2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

PSS action sequence modeling using Machine Learning

Moonki Jang , Samsung Electronics co., Ltd

SAMSUNG

accellera SYSTEMS INITIATIVE

Agenda

- Background of Motivation
- Introduction of PSS sequence model
- Machine learning implementation
- CASE STUDY : PCIe deadlock condition reproduction
- Conclusion

- Why is deadlock verification mainly done in the silicon level?
- Due to deadlocks found at the silicon level:
  - Causes performance degradation due to SW Workaround
  - Huge expense for mask revision
- For these reasons, we've been find way to verify deadlock on the early stage of pre silicon level

- Why is deadlock verification difficult at the pre silicon level?
  - A deadlock is caused by a combination of certain conditions
  - However, it is difficult to reproduce the conditions and combinations
  - Random test is not suitable for pre silicon

- A different deadlock verification methodology is needed that utilizes the advantages of pre-silicon level only.

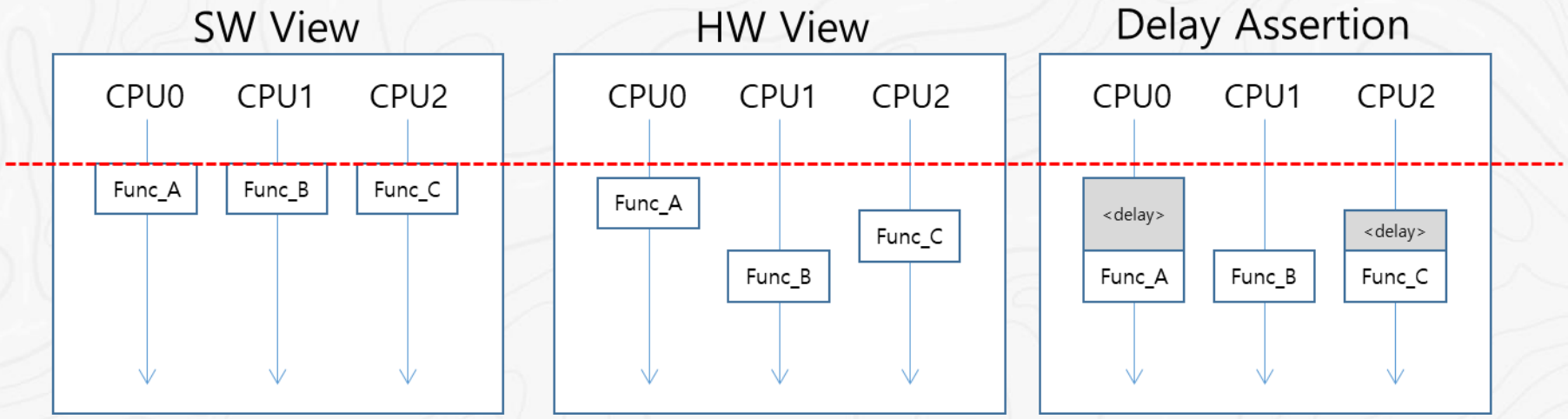
- At the pre silicon level, internal signals can be monitored.
  - This means that the target condition can be created rather than waiting for it to occur.
  - It is not easy to simultaneously generate different hardware events from the CPU running through software.

- We will show you how we made it possible in this study.

Agenda

- Background of Motivation
- Introduction of PSS action sequence model
- Machine learning implementation
- CASE STUDY : PCIe deadlock condition reproduction
- Conclusion

- Even if SW is executed at the same time, the resulting HW event occurs differently.



- We had to insert delay to make synchronized HW events.

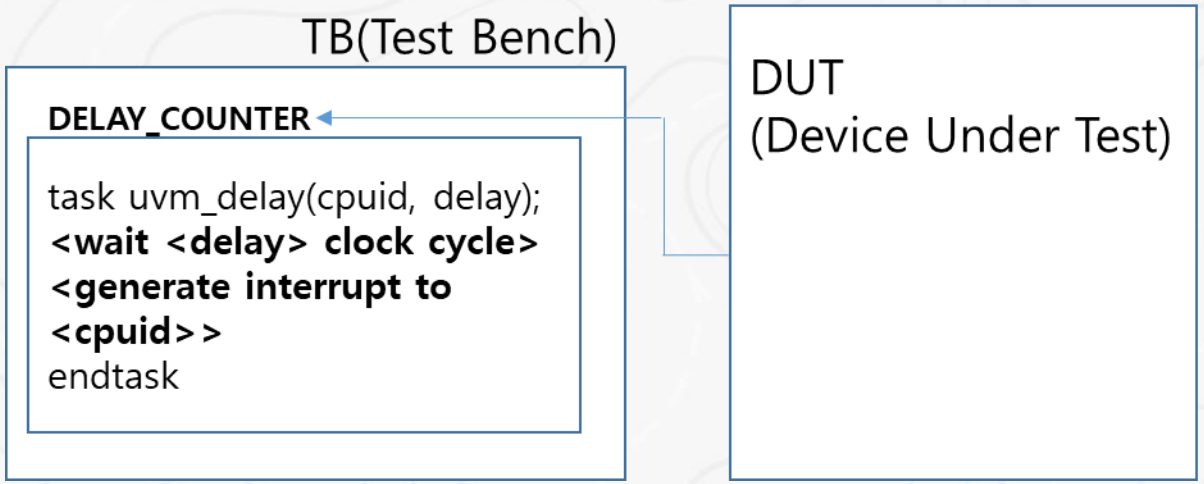
- Problem of traditional SW delay

```
//Loop delay
…
i=100;
While(i--);
…
```

```
//NOP instruction
…
NOP;
NOP;
…
```

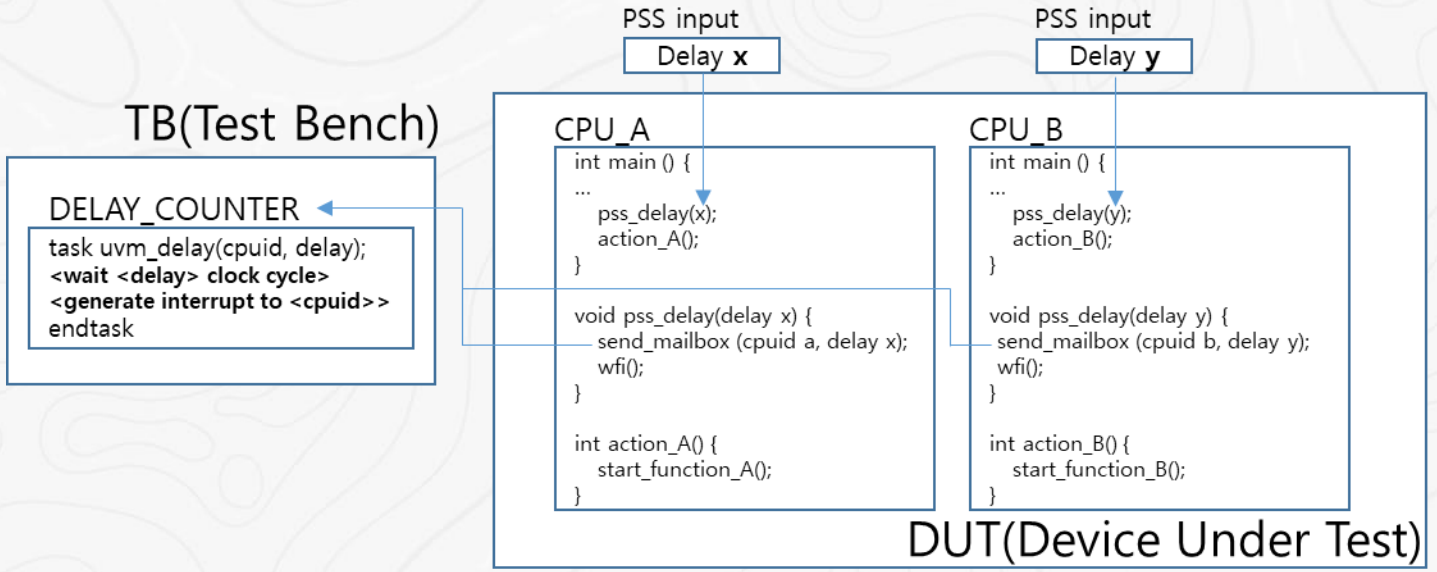
- We need delay that has linearity and cycle accuracy resolution

- Introduction of Delay counter

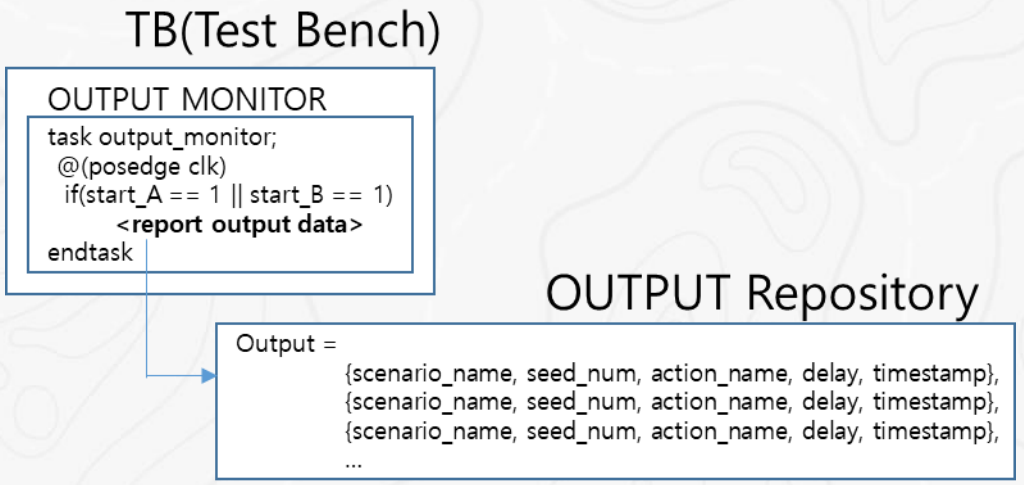


- We can create a linear and high-resolution delay through above DELAY_COUNTER.

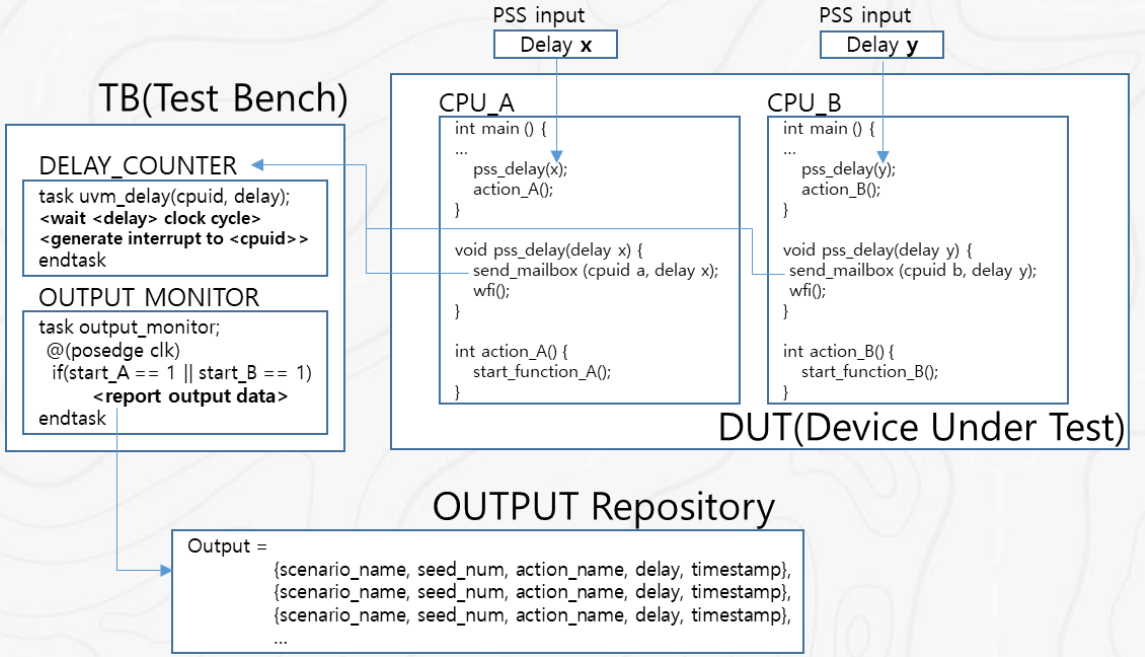
- PSS test generation
  - For action synchronization, PSS delay is added before the target action.
  - PSS delay has configurable clock cycle delay



- Output monitor and output repository
  - The output monitor checks whether a preset target condition has occurred.
  - The output repository is a space where the information output by the output monitor is collected



- Complete structure of PSS action sequence model

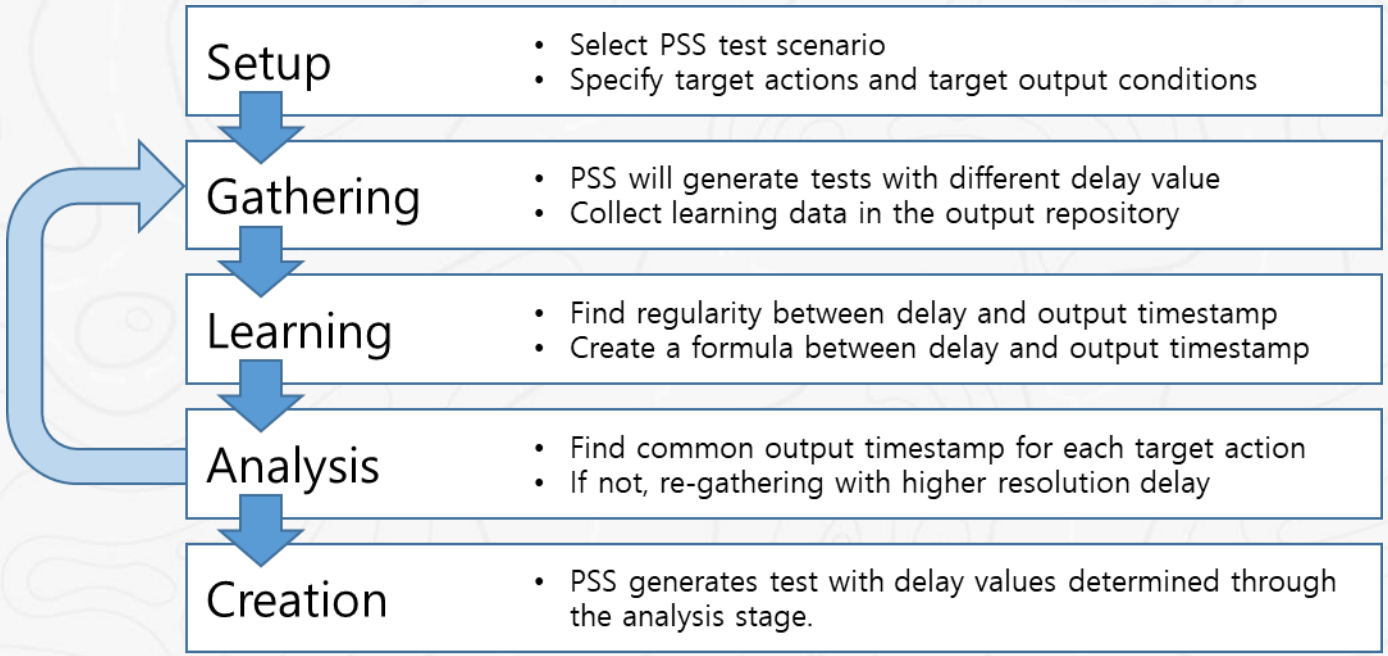


- We've implement Machine learning algorithm for finding proper delay value x and y

Agenda

- Background of Motivation
- Introduction of PSS sequence model
- Machine learning implementation
- CASE STUDY : PCIe deadlock condition reproduction
- Conclusion

- ML sequence modeling flow

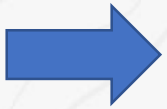


- Setup / Gathering stage
- In the Setup stage
  - Selects the scenario to be used for test creation
  - Selects the target action to control the sequence
  - Set the pss_delay value

- In the Gathering stage
  - the PSS tool creates tests and performs simulations according to configured batch mode script
  - After running the generated tests, store the output monitor's result report in the output repository.

- Learning stage
  - We can create coordinates with delay and timestamp values
  - The goal of Learning stage is to obtain the following linear equation by identifying the tendency between each coordinate

Action A	Action B
(0, 1)	(0, 2)
(1, 3)	(1, 5)
(2, 5)	(2, 8)

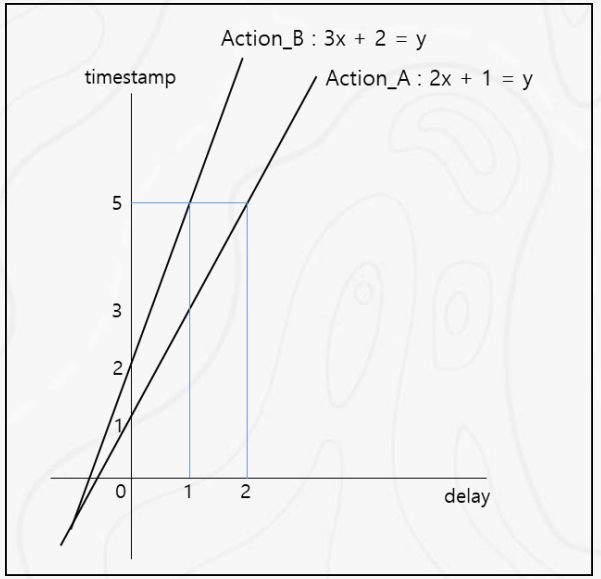


Action_A : $2X_1 + 1 = Y$ (X_1,X_2: delay)
Action_B : $3X_2 + 2 = Y$ (Y : Output timestamp)

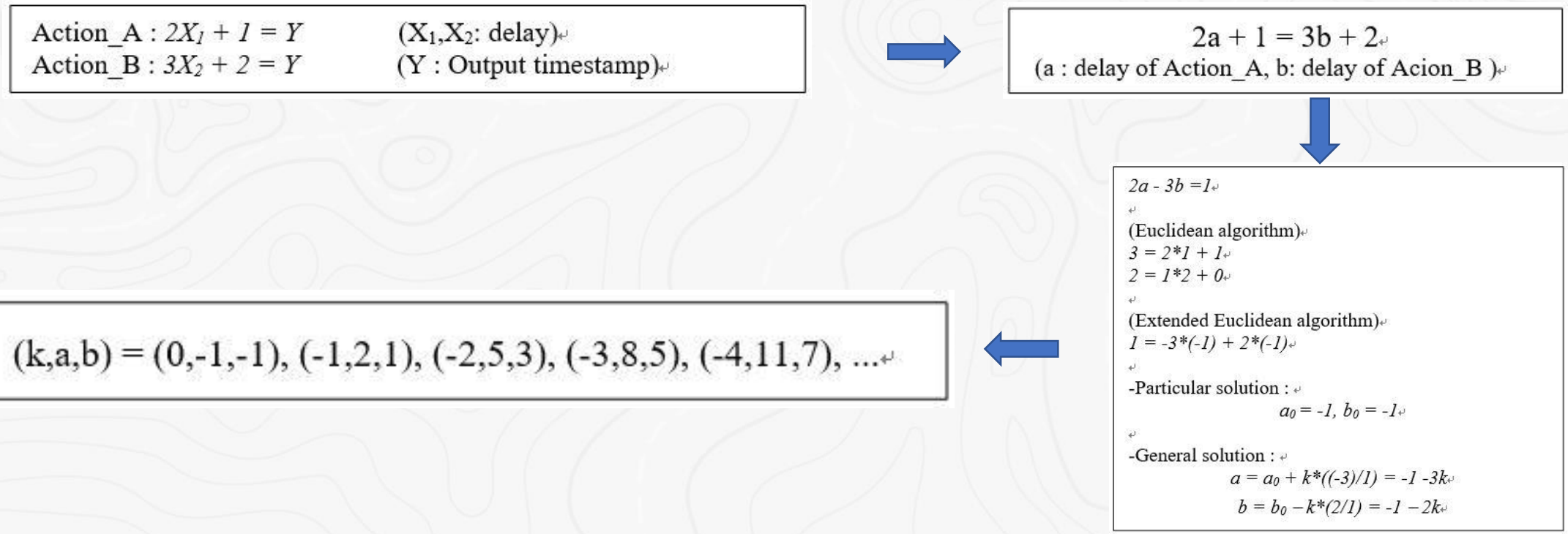
ax + b = y, a (slope) = increment of timestamp / increment of delay

- Analysis stage
  - Uses the formula created in the learning stage to find the delay value where the output condition of the target actions occurs simultaneously

Action_A : $2X_1 + 1 = Y$	(X_1,X_2: delay)
Action_B : $3X_2 + 2 = Y$	(Y : Output timestamp)



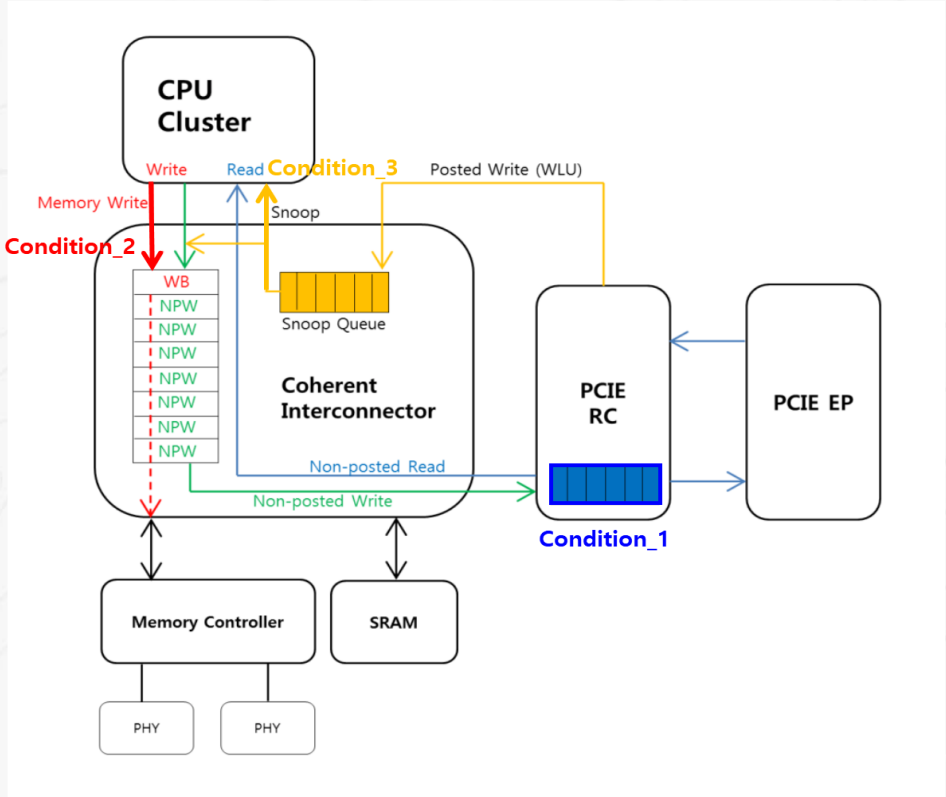
- Analysis stage
  - We can find the possible combinations of X1 and X2 using the Extended Euclidean algorithm as follows. (refer to the paper for details)



Agenda

- Background of Motivation
- Introduction of PSS sequence model
- Machine learning implementation
- CASE STUDY : PCIe deadlock condition reproduction
- Conclusion

- PCIe deadlock case
  - PCIE deadlock is caused by protocol conflict between PCIE and ACE bus interconnector



- PCIE deadlock appears when the following conditions occur simultaneously
  - Condition_1 : Generate PCIE RC buffer full (generated by Action_1)
  - Condition_2 : Generate Writeback transaction (Action_2)
  - Condition_3 : Snoop generated from posted write of PCIE (Action_3)

- Target condition of each action that described in target output file

Action_1	top.dut.BLK_HSIO.xxx.pcie_rc_buf_full ==1
Action_2	top.dut.BLK_CPUCL.xxx.AWVALID && top.dut.BLK_NOCL0.xxx.AWREADY && top.dut.BLK_CPUCL.xxx.AWSNOOP[2:0]=='0b011'
Action_3	top.dut.BLK_NOCL0.xxx.ACVALID && top.dut.BLK_CPUCL.xxx.ACREADY && top.dut.BLK_NOCL0.xxx.ACSNOOP[3:0]=='0b1101'

- Gathering stage
  - Tests have been generated by increasing the pss_delay value one step by one from 0 to 100 using the PSS tool
  - learning data stored in the output repository is as shown in the figure below

```
//scenario,seed_num,action,delay,timestamp
{pcie_np_wr_pw_ml,  8875, cpu_writeback, 0, 874352},
{pcie_np_wr_pw_ml,  8897, pcie_ep_mem_write, 0, 874354},
{pcie_np_wr_pw_ml,  8964, pcie_config_write_rc_full, 0, 1496546},
{pcie_np_wr_pw_ml,  9324, cpu_writeback, 1, 874354},
{pcie_np_wr_pw_ml,  9357, pcie_ep_mem_write, 1, 874357},
{pcie_np_wr_pw_ml,  6853, pcie_config_write_rc_full, 1, 1496554},
{pcie_np_wr_pw_ml,  9874, cpu_writeback, 2, 874356},
{pcie_np_wr_pw_ml,  10543, pcie_ep_mem_write, 2, 874360},
{pcie_np_wr_pw_ml,  13780, pcie_config_write_rc_full, 2, 1496564},
{pcie_np_wr_pw_ml,  3543, cpu_writeback, 3, 874358},
{pcie_np_wr_pw_ml,  3876, pcie_ep_mem_write, 3, 874363},
{pcie_np_wr_pw_ml,  8423, pcie_config_write_rc_full, 3, 1496572},
{pcie_np_wr_pw_ml,  6980, cpu_writeback, 4, 874360},
{pcie_np_wr_pw_ml,  7005, pcie_ep_mem_write, 4, 874366},
{pcie_np_wr_pw_ml,  12098, pcie_config_write_rc_full, 4, 1496580},
...
```

- Learning stage
  - The learning data collected in the output repository is organized in the form of coordinates for each action as follows for linear regression analysis

Action 1	Action 2	Action 3
(0, 1496546)	(0, 874352)	(0, 874354)
(1, 1496554)	(1, 876154)	(1, 874357)
(2, 1496564)	(2, 874356)	(2, 874360)
(3, 1496572)	(3, 874358)	(3, 874363)
…	…	…
$y=8x_1+1496546$	$y=2x_2+874352$	$y=3x_3+874354$

- Analysis stage
  - Using the Extended Euclidean algorithm as shown below

- Linear equation of Action_2 : $y=2x_2+874352$
- Linear equation of Action_3 : $y=3x_3+874354$
- Particular solution : $X_2 = 4, X_3 = 2$
- General solution (k : integer value)

$$X_2 = 4 - 3k$$
$$X_3 = 2 - 2k$$

* If k has a value of -103699:

$$X_2 = 4 - 3*(-103699) = 311101$$

$$X_3 = 2 - 2*(-103699) = 207400$$

$$Action\_2: y=2x311101 + 874352 = 1496554$$
$$Action\_3: y=3x207400 + 874354 = 1496554$$

  - $pss\_delay_{action\_1} = 1$, $pss\_delay_{action\_2} = 311101$, and $pss\_delay_{action\_3} = 207400$

Agenda

- Background of Motivation
- Introduction of PSS sequence model
- Machine learning implementation
- CASE STUDY : PCIe deadlock condition reproduction
- Conclusion

Conclusion

- Through this study:
  - We were able to discover new possibilities of PSS through Machine Learning
  - We enable deadlock verification that was considered impossible at the simulation level.
    - It should be possible to predict the risk expected condition through the risk assessment process.

- Question and Answer
  - Please feel free to contact me (moonki.jang@gmail.com)