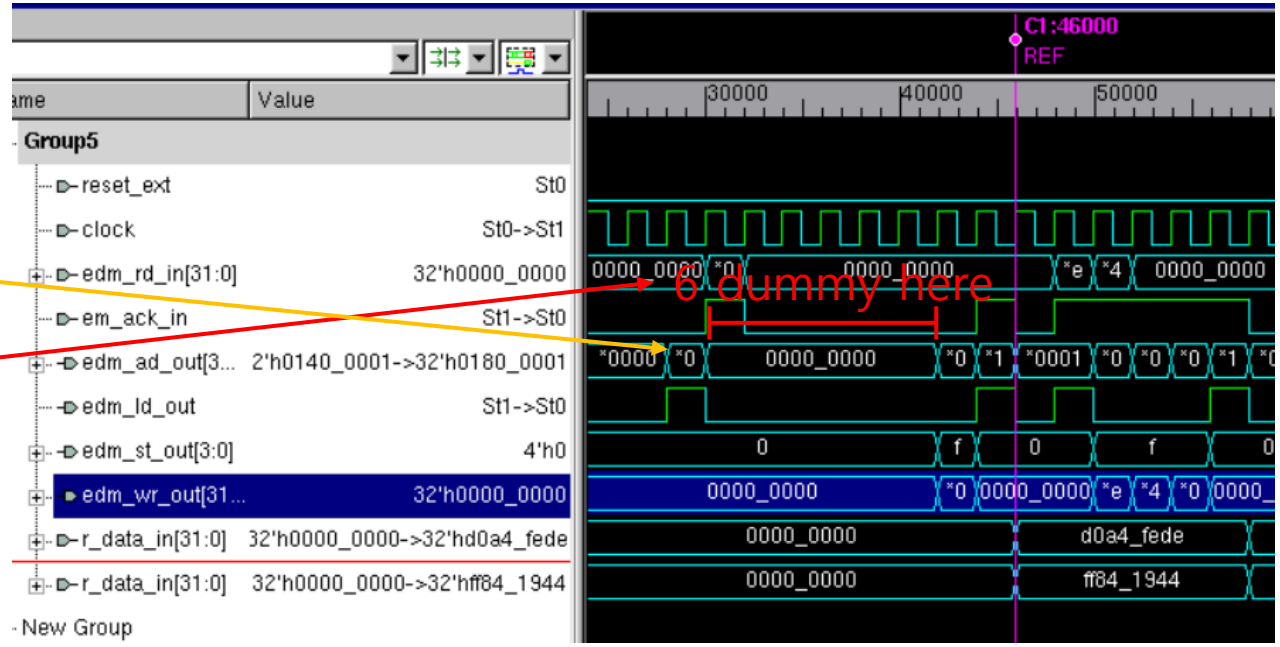
- **Implement POV-cont'd**
  - Synchronizing with dummy
    - Dummy codes were inserted during IDLE state when a core had no tasks to perform

```
lw      t4, 0(s3)       # W[0](SRAM_W->t4)

addi    x0, x0,  1      # dummy
addi    x0, x0,  1      # dummy
addi    x0, x0,  1      # dummy
addi    x0, x0,  1      # dummy
addi    x0, x0,  1      # dummy
addi    x0, x0,  1      # dummy

# Load twiddle factors for all stages (32 values)
# Butterfly 0-31: Load W[0] for stage 0 (all W[0])

# Butterfly 0
sw t4, 0(s6)      # W[0](t4->dpath2)
lw t5, 4(s4)      # o_d0(dpath0->t5)
lw t6, 4(s5)      # o_d0(dpath1->t6)
sw t5, 0(s1)      # o_d0(t5->SRAM_0)
sw t6, 0(s2)      # o_d0(t6->SRAM_1)
```



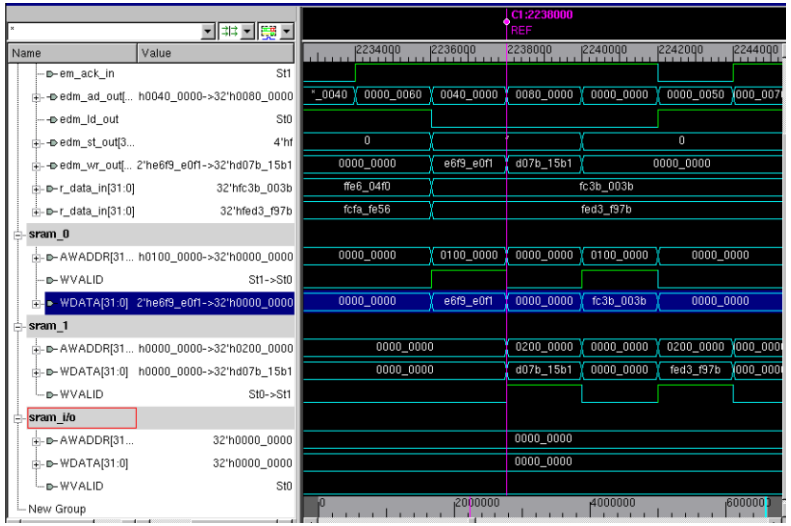
How to Debug (1/7)

KU KONKUK UNIVERSITY

- Using C & Excel for debugging with Waveforms

```
if (set == 0 && l == 1)
{
    printf("State of SRAM_0 after 0 stage of first set\n");
    for (j = 0; j < L_FFT; j++)
        printf("SRAM_0[%2d] = \t0x%8x\n", j, SRAM_0[j]);
}
if (set == 0 && l == 2)
{
    printf("State of SRAM_0 after 1 stage of first set\n");
    for (j = 0; j < L_FFT; j++)
        printf("SRAM_0[%2d] = \t0x%8x\n", j, SRAM_0[j]);
}
if (set == 0 && l == 4)
{
    printf("State of SRAM_0 after 2 stage of first set\n");
    for (j = 0; j < L_FFT; j++)
        printf("SRAM_0[%2d] = \t0x%8x\n", j, SRAM_0[j]);
}
if (set == 0 && l == 8)
{
    printf("State of SRAM_0 after 3 stage of first set\n");
    for (j = 0; j < L_FFT; j++)
        printf("SRAM_0[%2d] = \t0x%8x\n", j, SRAM_0[j]);
}
if (set == 0 && l == 16)
{
    printf("State of SRAM_0 after 4 stage of first set\n");
    for (j = 0; j < L_FFT; j++)
        printf("SRAM_0[%2d] = \t0x%8x\n", j, SRAM_0[j]);
}
```

stage0		stage1		stage2		stage3		stage4		stage5	
SRAM_0[0] =	0xd0a4fede	SRAM_0[0] =	0xee7cf4bc	SRAM_0[0] =	0xed3bfb07	SRAM_0[0] =	0xf515fe6e	SRAM_0[0] =	0xfacafd64	SRAM_0[0] =	0xfb0ff9b7
SRAM_0[1] =	0xff841944	SRAM_1[0] =	0x 8fe0df3	SRAM_0[1] =	0xf9d9103d	SRAM_0[1] =	0x 4a41692	SRAM_0[1] =	0x 1e10b08	SRAM_0[1] =	0xfd6706c0
SRAM_0[2] =	0x c54ea9a	SRAM_0[4] =	0xe2280a22	SRAM_0[2] =	0xf068fcf7	SRAM_0[2] =	0xf7f0043f	SRAM_0[2] =	0x 3c9fde2	SRAM_0[2] =	0x 35cfb3c
SRAM_0[3] =	0xfd5e1278	SRAM_1[4] =	0xf6860b51	SRAM_0[3] =	0xfd0406f7	SRAM_0[3] =	0xfe7809ad	SRAM_0[3] =	0x 8200af	SRAM_0[3] =	0xfdbf02c3
SRAM_0[4] =	0xfc2ffffa	SRAM_0[8] =	0xebf90152	SRAM_0[4] =	0x 141f9b5	SRAM_0[4] =	0x 768fd4c	SRAM_0[4] =	0x 4adfb93	SRAM_0[4] =	0x 1330140
SRAM_0[5] =	0xdf83feac	SRAM_1[8] =	0xe3d6fe0a	SRAM_0[5] =	0x f25fdb5	SRAM_0[5] =	0x 96eff9d	SRAM_0[5] =	0x 8abfe70	SRAM_0[5] =	0x 4e20010
SRAM_0[6] =	0xdbc402aa	SRAM_0[12] =	0x1035fea8	SRAM_0[6] =	0xf1c00d2b	SRAM_0[6] =	0xf52e0a27	SRAM_0[6] =	0x 3b7054d	SRAM_0[6] =	0x 2e70157
SRAM_0[7] =	0x 298e82a	SRAM_1[12] =	0xfbac00a2	SRAM_0[7] =	0xf9820459	SRAM_0[7] =	0x 2c107a1	SRAM_0[7] =	0xfa9305e9	SRAM_0[7] =	0xfad4001e
SRAM_0[8] =	0xf3b20ced	SRAM_0[16] =	0xfc0c0f69	SRAM_0[8] =	0xfcf001d7	SRAM_0[8] =	0xf825fc98	SRAM_0[8] =	0xfd8802ba	SRAM_0[8] =	0x 2f0441
SRAM_0[9] =	0xe5a022a3	SRAM_1[16] =	0x 5e2376	SRAM_0[9] =	0x 332209d	SRAM_0[9] =	0xf534f9aa	SRAM_0[9] =	0xfda000d8	SRAM_0[9] =	0xfc0f0357
SRAM_0[10] =	0x 46711e6	SRAM_0[10] =	0xf7a6fd83	SRAM_0[10] =	0xf77807c8	SRAM_0[10] =	0xf878f8b7	SRAM_0[10] =	0x 4ffaf1	SRAM_0[10] =	0x 3ecfaee
SRAM_0[11] =	0xdbb71b1c	SRAM_1[11] =	0xe542ff2c	SRAM_0[11] =	0xf48504aa	SRAM_0[11] =	0xfe8cfd49	SRAM_0[11] =	0xfeb9ff24	SRAM_0[11] =	0xfe73ff9e
SRAM_0[12] =	0xebc7eb91	SRAM_0[12] =	0xfdd4f446	SRAM_0[12] =	0xff1c0d91	SRAM_0[12] =	0xf9d8fc68	SRAM_0[12] =	0xfe16fe80	SRAM_0[12] =	0xfdd1ff34
SRAM_0[13] =	0xe55d14cd	SRAM_0[13] =	0xef36194e	SRAM_0[13] =	0xfd2c02d9	SRAM_0[13] =	0x 5b6fe17	SRAM_0[13] =	0x 3f0470	SRAM_0[13] =	0xfe5f02d5
SRAM_0[14] =	0x fe2fcfb	SRAM_0[14] =	0xedf2f74a	SRAM_0[14] =	0x 2ef5ba	SRAM_0[14] =	0xfc920303	SRAM_0[14] =	0xfbabfd62	SRAM_0[14] =	0xff5f050c
SRAM_0[15] =	0xe230f911	SRAM_0[15] =	0xf626fb7e	SRAM_0[15] =	0xf0bdfa82	SRAM_0[15] =	0xf6c1fcb7	SRAM_0[15] =	0xf9dff970	SRAM_0[15] =	0xf8a4fc40
SRAM_0[16] =	0x1b101af2	SRAM_0[16] =	0xfb51fd8b	SRAM_0[16] =	0xf7a7ff47	SRAM_0[16] =	0x 81fc5a	SRAM_0[16] =	0xfa4a010a	SRAM_0[16] =	0xfcfe00fe
SRAM_0[17] =	0xffd0f27a	SRAM_0[17] =	0xfb33fce5	SRAM_0[17] =	0xf63003b3	SRAM_0[17] =	0xff3dff55	SRAM_0[17] =	0x 2c30b8a	SRAM_0[17] =	0x 63a0516
SRAM_0[18] =	0xdb92e023	SRAM_0[18] =	0x1fbf1d67	SRAM_0[18] =	0x112f06c0	SRAM_0[18] =	0x11b0fe25	SRAM_0[18] =	0xf427065c	SRAM_0[18] =	0xf58104f7
SRAM_0[19] =	0xf8b0f697	SRAM_0[19] =	0x 49df595	SRAM_0[19] =	0x 684f908	SRAM_0[19] =	0x 6bcfa82	SRAM_0[19] =	0xfdf608fd	SRAM_0[19] =	0x 407b9
SRAM_0[20] =	0x 3e503a6	SRAM_0[20] =	0xf3fe0105	SRAM_0[20] =	0x 3a9fe43	SRAM_0[20] =	0x 5b9fd07	SRAM_0[20] =	0x 2bb01b9	SRAM_0[20] =	0xff960103
SRAM_0[21] =	0xeaa10566	SRAM_0[21] =	0xee18fcf4	SRAM_0[21] =	0x 502f931	SRAM_0[21] =	0x 6aa050e	SRAM_0[21] =	0x c3012c	SRAM_0[21] =	0x 2a6ff3a
SRAM_0[22] =	0xe418fe64	SRAM_0[22] =	0x fe602a1	SRAM_0[22] =	0x e8f16a6	SRAM_0[22] =	0x 6911109	SRAM_0[22] =	0xf17704d9	SRAM_0[22] =	0xf76105d6
SRAM_0[23] =	0x b7ef190	SRAM_0[23] =	0xfc880872	SRAM_0[23] =	0xfe18fc8c	SRAM_0[23] =	0xf93bf37b	SRAM_0[23] =	0x 82d01b7	SRAM_0[23] =	0x 205015e
SRAM_0[24] =	0x1ac7fd9c	SRAM_0[24] =	0x d930138	SRAM_0[24] =	0x 95df96e	SRAM_0[24] =	0xf72502ec	SRAM_0[24] =	0xfa9cf9de	SRAM_0[24] =	0xfcf7ff6a
SRAM_0[25] =	0x1029141e	SRAM_0[25] =	0xf6c5ffbc	SRAM_0[25] =	0x 994fe86	SRAM_0[25] =	0xf6f3045d	SRAM_0[25] =	0xf794f8d2	SRAM_0[25] =	0xfa2efae8
SRAM_0[26] =	0x 6104d5	SRAM_0[26] =	0x d33fc64	SRAM_0[26] =	0x14430579	SRAM_0[26] =	0xff7e089b	SRAM_0[26] =	0xf829fdc5	SRAM_0[26] =	0xfc650537
SRAM_0[27] =	0x14a7dd63	SRAM_0[27] =	0x19631462	SRAM_0[27] =	0x 66004e4	SRAM_0[27] =	0xffc8fe86	SRAM_0[27] =	0xffd3fe24	SRAM_0[27] =	0xfae7fe0a



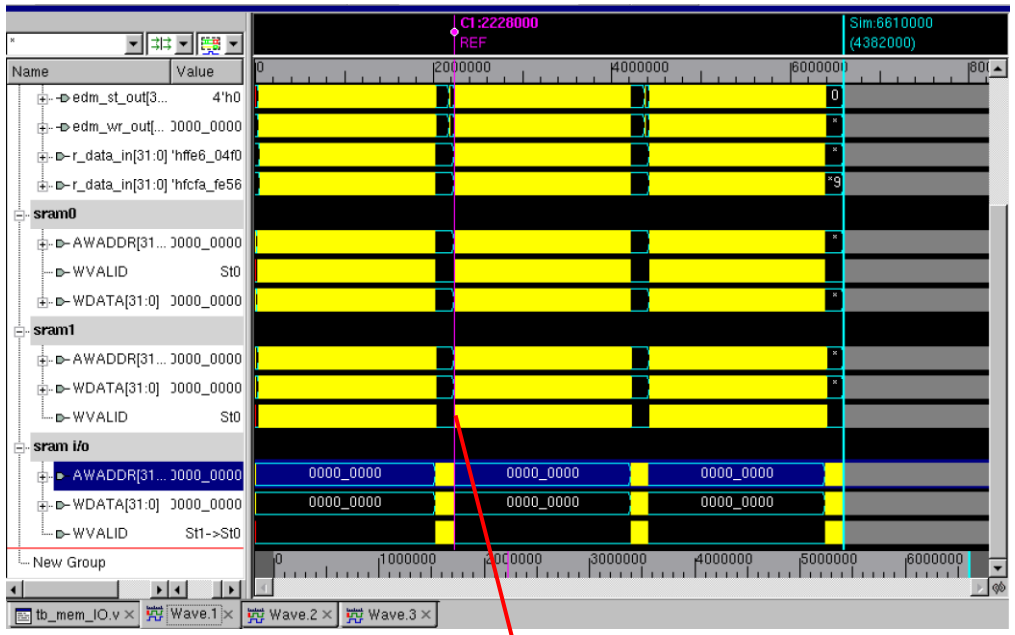
How to Debug (2/7)

- **Debugging Example (Timing analysis)**
  - See the output result (console)

Output 0~63(set0)correct, 64~ wrong..

```
Output count has reached the total number of samples.
Validating bit-accurate output...
    - Mismatch @ output[0x0040]; expected 0xf551fc78, got 0xfd23fa07
    - Mismatch @ output[0x0041]; expected 0x0184f8db, got 0xfe450daf
    - Mismatch @ output[0x0042]; expected 0xfce5fe76, got 0x00f7fbcf
    - Mismatch @ output[0x0043]; expected 0xfe16f4f1, got 0x0169119d
    - Mismatch @ output[0x0044]; expected 0x03a4fd74, got 0xfe7a00ae
    - Mismatch @ output[0x0045]; expected 0x0561fd93, got 0xf522004b
```

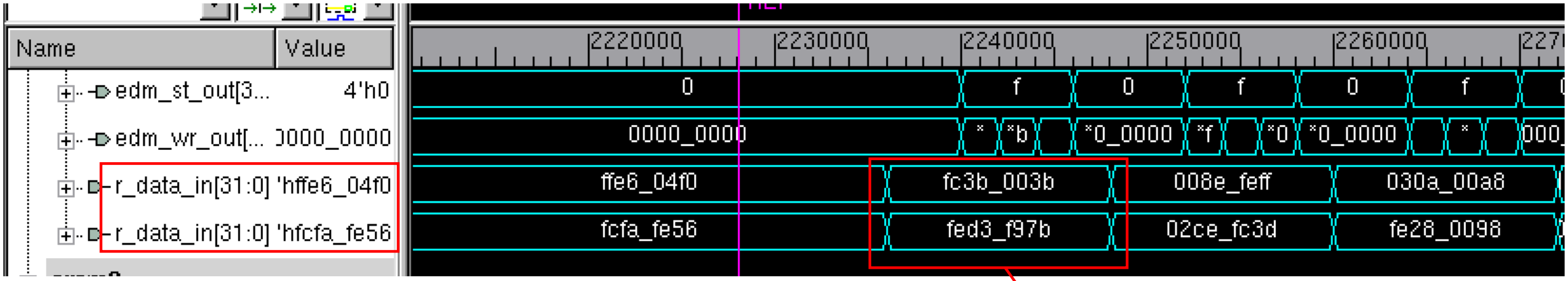
See the set1_stage 0



Set1_stage 0

How to Debug (4/7)

- **Debugging Example (Timing analysis)-cont'd**
  - See the d0, d1 & twiddle factor timing

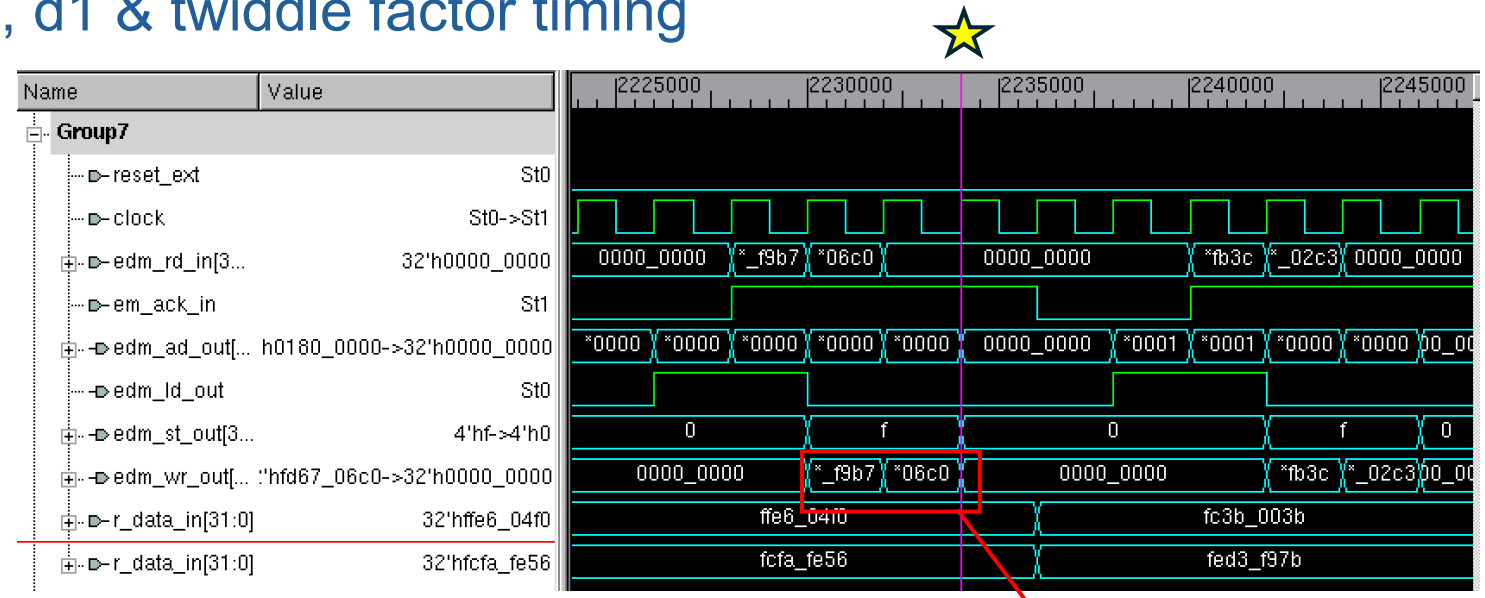


<risv-1 waveform>

First dpath output = wrong

How to Debug (5/7)

- **Debugging Example (Timing analysis)-cont'd**
  - See the d0, d1 & twiddle factor timing

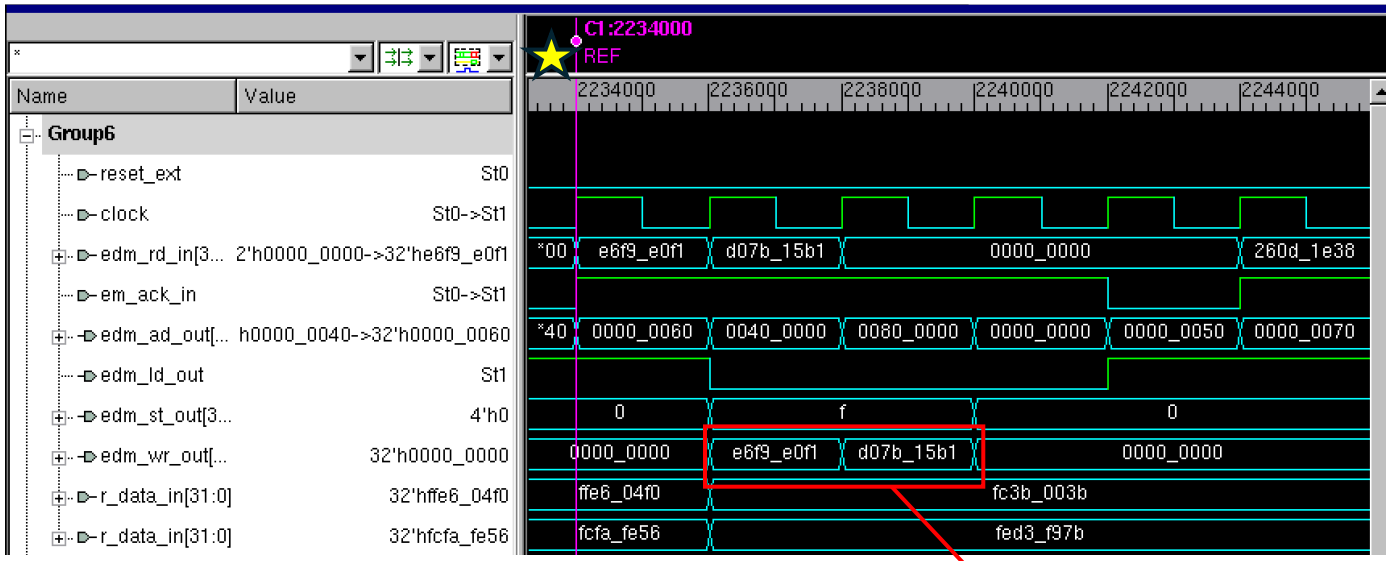


<risv-2 waveform>

First dpath input = wrong

How to Debug (6/7)

- **Debugging Example (Timing analysis)-cont'd**
  - See the input timing



<risv-2 waveform>

Right input is to late
=> **Compute core (risv-1) need stall**

How to Debug (7/7)

KU KONKUK UNIVERSITY

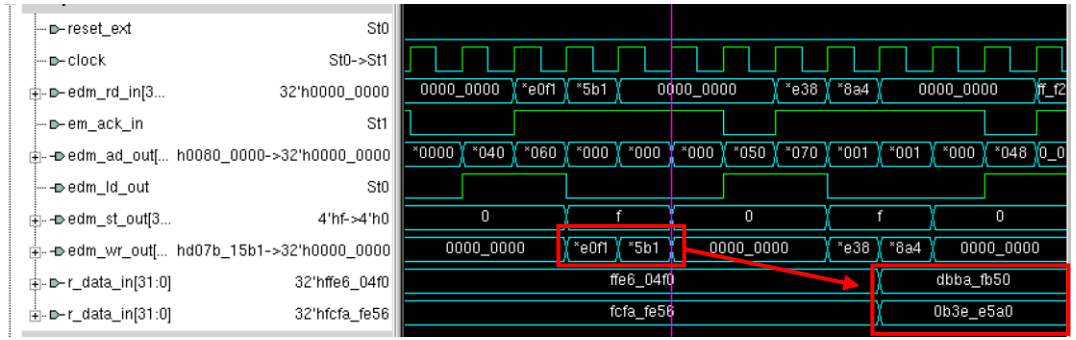
- **Debugging Example (Timing analysis)-cont'd**
  - Add 6 dummy(nop) to data_m2.s and check

```
#butterfly 31
lw s11, 124(s1) #sram_0[31]->s11
sw s11, 248(s7) #s11->sram i/o
addi    x0,     x0,     0x0000      # dummy

addi    x0,     x0,     0x0000      # dummy
addi    x0,     x0,     0x0000      # dummy
addi    x0,     x0,     0x0000      # dummy
addi    x0,     x0,     0x0000      # dummy
addi    x0,     x0,     0x0000      # dummy
addi    x0,     x0,     0x0000      # dummy

# ========== set0 End ==========
```

<data_m2.s>



<risv-1 waveform>

Right timing! => correct result!!

```
Output count has reached the total number of samples.
Validating bit-accurate output...
    - Total mismatches: 0 / 192
Checking SRAM occupancy...
    - SRAM 2: occupancy = 32
    - SRAM 3: occupancy = 32
Test passed. Ending simulation.
$finish called from file "../../tb/tb_mem_IO.v", line 132.
$finish at simulation time              6634000
Simulation complete, time is 6634000 ps.
```

<console>

Ablation Study(1/4)

KU KONKUK UNIVERSITY

- **Pipelining With Memory Flag**
  - Code for Flagging

```
    # Clear sync flags initially
    sw      x0,     0(s2)           # Clear ready flag
    sw      x0,     4(s2)           # Clear done flag

main_loop:
    bge     s5,     s4,     end_infinite

    # Wait for Core 2 to be ready (check if previous output is done)
    beq     s5,     x0,     skip_wait   # Skip wait for first set
wait_core2:
    lw      t0,     4(s2)           # Check done flag from Core 2
    beq     t0,     x0,     wait_core2
    sw      x0,     4(s2)           # Clear done flag
```

  - Execution time & Memory Capacity

```
Output count has reached the total number of samples.
Validating bit-accurate output...
    - Total mismatches: 0 / 192
Checking SRAM occupancy...
    - SRAM 2: occupancy = 64
    - SRAM 3: occupancy = 3
Test passed. Ending simulation.
$finish called from file "../../tb/tb_mem_IO.v", line 132.
$finish at simulation time              60022000
Simulation complete, time is 60022000 ps.
```

■ Performance

- Simulation Time : 60,022,000[ps]

■ Memory Capacity

- Used Memory: SRAM 0: 64(128B)
- Used Memory: SRAM 1: 3(128B)

SRAM 1 for Flags

Ablation Study(2/4)

KU KONKUK UNIVERSITY

▪ Parallel Operations /w TP_1 code

- Reuse TP_1 code & dummy for Sync

```
# reg에 Twiddle Factor을 담아두고, 재사용하려한다.
lw      a7, 0(s3)           # a7 = twiddle[0]
lw      a6, 64(s3)          # a6 = twiddle[16]

# i = 0
# n = 0
# stage[0][0]

# stage[0][0]의 연산 결과를 stage[0][1]로 전달하는데
# SRAM 0의 sw/lw 과정을 제하기위해 가용한 reg를 최대한 사용하였다.

# Input
# 아래는 DIT 형식에 맞추어 SRAM I/O으로부터의 Input을 받는 과정이다.
# e.g. x[0], x[32], x[16]...순서...
lw      t0, 0(s0)
lw      t1, 128(s0)
lw      t2, 64(s0)
lw      t3, 192(s0)
lw      t4, 32(s0)
lw      t5, 160(s0)
lw      t6, 96(s0)
lw      a0, 224(s0)
```

- Execution time & Memory Capacity

```
Output count has reached the total number of samples.
Validating bit-accurate output...
    - Total mismatches: 0 / 192
Checking SRAM occupancy...
    - SRAM 2: occupancy = 64
    - SRAM 3: occupancy = 64
    - SRAM 5: occupancy = 64
Test passed. Ending simulation.
$finish called from file "../../tb/tb_mem_IO.v", line 132.
$finish at simulation time                6044000
Simulation complete, time is 6044000 ps.
```

Performance X 10!!

■ Performance

- Simulation Time : 6,044,000[ps]

■ Memory Capacity

- Used Memory: SRAM 0: 64(128B)
- Used Memory: SRAM 1: 64(128B)
- Used Memory: SRAM 2: 64(128B)

Ablation Study(3/4)

KU KONKUK UNIVERSITY

- **Pipelining Cores**

  - code

```
stage0:
    # Initial synchronization - wait for m1 to start loading

    lw      t4, 0(s3)       # W[0](SRAM_W->t4)

    addi    x0, x0,  1      # dummy
    addi    x0, x0,  1      # dummy
    addi    x0, x0,  1      # dummy
    addi    x0, x0,  1      # dummy
    addi    x0, x0,  1      # dummy
    addi    x0, x0,  1      # dummy

    # Load twiddle factors for all stages (32 values)
    # Butterfly 0-31: Load W[0] for stage 0 (all W[0])

    # Butterfly 0
    sw t4, 0(s6)      # W[0](t4->dpath2)
    lw t5, 4(s4)      # o_d0(dpath0->t5)
    lw t6, 4(s5)      # o_d0(dpath1->t6)
    sw t5, 0(s1)      # o_d0(t5->SRAM_0)
    sw t6, 0(s2)      # o_d0(t6->SRAM_1)
```

Memory X 3!!

  - Execution time & Memory Capacity

```
Output count has reached the total number of samples.
Validating bit-accurate output...
    - Total mismatches: 0 / 192
Checking SRAM occupancy...
    - SRAM 2: occupancy = 32
    - SRAM 3: occupancy = 32
Test passed. Ending simulation.
$finish called from file "../../tb/tb_mem_IO.v", line 132.
$finish at simulation time              6804000
Simulation complete, time is 6804000 ps.
```

■ Performance

- Simulation Time : 6,804,000[ps]

■ Memory Capacity

- Used Memory: SRAM 0: 32(128B)
- Used Memory: SRAM 1: 32(128B)

Ablation Study(4/4)

KU KONKUK UNIVERSITY

- **Pipeling Cores (Optimized)-final version**

  - code

```
stage0:
    # Initial synchronization - wait for m1 to start loading

    lw      t4, 0(s3)       # W[0](SRAM_W->t4)

    addi    x0, x0,  1      # dummy
    addi    x0, x0,  1      # dummy
    addi    x0, x0,  1      # dummy
    addi    x0, x0,  1      # dummy
    addi    x0, x0,  1      # dummy
    addi    x0, x0,  1      # dummy

    # Load twiddle factors for all stages (32 values)
    # Butterfly 0-31: Load W[0] for stage 0 (all W[0])

    # Butterfly 0
    sw t4, 0(s6)      # W[0](t4->dpath2)
    lw t5, 4(s4)      # o_d0(dpath0->t5)
    lw t6, 4(s5)      # o_d0(dpath1->t6)
    sw t5, 0(s1)      # o_d0(t5->SRAM_0)
    sw t6, 0(s2)      # o_d0(t6->SRAM_1)
```

  - Execution time & Memory Capacity

```
Output count has reached the total number of samples.
Validating bit-accurate output...
    - Total mismatches: 0 / 192
Checking SRAM occupancy...
    - SRAM 2: occupancy = 32
    - SRAM 3: occupancy = 32
Test passed. Ending simulation.
$finish called from file "../../tb/tb_mem_IO.v", line 132.
$finish at simulation time              6634000
Simulation complete, time is 6634000 ps.
```

Performance X 1.1 !!

■ Performance

- Simulation Time : 6,634,000[ps]

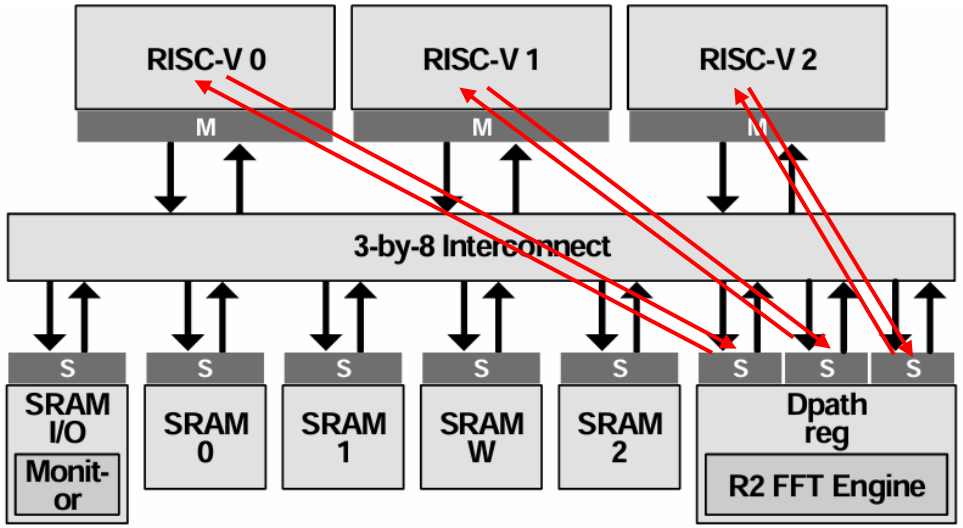
■ Memory Capacity

- Used Memory: SRAM 0: 32(128B)
- Used Memory: SRAM 1: 32(128B)

Roofline Model(1/4)

KU KONKUK UNIVERSITY

- **System Analysis**
  - Roofline analysis



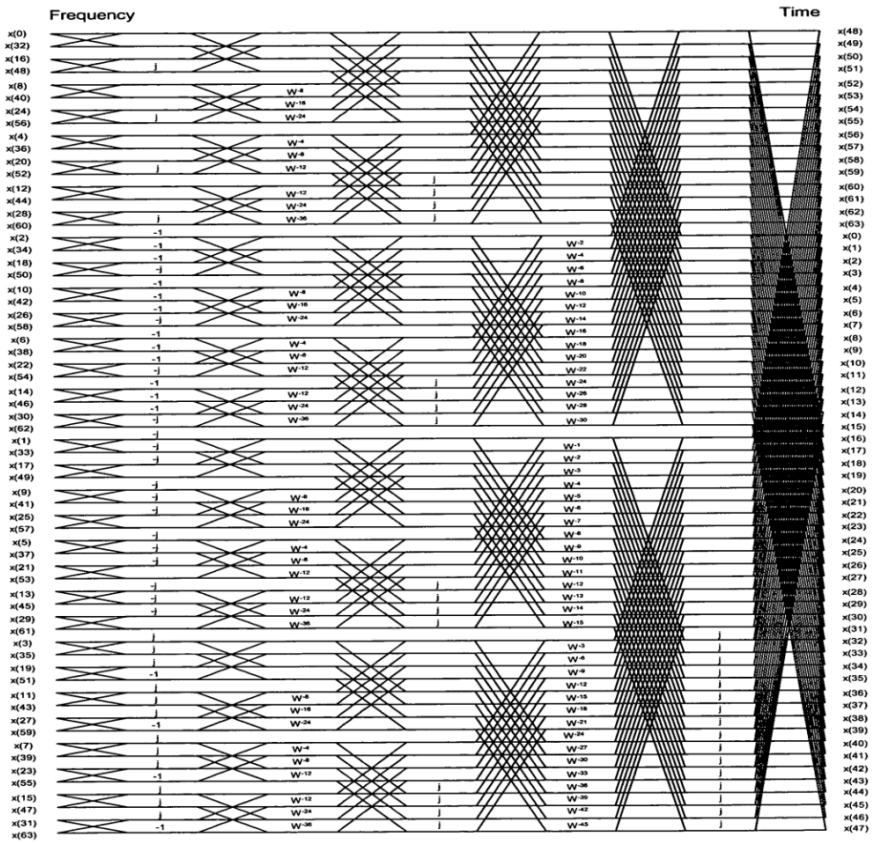
Y axis (perf) : BF/cycle
X axis (IO) : BF/BYTE

Performance limit : 1BF/cycle
BW_m (Slope) : 12Byte/cycle (3core)

Roofline Model(2/4)

KU KONKUK UNIVERSITY

- **Compute Performance (BF/Cycle)**



32BF/stage
6stage/1set
32*6=192BF/set
(192BF/set)*3set=**576BF**

2000ps/1cycle

- Pipelining with memory flag

(60022000 ps)*(1cycle/2000ps)=30,011cycle
576BF/30,011cycle = **0.019 [BF/cycle]**

- Parallel Operations /w TP_1 code

(6044000 ps)*(1cycle/2000ps)=3,022cycle
576BF/3,022cycle = **0.191 [BF/cycle]**

- Pipelining Cores

(6804000 ps)*(1cycle/2000ps)=3,402cycle
576BF/3,402cycle = **0.169 [BF/cycle]**

- Final Ver.

(6634000 ps)*(1cycle/2000ps)=3,317cycle
576BF/3,317cycle = **0.174 [BF/cycle]**

Roofline Model(3/4)

KU KONKUK UNIVERSITY

- **Compute IO (BF/Byte)-cont'd**

1BF => 3LW+3SW (input), 2LW+2SW(output)

IO = 1BF/(10*4)BYTE = 0.025(BF/BYTE)

- Pipelining with memory flag

-Total

Input load : 192*4 Byte
Flag check : (6*3*4)*576Byte (plag poling)
Compute : 10*4*576 Byte
Output store : 192*4 Byte

IO = 576BF / (192*4 + 6*3*4*576 +
10*4*576 +192*4)
= 576BF / 66048Byte
= 0.0087(BF/BYTE)

- Parallel Operations /w TP_1 code

-Total

Input load : 192*4 Byte
Compute(d1,d2) : 8*4*576 Byte
Compute(W) : (1+1+2+4+8+24+576)*4Byte
Output store : 192*4 Byte

IO = 576BF / 22432Byte
= 0.0257(BF/BYTE)

Roofline Model(4/4)

KU KONKUK UNIVERSITY

▪ Compute IO (BF/Byte)-cont'd

- Pipelining Cores & Final ver.

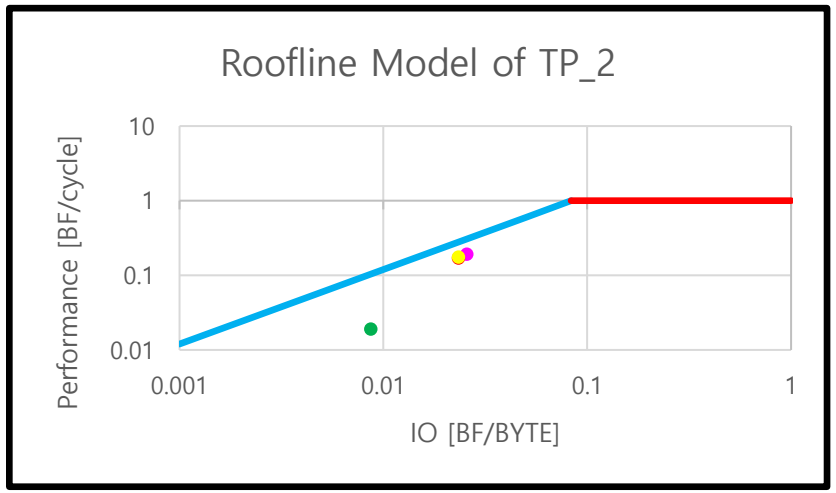
-Total

Input load : 192*4 Byte
Compute : 10*4*576 Byte
Output store : 192*4 Byte

IO = 576BF / 24576 Byte
= 0.0234(BF/BYTE)

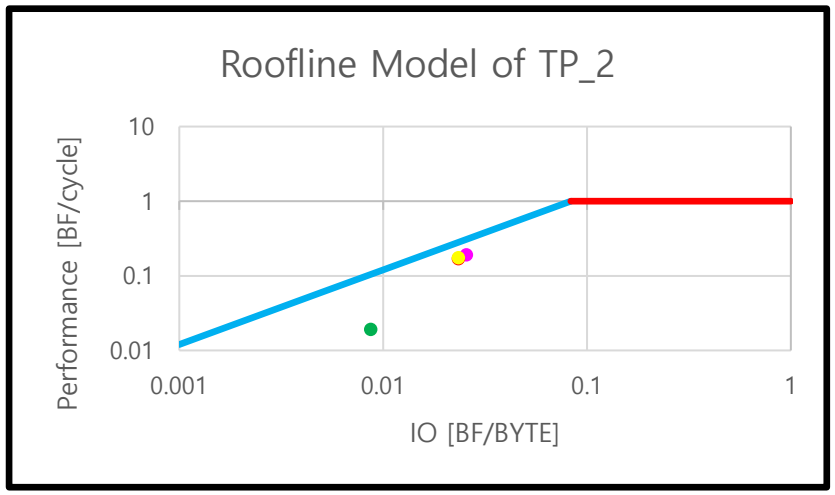
▪ Draw roofline model /w Excel

	A	B	C	D	E	F	G
1		BF/byte	Performance(BF/cycle)	Memory line	Compute line	X_roofline	Y_roofline
2	Pipelining with memory flag	0.0087	0.019	0.1044	1	0.001	0.012
3	Parallel Operations /w TP_1 cc	0.0257	0.191	0.3084	1	0.005	0.06
4	Pipelining Cores	0.0234	0.169	0.2808	1	0.01	0.12
5	Final ver.	0.0234	0.174	0.2808	1	0.05	0.6
6						0.0833	1
7						0.1	1
8						0.5	1
9						1	1



Our system is Memory Bound !!

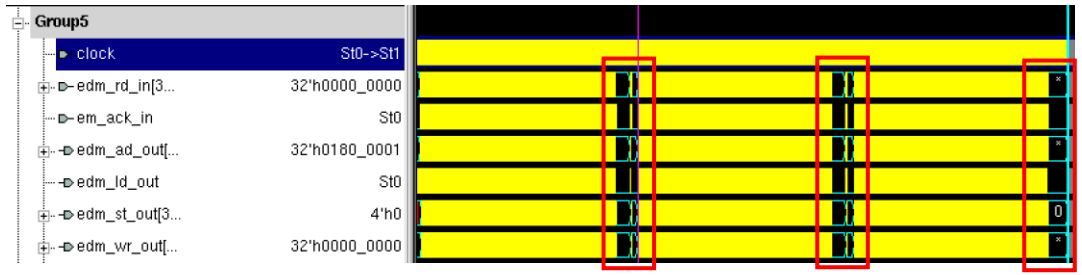
How to improve



Our system is Memory Bound !!

- Reuse twiddle factor
- Using register
- Etc..

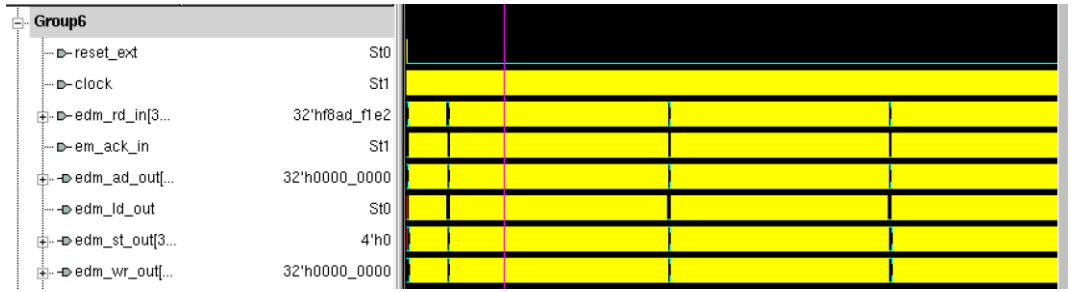
How to improve



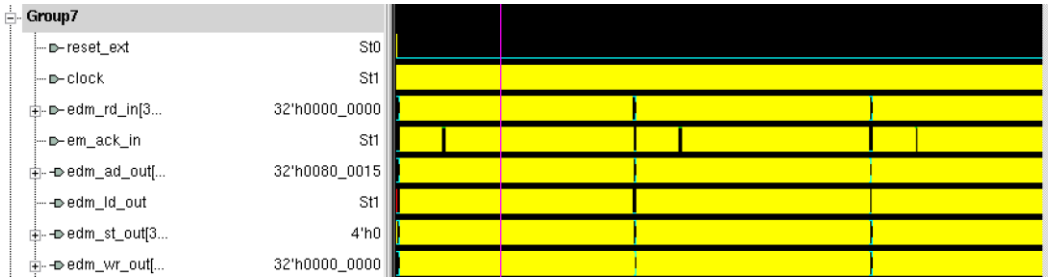
<core-0 waveform>

: core 0 is wasted..

If core 0 do LW & SW Next set input data?



<core-1 waveform>



<core-2 waveform>

core 1, 2 are good!!

Summary

KU KONKUK UNIVERSITY

- **Compare with TP_1**
  - 3 cores vs Single core

	Performance[ps]	Memory Capacity[B/B]
Single Core (TP_1)	14,190,000[ps]	256B/256B
3 Cores (TP_2)	6,634,000[ps]	256B/768B