Fakulti:	**FAKULTI KEJURUTERAAN ELEKTRIK**
Nama Matapelajaran: MAKMAL PBL TAHUN 3 Kod Matapelajaran : SKEL 3742	Semakan : 1 Tarikh Keluaran : 2020 Pindaan Terakhir : 2020 No. Prosedur : **PK-UTM-FKE-(0)-10**

SKEL 3742

SEKOLAH KEJURUTERAAN ELEKTRIK
UNIVERSITI TEKNOLOGI MALAYSIA
KAMPUS SKUDAI
JOHOR

VLSI SYSTEM DESIGN
LABORATORY (VLSI DESIGN)

Design Of 64-Bit Registered ALU Circuits With and Without Clock Gating.

Prepared by :	Certified by :
Mr. Izam bin Kamisian Dr. Muhammad Afiq Nurudin bin Hamzah Dr. Shahidatul Sadiah binti Abdul Manan	P.M. Dr. Ir. Rubita binti Sudirman (Head of ECE Department)
Signature :	Signature :
Stamp :	Stamp :
Date : 9 February 2020	Date : 9 February 2020

Objective

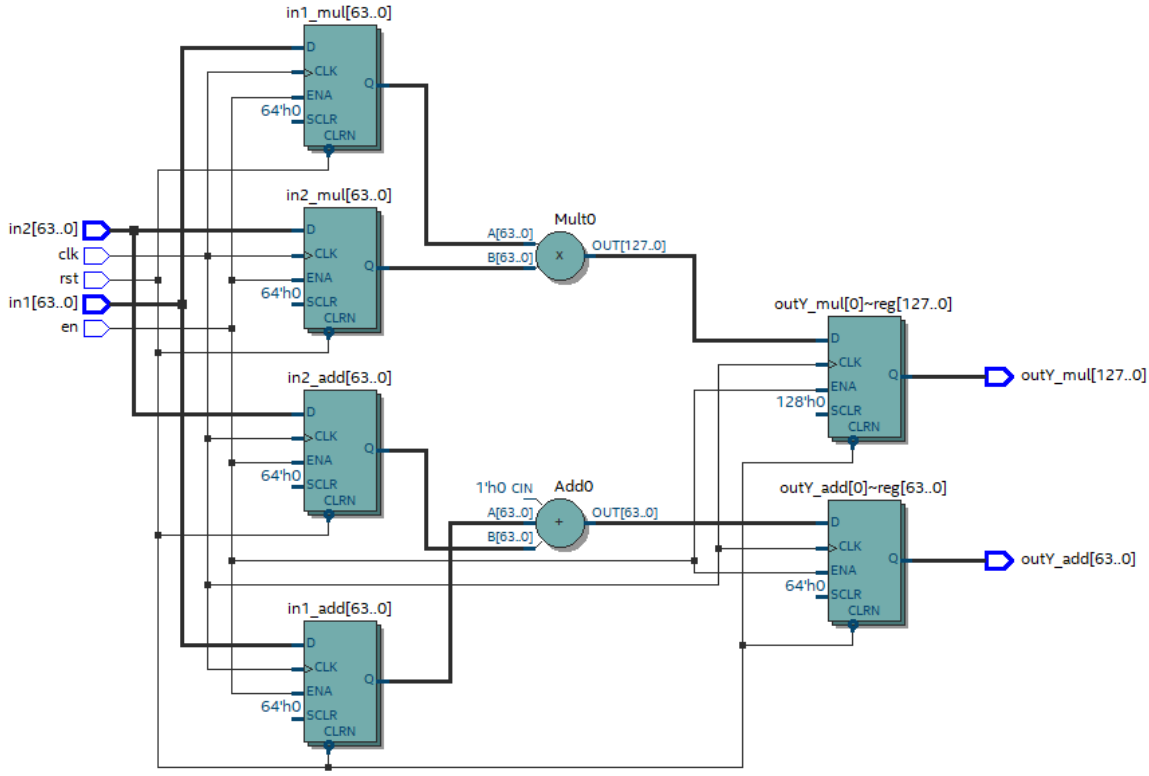
The goals of this assignment are

1. To perform a complete RTL to GDS design flow.
2. To compare the dynamic power consumption between circuit with and without clock gating.

You will take a given design through the process of functional verification, logic synthesis (with and without clock gating), physical implementation (place & route) and static timing analysis.

Design Overview

The design that you will be working on is a pair of 64-bit arithmetic operations, e.g., addition and multiplication. *Each cycle will conduct a different combination of arithmetic operation, refer Table 1.* The design has both the inputs and outputs registered. The inputs are feed to both operations to give the corresponding results as shown in the following figure. The circuit which compiled without clock gating will run both operations simultaneously, while the other one will run only one operation depending on the value of the input signal, "en". Meanwhile, the clock gating part is not in the design, but it is done in synthesis process using DC tool.



in1, in2, outY_add is 64-bits, outY_mul is 128-bits

Table 1. Arithmetic Combination for Each Cycle.

Cycle	Arithmetic Combination
1	Addition and multiplication
2	Subtraction and multiplication
3	Addition and division
4	Subtraction and division

Design Constraints

- Pre-layout constraints:
  - Clock port: clk
  - Clock period: 12ns
  - Clock uncertainty: 0.3ns
  - Clock latency: 1.0ns
  - Clock transition: 0.2ns
  - Input delay on all inputs except clock: 1.0ns
  - Output delay on all outputs: 1.0ns
  - Driving cell on all inputs except clock: NBUFFX4_LVT
  - Load on all outputs: 0.2
  - Operating condition: ss0p95v125c
  - Area: smallest possible that meets setup timing
  - Leave all others to their respective default values
- Post-layout constraints:
  - Clock port: clk (clock must be propagated)
  - Clock period: 12ns
  - Input delay on all inputs except clock: 1.0ns
  - Output delay on all outputs: 1.0ns
  - Driving cell on all inputs except clock: NBUFFX4_LVT
  - Load on all outputs: 0.2ff
  - Operating condition: ss0p95v125c
    - Analysis type: On chip variation
  - Leave all others to their respective default values
- Physical constraints
  - Max routing layer is M1 and min routing layer is M6.
  - Power net: VDD
  - Ground net: VSS
  - Supply voltage: 0.95V
  - Max IR drop: 20%

Tasks

The Verilog RTL source code for this assignment is in the directory **assignment/source**. You need to modify the source code (**arith64.v**) according to the arithmetic operations given in Table 1.

1. Verification
    a. Perform all functional verification task in the directory /**assignment/verification**.
    b. Use VCS to perform all functional verification task.
    c. Create a Verilog testbench to verify the given RTL design. Your testbench must validate
        i. Clock function
        ii. Reset and enable
        iii. Multiple combinations of input data – minimum of at least 20 different input data.
    d. Save your Verilog testbench as **tb.v** in the directory **assignments/source**.
    e. Run the verification in VCS using your testbench and the arith64.v design file. Based on the waveform, explained to the lab instructor that the design has obtained the expected outcome.

        Example: if in1 is 2 and in2 is 10, output of multiplication is 20 on the correct clock cycle.
    f. Verify the function of the design is correct. If necessary, make changes to your testbench. Note that it will take two cycles for data to come out.

2. Synthesis
    a. Perform all synthesis task in the directory **/assignment/synthesis**.
    b. The library and tool settings have already been completed. Do not modify the **.synopsys_dc.setup** file.
    c. Use DC Ultra in topo mode to perform all synthesis tasks.

        dc_shell -topo
    d. Based on the **pre-layout** design constraints given above, create a constraints file with the name **dc_constraints.tcl** and save it to the **assignment/scripts** directory. *The following constraints command are not given in the prelab*.
        - set_clock_latency 1.0 [get_clock clk]
        - set_input_delay 1.0 -clock clk [remove_from_collection [all_inputs] [get_port clk]]
        - set_driving_cell -lib_cell NBUFFX4_LVT [remove_from_collection [all_inputs] [get_port clk]]
        - set_operating_conditions ss0p95v125c
    e. Create a script called **dc.tcl** for the circuit without clock gating and **dc_cg.tcl** for the circuit with the clock gating. Then, put both **.tcl** files in the **/scripts** directory. The tcl scripts must contain the commands to
        i. Read in the design
        ii. Set the correct current design
        iii. link the design
        iv. Apply the design constraints using the file **dc_constraints.tcl** for both with and without clock gating circuits.

v. Optimize the design to meet setup timing constraints. For the circuit with the clock gating, use option "-gate_clock" to compile.
vi. Generate the constraints and setup timing reports. Save the reports to **/assignment/reports/DC** directory. All reports generated by DC must start with the prefix **dc_**. For example, a timing report will be called: dc_timing.txt
vii. Save the optimized design in **ddc** format and as a Verilog netlist in the directory: **/assignment/netlist**
viii. Exits DC upon completion of the above tasks.

f. Execute the run script using the following command structure:
dc_shell -topo -f ../scripts/run.tcl |tee -i run.log
for without clock gating.
dc_shell -topo -f ../scripts/run_cg.tcl |tee -i run.log
for with clock gating.

g. Once optimization completes, view the reports using any text editor (for example, gedit). Make sure you meet all setup timing constraints. ***You can ignore area violations***.

3. Physical Implementation
   a. Perform all physical implementation task for both circuits with and without the clock gating in the directory **/assignment/pnr**.
   b. The library and tool settings have already been completed. Do not modify the **.synopsys_dc.setup** file.
   c. Use IC Compiler to perform all physical implementation tasks.
   d. Using the **ddc** file generated from the synthesis task (saved in the **netlist** directory),
      i. Read the **ddc** file into IC Compiler
      ii. Perform a basic floorplan.
         1. This is a block level design with no IO pads.
         2. You can place the pins in any order
         3. Use the physical constraints given above, all other physical constraints are up to user discretion.
      iii. Run the placement and routing commands.
      Make sure you run "**set_separate_process_options -placement false**" before "place_opt"
      iv. Run the necessary timing and constraints reports (just like in DC) to verify you have met the setup timing constraints. Screenshot your layout for evidence to be included in the final report.
      v. Save all the reports to **/assignment/reports/ICC** directory. All reports generated by ICC must start with the prefix icc_.
      vi. Run the **verify_zrt_route** and **verify_lvs** command to ensure the design has no physical rule violations (DRC/LVS checks)
      vii. Save the completed design in **ddc** and **Verilog netlist** formats to the **netlist** directory. Use filename **arith64_postlayout** to differentiate the data generated from IC Compiler compared with Design Compiler.
      viii. Write out the parasitic file in **SPEF** format, save it in the directory **netlist**.

e. Make sure you meet all setup timing constraints. ***You can ignore area violations***.
f. All the tasks can be performed using either IC Compiler GUI (**icc_shell -gui**) or in shell mode (**icc_shell**) using your own **.tcl** scripts, or combination of GUI and shell mode. If you do use any .tcl scripts, save all the scripts with the proper prefix to the /**scripts** directory.

4. Static Timing Analysis
    a. Perform all static timing analysis task for both circuits with and without the clock gating in the directory **/assignment/sta**.
    b. The library and tool settings have already been completed. Do not modify the **.synopsys_pt.setup** file.
    c. Use PrimeTime to perform all static timing analysis tasks.
    d. Based on the **post-layout** design constraints given above, create a constraints file with the name **pt_constraints.tcl** and save it to the **/assignment/scripts** directory.
    e. Create **pt.tcl** scripts for both circuits and place it in the /**scripts** directory. The script must contain the commands to
        i. Read in the design using Verilog netlist generated from the physical implementation task.
        ii. Set the correct current design.
        iii. Link the design.
        iv. Apply the design constraints using the file **pt_constraints.tcl**.
        v. Read in the **SPEF** file generated from physical implementation task, which was saved in the **netlist** directory.
        vi. Enable the power analysis by command
        set power_enable_analysis true
        vii. Generate a report to check the quality of the parasitic back-annotation.
        viii. Generate reports to show all the constraints have been applied correctly.
        ix. Generate a report for showing the current status of the design (**report_analysis_coverage**).
        x. Generate the constraints and setup timing reports.
        xi. Save all the reports to **/assignment/reports/PT** directory. All reports generated by PT must start with the prefix **pt_**. For example, a timing report will be called: pt_timing.txt
        xii. Exits PT upon completion of the above tasks.

    f. Execute the run script using the following command structure:
        pt_shell -f ../scripts/pt_run.tcl |tee -i run.log
    for without clock gating.
        pt_shell -f ../scripts/pt_run_cg.tcl |tee -i run.log
    for with clock gating.

g. View the reports using any text editor (for example, gedit). Make sure you meet all setup timing constraints. ***You can ignore area violations***.

Deliverables

Hand in the following data to your lab instructor:

1. Netlist (prelayout, postlayout)
2. Reports (DC, ICC, PT)
3. Run (execution) logs (dc.tcl, dc_cg.tcl, pt.tcl)
4. Summary report of your results from DC, ICC and PT

Include the following discussion in your report.

1. Discuss the timing slack difference for pre-layout and post-layout.
2. Tabulate the timing slacks, dynamic power dissipation, and area violations for the circuits with and without clock gating from DC, ICC, and PT tools.
3. What are the advantages and disadvantages of clock gating in term of timing, power, and area? Discuss with justification from the tabulated results.

End of assignment. Thank you.