NYCU-EE IC LAB – Fall 2024

Lab08 Exercise

Design: Self-attention

Data Preparation

1. Extract files from TA's directory:

   % tar xvf ~iclabTA01/Lab08.tar

2. The extracted LAB directory contains:
   a. **EXERCISE/**
   b. **EXERCISE_wocg/**
   c. **PRACTICE/**
   d. **JG/**

Design Description

Self-attention is a mechanism used in neural networks, particularly in transformer models, to determine the importance of each part of an input sequence when making predictions. Scaled dot-product attention projects input matrices (Query (Q), Key (K), and Value (V)), computes scaled dot-product attention scores, applies the Softmax function to these scores, and generates a context matrix, allowing the model to focus on relevant parts of the input sequence.

In this lab, you are tasked with designing a simple hardware accelerator for Self-attention. The accelerator should efficiently handle matrix operations for input sizes of 1x8, 4x8, 8x8. For simplicity, the Softmax function has been replaced with ReLU. Moreover, you should implement clock gating to save power.

- **Self-attention**

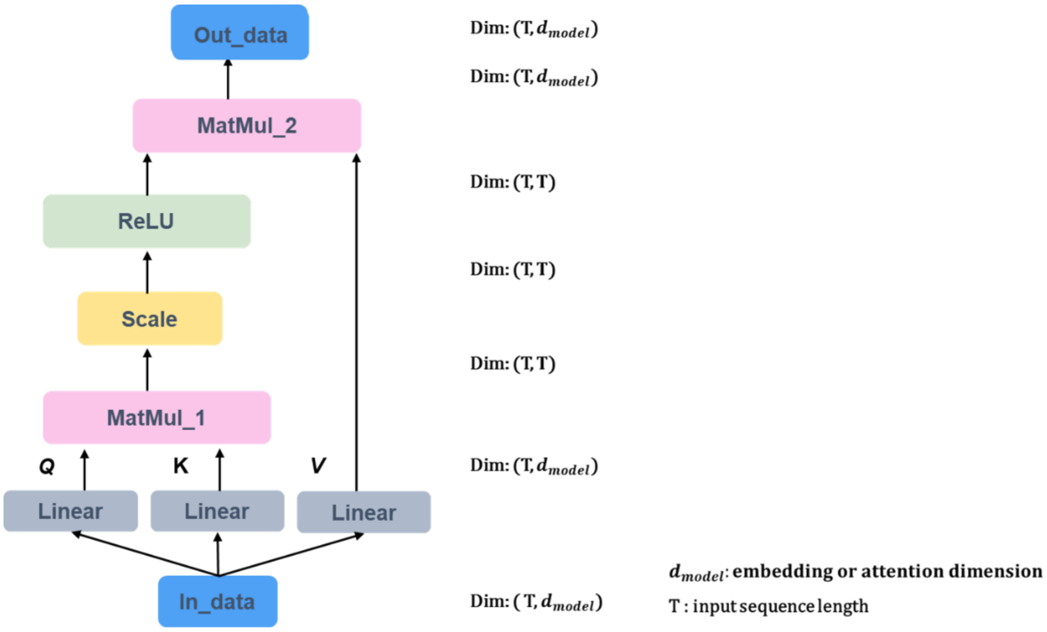


Fig1. Self-attention

Formula of QKV Generation

$$Q = XW_Q\ , K = XW_K, V = XW_V$$

Formula of MatMul_1

$$A\ =\ QK^T$$

Formula of Scale

$$scale\ =\ \frac{QK^T}{3}$$

Formula of ReLU

ReLU (Rectified Linear Unit) is a simple activation function used in neural networks. It returns the input value if it's positive, and outputs zero if the input is negative.

$$\mathrm{ReLU}(x_i)\ =\ \max\,(0, x_i)$$

$$S = ReLU(\frac{QK^T}{3})$$

Formula of MatMul_2

$$P = ReLU(\frac{QK^T}{3})V$$

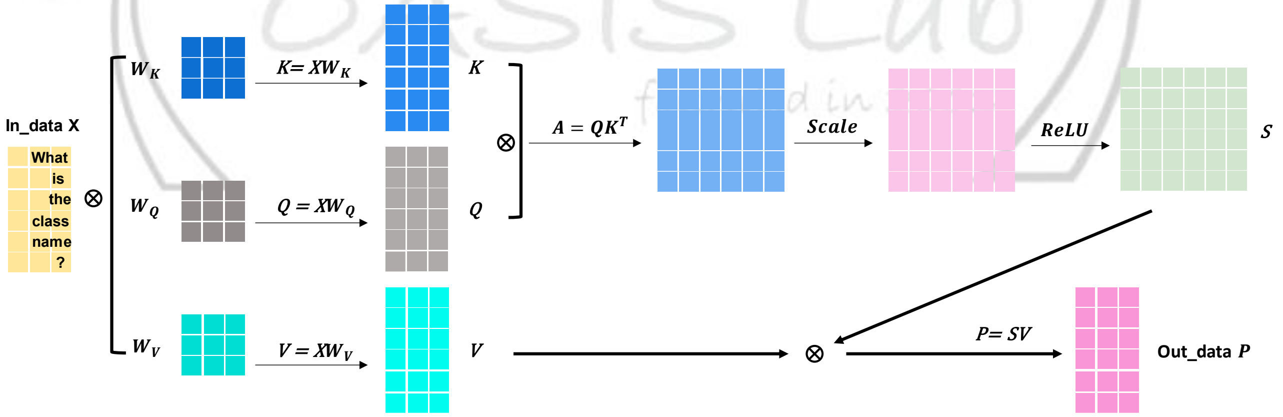


Fig2. Example of Self-attention

Original Formula of Scaled dot-product attention

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_{model}}})V$$

Note: All of the computations mentioned above must ensure that their process and results do not overflow. To be simpler, Softmax function is changed to ReLU and scaling factor is changed from d_{model} to 3.

Description of input signals and output signals

When **in_valid** is high, **T** will be received in the first cycle, and the 8-bit signed **in_data** signal will receive **T*8 cycles** continuously to represent input tokens. First, $\boldsymbol{W_Q}$ runs for 64 cycles, followed by $\boldsymbol{W_K}$ for 64 cycles, and finally $\boldsymbol{W_V}$ for 64 cycles. Each 8-bit signed $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$, $\boldsymbol{W_V}$ signal receives **64** continuous cycles as weights to generate matrix **Q, K, and** V. After all weights have been provided (**192** cycles), **in_valid** will be pulled low. Note that the input signals **in_data,** $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$ **and** $\boldsymbol{W_V}$ are all sent in raster scan order. After completing all process, **out_data** should be outputted **T*8 cycles** in raster scan order when **out_valid** is high.

Lab Hint

In this lab, you need to design two version of module.

- **Stage 1**

  At EXERCISE_wocg/**SA_wocg.v**, which means SA without clock gating, you should design a SA module as below figure.

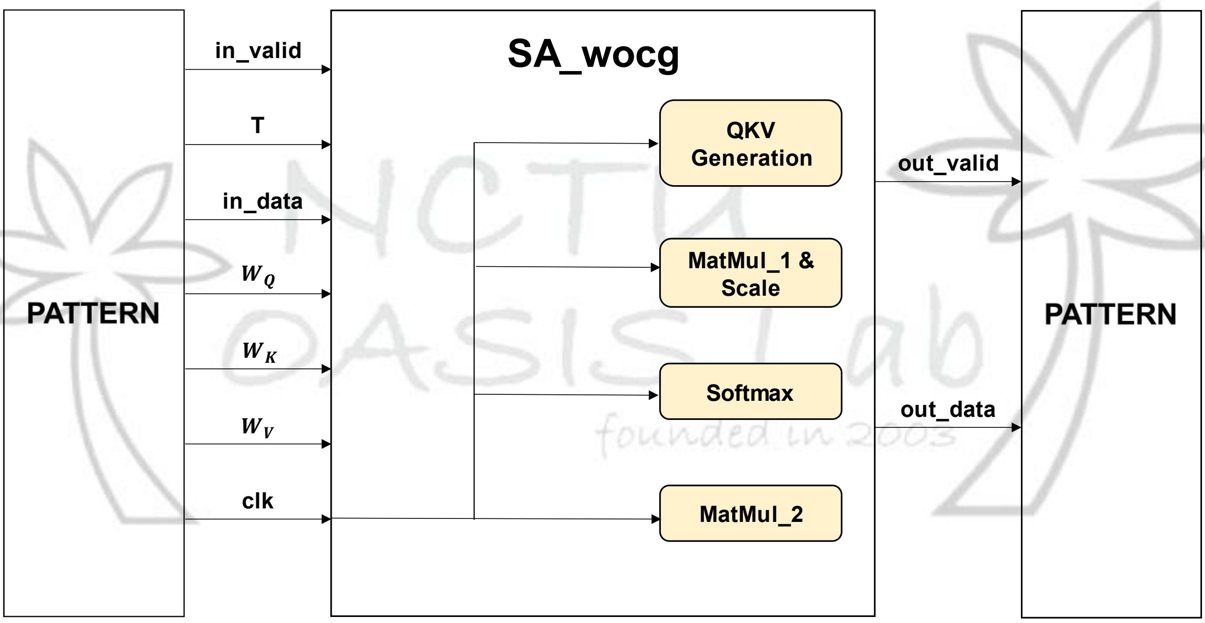


Fig3. Self-attention without clock gating

- **INPUT:** Receive **T, in_data,** $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$ **and** $\boldsymbol{W_V}$ from PATTERN
  - → The [3:0] **T** will be given in the first cycle.
  - → The [7:0] **in_data** will be given T*8 cycles.
  - → The [7:0] $\boldsymbol{W_Q}$ will be given 8*8 = 64 cycles.
  - → The [7:0] $\boldsymbol{W_K}$ will be given 8*8 = 64 cycles.
  - → The [7:0] $\boldsymbol{W_V}$ will be given 8*8 = 64 cycles.
- **SA_wocg:** Design module with some submodules performs series processing described as above.
- **OUTPUT:** Output resulting [63:0] **out_data**.

- **Stage 2**

At EXERCISE/**SA.v**, you should design a **SA module with clock gating cell** as below figure. Main different part is clock gating cell and cg_en input signal.

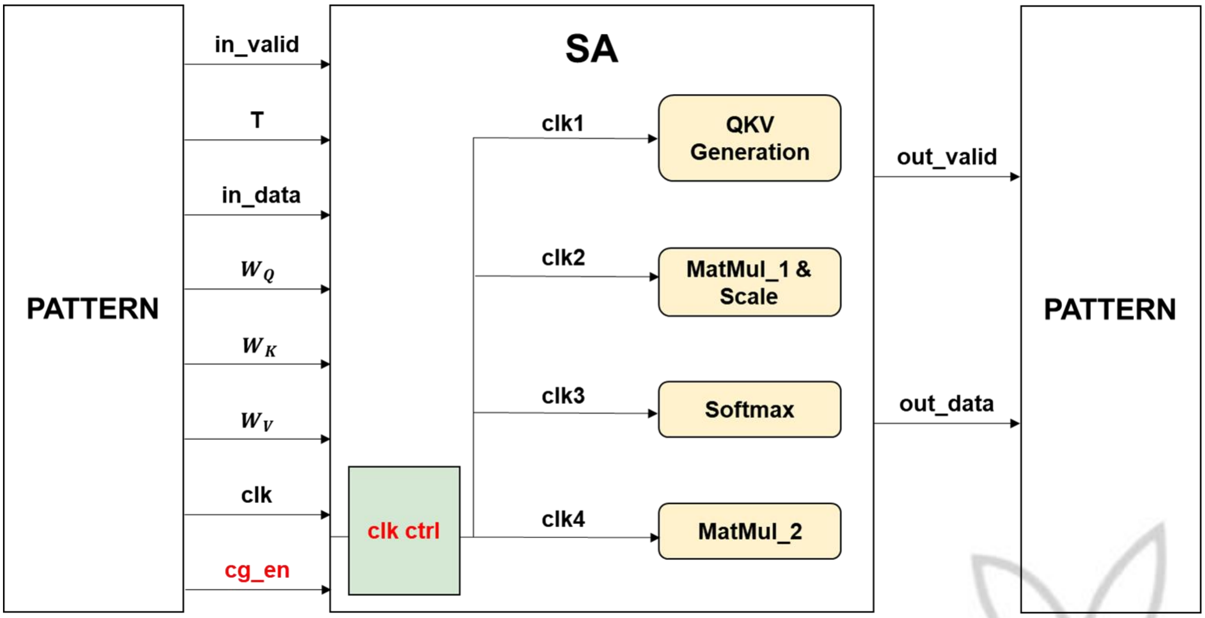


Fig5. Self-attention with clock gating

- **INPUT:** Receive **T, in_data,** $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$, $\boldsymbol{W_V}$ **and cg_en** from PATTERN
  → The [3:0] **T** will be given in the first cycle.
  → The [7:0] **in_data** will be given T*8 cycles.
  → The [7:0] $\boldsymbol{W_Q}$ will be given 8*8 = 64 cycles.
  → The [7:0] $\boldsymbol{W_K}$ will be given 8*8 = 64 cycles.
  → The [7:0] $\boldsymbol{W_V}$ will be given 8*8 = 64 cycles.
- **SA:** Design module. You should add clock gating cell in the design module.
  If **cg_en** is high, 4 processing blocks can perform clock gating; otherwise, if **cg_en** is low, 4 processing blocks follow **clk**.
- **OUTPUT:** Output resulting [63:0] **out_data**.

Inputs and Outputs

I/O	Signal Name	Bit Width	Description
Input	**clk**	1	Clock
Input	**rst_n**	1	Asynchronous active low reset
Input	**cg_en**	1	If **cg_en** is high, the series processing blocks should execute clock gating. Otherwise, if **cg_en** is low, the processing blocks follow **clk**.
Input	**in_valid**	1	High when input signals are valid.

Input	**T**	4	The input sequence length **T** will be given in the first cycle when **in_valid** is high. **To be simpler, T = 1, 4, 8**
Input	**in_data**	8	**in_data** is valid when **in_valid** is high. The **in_data** will be given in T*d_{model} cycles continuously in raster scan order. To be simpler, d_{model} always equals 8. The value are signed integers. (-128~127)
Input	$\boldsymbol{W_Q}$	8	$\boldsymbol{W_Q}$ is valid when **in_valid** is high. The $\boldsymbol{W_Q}$ will be given in 8*8 = 64 cycles continuously in raster scan order. The value are signed integers. (-128~127)
Input	$\boldsymbol{W_K}$	8	$\boldsymbol{W_K}$ is valid when **in_valid** is high. The $\boldsymbol{W_K}$ will be given in 8*8 = 64 cycles continuously in raster scan order. The value are signed integers. (-128~127)
Input	$\boldsymbol{W_V}$	8	$\boldsymbol{W_V}$ is valid when **in_valid** is high. The $\boldsymbol{W_V}$ will be given in 8*8 = 64 cycles continuously in raster scan order. The value are signed integers. (-128~127)
Output	**out_valid**	1	Should set to high when your **out_data** is ready.
Output	**out_data**	64	Output the resulting data in raster scan order. **out_data** should be given in T*d_{model} cycles. To be simpler, d_{model} always equals 8.

Specifications

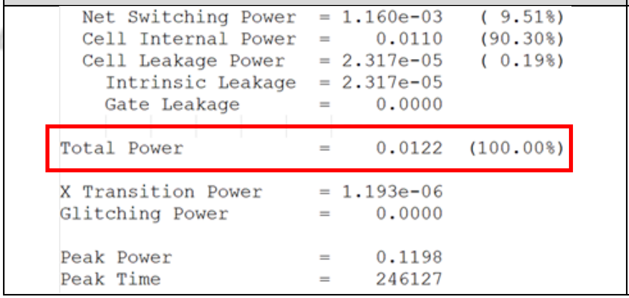
1. Top module name: SA (File name: SA.v)
2. Input pins: clk, rst_n, cg_en, in_valid, [3:0] T, [7:0] in_data, [7:0] W_Q, [7:0] W_K, [7:0] W_V
   Output pins: **out_valid, [63:0] out_data**
3. Use **asynchronous** reset active low architecture.
4. All your output register should be set zero after reset.
5. Changing clock period is prohibited **(fixed at 50ns).**
6. The instance name of the clock gating cell (GATED_OR) should be GATED_XXX (e.g., GATED_cnt).

```
GATED_OR GATED_out (
    .CLOCK(clk), .SLEEP_CTRL(G_sleep_out),
    .RST_N(rst_n), .CLOCK_GATED(G_clock_out)
);
```

7. After synthesis, check the “SA.area” and “SA.timing” in the folder “Report”. The area report is valid only when the slack in the end of “SA.timming” is **non-negative** and the result should be **MET**.

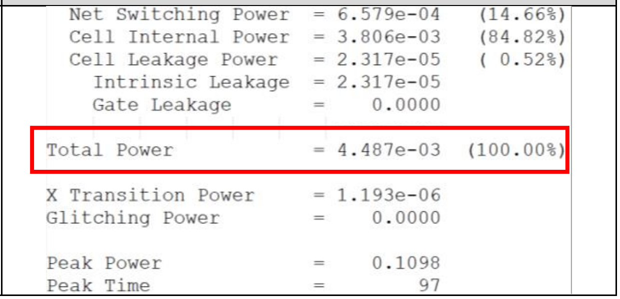
8. The next input will come in **2~5** cycles after your **out_valid** is pulled down.
9. The synthesis result **cannot** contain any **LATCH except for clock gating latch.**
10. The syntheses result **cannot** contain any error.
11. The output loading is set to 0.05.
12. Input delay and output delay are 0.5*Clock Period.
13. You can’t have timing violation in gate-level simulation.
14. You **can’t use memory** in this lab.
15. You **can’t use DesignWare IP** in this lab.
16. The latency is limited in **2000 cycles**.
    latency = 1 + execution latency
17. The **out_valid** cannot overlap with **in_valid**.
18. The **out_valid** should set to high when **out_data** is valid.
19. The **out_valid** should set to low when your **out_data** is invalid.
20. The **out_data** should be correct when **out_valid** is high.
21. The **out_data** should be 0 when **out_valid** is low.
22. Your design should have **at least 10% power reduction** from **cg_en-off** to **cg_en-on**, otherwise it will be regarded as failed. $\frac{P_{cg\_enoff} - P_{cg\_enon}}{P_{cg\_enoff}} \geq 10\%$
23. **Power report position: 04_PTPX/Report/SA_POWER** or **SA_CG_POWER**
    Report Total Power Example:

SA_POWER (w/o clock gating)



```
  Net Switching Power  = 1.160e-03   ( 9.51%)
  Cell Internal Power  =    0.0110   (90.30%)
  Cell Leakage Power   = 2.317e-05   ( 0.19%)
    Intrinsic Leakage  = 2.317e-05
    Gate Leakage       =    0.0000

Total Power            =    0.0122  (100.00%)

X Transition Power     = 1.193e-06
Glitching Power        =    0.0000

Peak Power             =    0.1198
Peak Time              =    246127
```

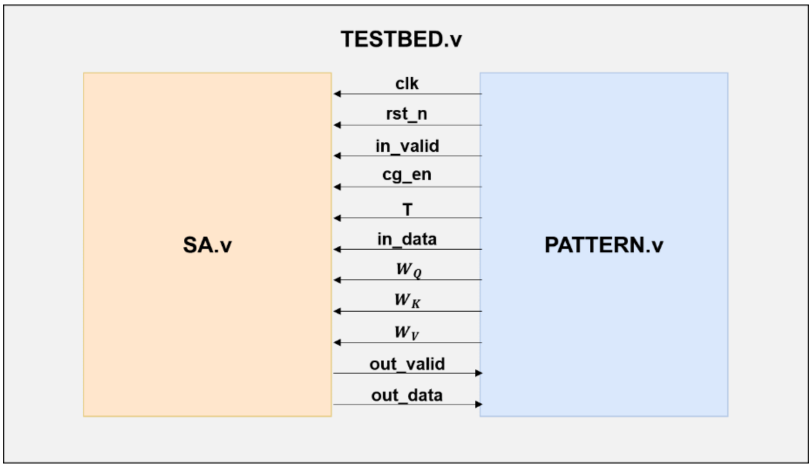
SA_CG_POWER (with clock gating)



```
  Net Switching Power  = 6.579e-04   (14.66%)
  Cell Internal Power  = 3.806e-03   (84.82%)
  Cell Leakage Power   = 2.317e-05   ( 0.52%)
    Intrinsic Leakage  = 2.317e-05
    Gate Leakage       =    0.0000

Total Power            = 4.487e-03  (100.00%)

X Transition Power     = 1.193e-06
Glitching Power        =    0.0000

Peak Power             =    0.1098
Peak Time              =        97
```

24. The gate level simulation cannot include any timing violation without the *notimingcheck* command.
25. Don’t use any wire/reg/submodule/parameter name called *error*, *congratulation*, *latch* or *fail* otherwise you will fail the lab. Note * means any char in front of or behind the word. e.g: error_note is forbidden.
26. Don’t write Chinese comments or other language comment in the file you turned in.
27. **Any error message during synthesis and simulation, regardless of the result will lead to failure in this lab.**
28. **Synthesis time should not exceed 2 hours.**
29. If your file violates the naming rule, you will lose 5 points.

Block Diagram



Grading Policy

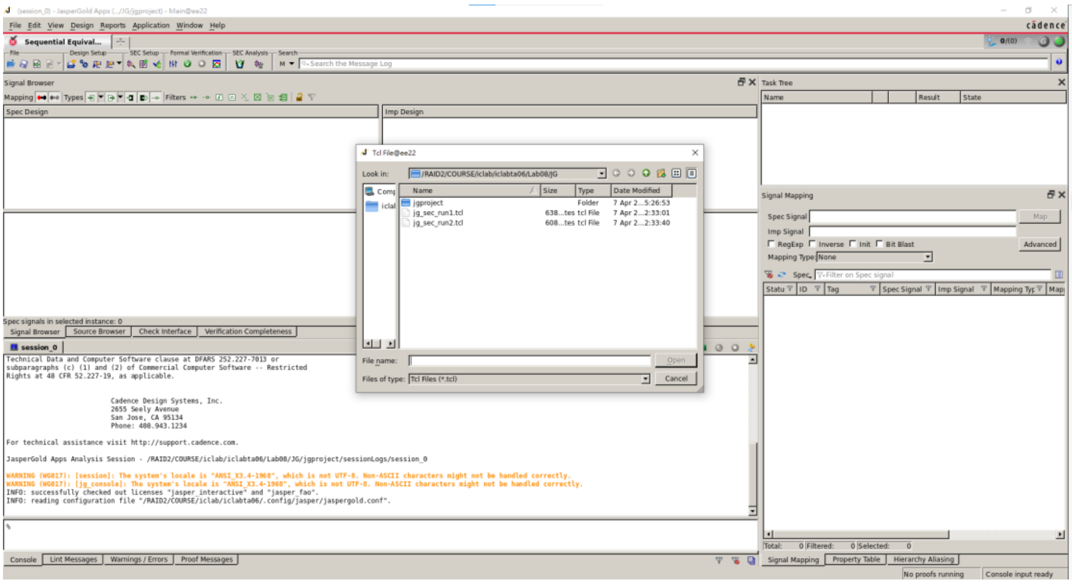
- Functionality (70%)
  - SA_wocg and SA: 60%
  - JasperGold SEC check (10%): Run1 (5%), Run2 (5%)
    (**JasperGold No 2nd demo chance**)
- Performance (30%)
  - (**Total Latency** * **Total Power** (*gated with CG*)) * **Area**
  - Total Latency = 1 + Execution Latency

Note

1. Please submit your design **SA_wocg.v** and **SA.v** in **Lab08/EXERCISE/09_SUBMIT**
   a. 1st_demo deadline: 2024/11/11 (Mon.) 12:00:00
   b. 2nd_demo deadline: 2024/11/13 (Wed.) 12:00:00
2. Template folders and reference commands:
   01_RTL/ (RTL simulation)
   ./01_run_vcs_rtl **./01_run_vcs_rtl_CG** *(only in EXERCISE folder)*
   02_SYN/ (Synthesis)
   ./01_run_dc_shell **./clk_dc_shell** *(only in EXERCISE folder)*
   (Check if there is any **latch** or **error** in your design in **syn.log**)
   (Check the timing of the design in /Report/**SA.timing**)
   03_GATE_SIM/ (Gate-level simulation)
   ./01_run **./02_run_cg** *(only in EXERCISE folder)*
   04_PTPX/ (PrimeTime power analysis)
   ./01_run_ptpx **./02_run_cg_ptpx**
   (Get the power of your design)
   09_SUBMIT/
   ./00_tar **./01_submit** **./02_check**
   You can key in **./09_clean_up** to clear all log files and dump files in each folder.

3. JasperGold SEC execution steps: In folder JG/

(1) ./01_run, ./02_run or **jg –sec &** and click the **File/Tcl_script/source**.



Tcl File@ee22

Look in: /RAID2/COURSE/iclab/iclabta06/Lab08/JG

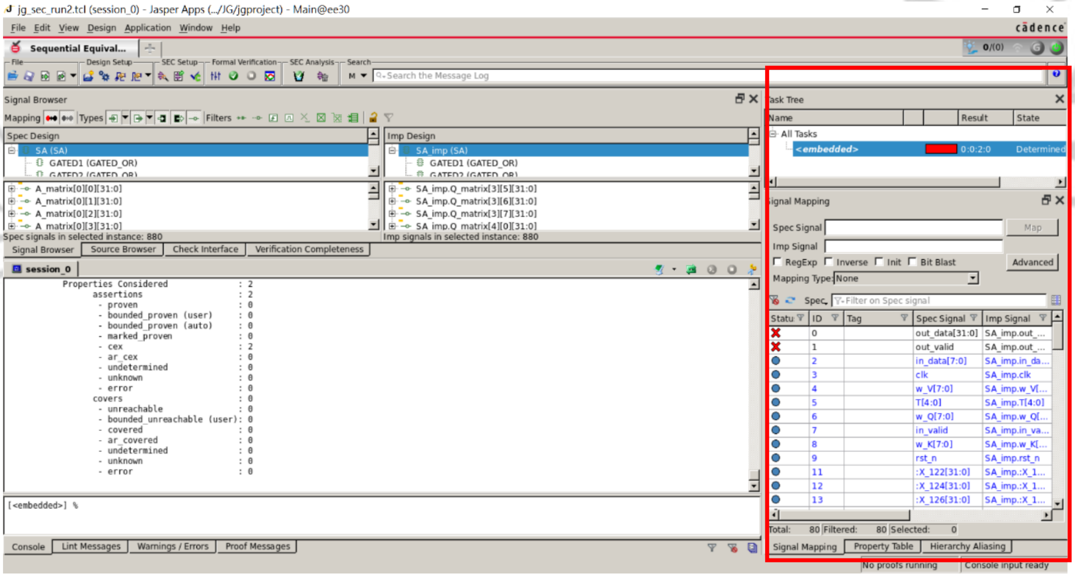
Name	Size	Type	Date Modified
jgproject		Folder	7 Apr 2...5:26:53
jg_sec_run1.tcl	638...tes	tcl File	7 Apr 2...2:33:01
jg_sec_run2.tcl	608...tes	tcl File	7 Apr 2...2:33:40

File name:

Files of type: Tcl Files (*.tcl)

```
Technical Data and Computer Software clause at DFARS 252.227-7013 or
subparagraphs (c) (1) and (2) of Commercial Computer Software -- Restricted
Rights at 48 CFR 52.227-19, as applicable.

                        Cadence Design Systems, Inc.
                        2655 Seely Avenue
                        San Jose, CA 95134
                        Phone: 408.943.1234

For technical assistance visit http://support.cadence.com.

JasperGold Apps Analysis Session - /RAID2/COURSE/iclab/iclabta06/Lab08/JG/jgproject/sessionLogs/session_0

WARNING (WG017): [session]: The system's locale is "ANSI_X3.4-1968", which is not UTF-8. Non-ASCII characters might not be handled correctly.
WARNING (WG017): [jg_console]: The system's locale is "ANSI_X3.4-1968", which is not UTF-8. Non-ASCII characters might not be handled correctly.
INFO: successfully checked out licenses "jasper_interactive" and "jasper_fao".
INFO: reading configuration file "/RAID2/COURSE/iclab/iclabta06/.config/jasper/jaspergold.conf".
```

(2) Check all properties pass.



jg_sec_run2.tcl (session_0) - Jasper Apps (.../JG/jgproject) - Main@ee30

Spec signals in selected instance: 880

Imp signals in selected instance: 880

```
            Properties Considered           : 2
                  assertions                : 2
                   - proven                 : 0
                   - bounded_proven (user)  : 0
                   - bounded_proven (auto)  : 0
                   - marked_proven          : 0
                   - cex                    : 2
                   - ar_cex                 : 0
                   - undetermined           : 0
                   - unknown                : 0
                   - error                  : 0
                  covers                    : 0
                   - unreachable            : 0
                   - bounded_unreachable (user): 0
                   - covered                : 0
                   - ar_covered             : 0
                   - undetermined           : 0
                   - unknown                : 0
                   - error                  : 0
[<embedded>] %
```

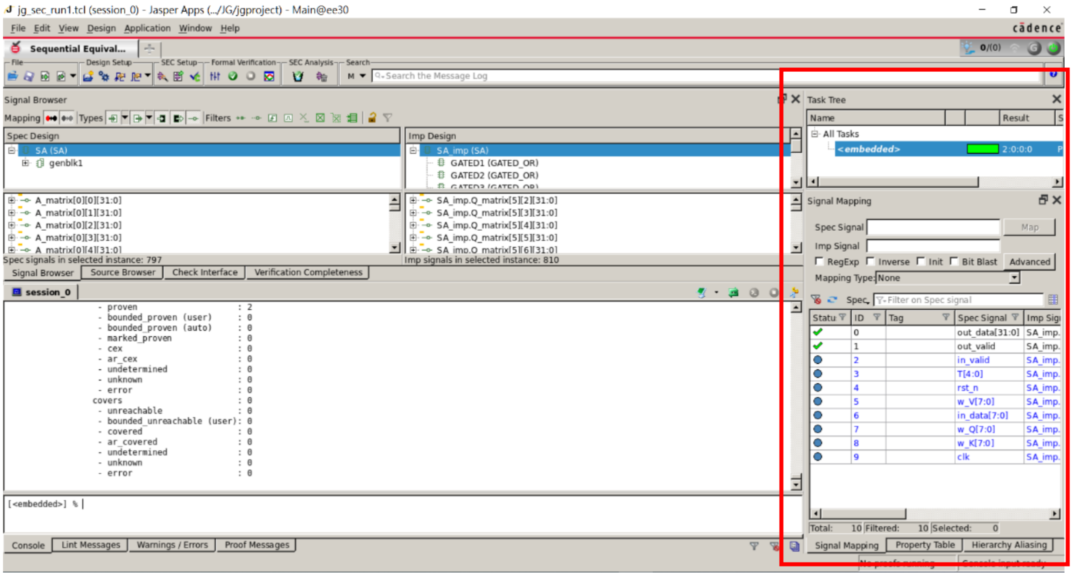
Task Tree

All Tasks

<embedded> 0:0:2:0 Determined

Statu	ID	Tag	Spec Signal	Imp Signal
✗	0		out_data[31:0]	SA_imp.out_...
✗	1		out_valid	SA_imp.out_...
●	2		in_data[7:0]	SA_imp.in_da...
●	3		clk	SA_imp.clk
●	4		w_V[7:0]	SA_imp.w_V[...
●	5		T[4:0]	SA_imp.T[4:0]
●	6		w_Q[7:0]	SA_imp.w_Q[...
●	7		in_valid	SA_imp.in_va...
●	8		w_K[7:0]	SA_imp.w_K[...
●	9		rst_n	SA_imp.rst_n
●	11		:X_122[31:0]	SA_imp.:X_1...
●	12		:X_124[31:0]	SA_imp.:X_1...
●	13		:X_126[31:0]	SA_imp.:X_1...

Total: 80 Filtered: 80 Selected: 0



jg_sec_run1.tcl (session_0) - Jasper Apps (.../JG/jgproject) - Main@ee30

Spec signals in selected instance: 797

Imp signals in selected instance: 810

```
                   - proven                 : 2
                   - bounded_proven (user)  : 0
                   - bounded_proven (auto)  : 0
                   - marked_proven          : 0
                   - cex                    : 0
                   - ar_cex                 : 0
                   - undetermined           : 0
                   - unknown                : 0
                   - error                  : 0
                  covers                    : 0
                   - unreachable            : 0
                   - bounded_unreachable (user): 0
                   - covered                : 0
                   - ar_covered             : 0
                   - undetermined           : 0
                   - unknown                : 0
                   - error                  : 0
[<embedded>] % |
```

Task Tree

All Tasks

<embedded> 2:0:0:0

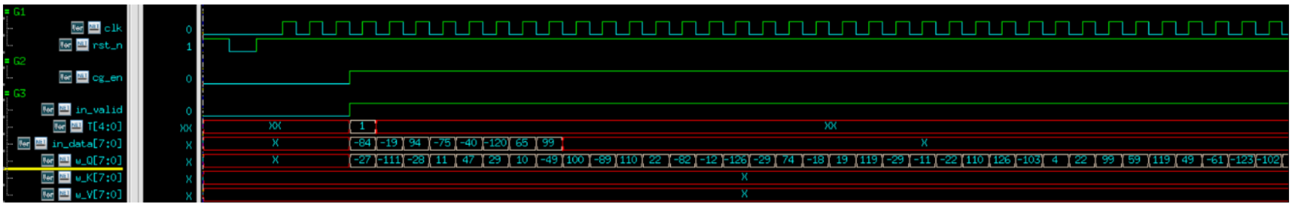
Statu	ID	Tag	Spec Signal	Imp Sig
✓	0		out_data[31:0]	SA_imp.
✓	1		out_valid	SA_imp.
●	2		in_valid	SA_imp.
●	3		T[4:0]	SA_imp.
●	4		rst_n	SA_imp.
●	5		w_V[7:0]	SA_imp.
●	6		in_data[7:0]	SA_imp.
●	7		w_Q[7:0]	SA_imp.
●	8		w_K[7:0]	SA_imp.
●	9		clk	SA_imp.

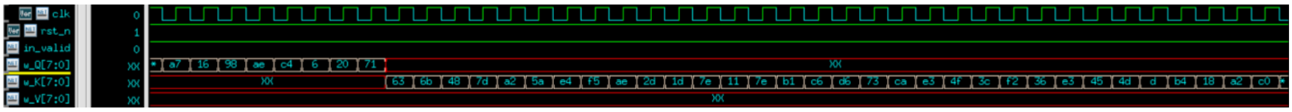
Total: 10 Filtered: 10 Selected: 0

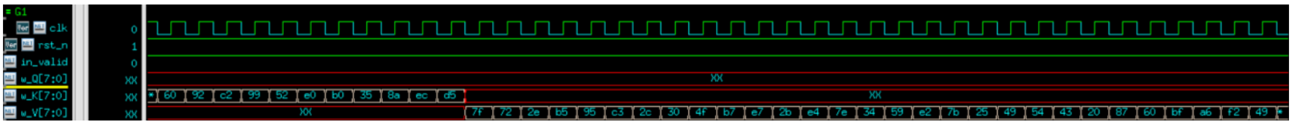
You should pass all tasks in both ./01_run & ./02_run.

Waveform Example

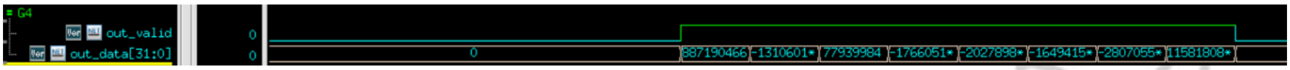
1. Input Signals



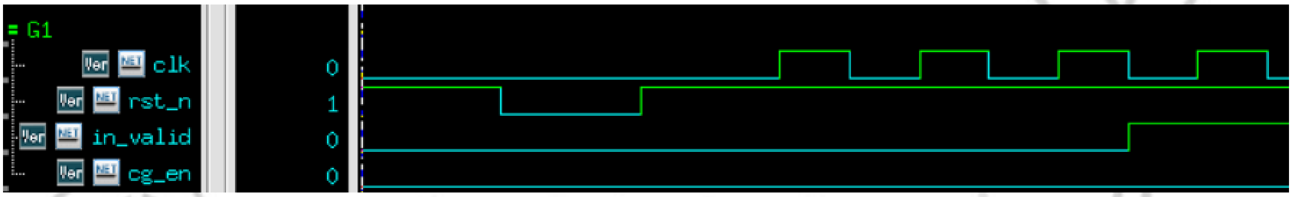

clk
rst_n
in_valid
w_Q[7:0]
w_K[7:0]
w_V[7:0]
a7 16 98 ae c4 6 20 71
63 6b 48 7d a2 5a e4 f5 ae 2d 1d 7e 11 7e b1 c6 d6 73 ca e3 4f 3c f2 36 e3 45 4d d b4 18 a2 c0


G1
clk
rst_n
in_valid
w_Q[7:0]
w_K[7:0]
w_V[7:0]
60 92 c2 99 52 e0 b0 35 8a ec d5
7f 72 2e b5 95 c3 2c 30 4f b7 e7 2b e4 7e 34 59 e2 7b 25 49 54 43 20 87 60 bf a6 f2 49

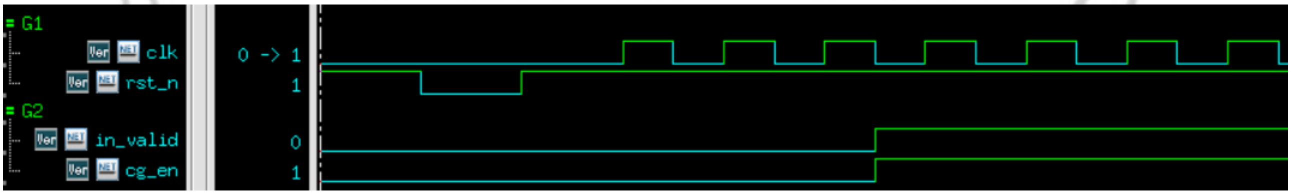
2. Output Signals



3. For ./EXERCISE/PATTERN.v , you need to set cg_en = 0:



4. For ./EXERCISE/PATTERN_CG.v , you need to set cg_en = 1:


G1
clk
rst_n
G2
in_valid
cg_en
0 -> 1
1
0
1