Group #:

G

Final Project

ENSC 350 1257

Project Submitted in Partial Fulfillment of the
Requirements for Ensc 350
Towards a Bachelor Degree in Engineering Science.

Last Name:	Last Name:	Last Name:
SID:	**SID:**	**SID:**

Intentionally left blank

Table of Contents:

Table of Contents		(?)
Objectives		(1)
Specifications		
	DUT – The Execution Unit Entity.	(2)
	DUT – Internal Subsystems.	(2)
	Verification – Testbench Specification.	(3)
General Tasks		
	1) Design Project Setup.	(4)
	2) Baseline Design.	(4)
	3a) Design Candidates – topological considerations.	(5)
	3a) Design Candidates – implementation considerations.	(6)
	3b) Testbench Design – Testbench for the DUT.	(7)
	4) Iterative Experiments – Synthesis & Simulation of the DUT.	(8)
	5) Analysis & Optimization – Analysis of Cost & Performance.	(9)
	6) Analysis & Optimization – Optimization Experiments.	(9)
Deliverables		
	FP: Full Documentation Requirements.	(10)
	The Archive Contents.	(10)
In-Lab Assessment		
	FP: Project Assessment Procedure.	(12)

SFU

Ensc 350-1257
Fall 2025

Final Project

OBJ

OBJ

Main Objectives:

1) To create multiple design candidates for a **RV64I** Execution Unit.
2) To synthesise all design candidates and compare relative cost.
3) To verify correct functional behaviour of all design candidates.
4) To verify timing performance of all design candidates.
5) To document all activities performed in accomplishing the project tasks.

Sub-Objectives:

1) To setup an organised procedure for Synthesis & Verification, using VHDL configurations and ModelSim scripts.

2) To create a ripple adder to be used as a baseline circuit for calibrating observed cost and performance metrics.

3) To create additional design candidates for the DUT, employing advanced design principles.

4) To create a single testbench used for both functional and timing verification of all DUT implementations.

5) To create test vectors for verifying all internal sub-systems of the DUT.

6) To synthesis, realise and compare relative cost for all design candidates on both a Cyclone IV and a Arria II FPGAs.

7) To identify and compare worst-case propagation delay from a timing verification using a Quartus generated timing netlist.

8) To analyse cost & performance results and then design candidates optimized for lowest cost, best performance and for optimal cost/performance ratio.

9) To improve your ability to maintain engineering notebooks and fully keep track of R&D activity.

10) To improve your ability to write reports that document R&D activity and technical reports that document actual designs.

Specifications - Device Under Test: (DUT)

The Execution Unit:

The Execution Unit should

- be compatible with all **RV64I** instructions, and
- include control inputs ExUFunc := {FuncClass, LogicFN, ShiftFN, AddnSub, ExtWord} encoded according to the following tables,

FuncClass		operation
0	0	shift/arith
0	1	logic
1	0	slt
1	1	sltu

LogicFn		operation
0	0	Lui
0	1	A xor B
1	0	A or B
1	1	A and B

ShiftFN		operation
0	0	arith
0	1	sll
1	0	srl
1	1	sra

- ExtWord = '1' for 32-bit instructions
- have three status outputs, ExUStatus := {Zero, AltB, AltBu}. These flags are typically used for Branch and Compare type instructions however they should always be computed to indicate the 64-bit status of the inputs A and B.
- should only use `std_logic` or `std_logic_vector` for its interface signals.
  (to be compatible with the types created by Quartus' netlist)

```
Entity ExecUnit is
Generic ( N : natural := 64 );
Port ( A, B : in std_logic_vector( N-1 downto 0 );
       FuncClass, LogicFN, ShiftFN : in std_logic_vector( 1 downto 0 );
       AddnSub, ExtWord : in std_logic := '0';
       Y : out std_logic_vector( N-1 downto 0 );
       Zero, AltB, AltBu : out std_logic  );
End Entity ExecUnit;
```

Internal Subsystems::

- The Execution Unit should be organized as three major subsystems, Arithmetic, Shift and Logic.
- The remainder of the circuit will contain elements that ensure control word actions as defined above.
- The three flags, { Zero, AltB, AltBu } are assumed to be correct for 64-bit operations. 32-bit instructions such as ADDW do not need to produce correct flags.
- The entity declarations of internal subsystems may vary for different design candidates. As such it would be wise to organize the higher level architectures to instantiate components and define the mapping of ports within configuration declarations.

Specifications – Verification:

Testbench Specification:

The Testbench should cycle through a reasonable number of test vectors,

- Test vectors may be created using external tools and stored in a text file,

- The test vectors in the file should be formatted with one line for each measurement; with values in the order

  { A, B, FuncClass, LogicFN, ShiftFN, AddnSub, ExtWord, Y, Zero, AltB, AltBu }

  - A B & Y are 64-bits in Hexadecimal, all other signals are in binary.
  - Outputs, Y, Zero, AltB, AltBu are known correct responses.

Example:

F0A5F0C3F0A5F0C3 0F5AF03C0F5AF03C 01 00 00 1 1 0F5AF03C0F5AF03C 010

- The output flags need only be valid in the case of Branch and Compare type instructions. Both the circuits and test vectors should construct the flags based solely on the inputs A and B irrespective of whether the values are valid, for ALL operations.

- For functional verification, test vectors should be selected to ensure correct operation of each output bit individually for both a rising event and also a falling event.

- For Timing verification, test vectors should be selected to cover sufficiently many critical paths of each design candidate.

- The DUT should be instantiated as a component; VHDL configurations and simulation scripts should be sensibly created to manage the binding of components and entities.

- Each measurement must be labelled with an index number and the DUT must be initialized with the stimuli ‘X’ at all input bits, held for a pre-defined time, PreStimTime.

- The output results for each simulation run should be stored in an appropriately named transcript file. Each string should contain the measurement index, the input values, the output results and additional information about the status of the measurement.
  (There should be a sensible convention in naming configurations, scripts & transcripts.)

- The overall experiment should be framed with a start notification and an end notification. Worst-case propagation delay should be reported in the end notification.

General Tasks:

1) Design Project Setup:

Refer to SW-Install-Setup-350-xxxx.pdf

- Prepare your filing structure. (§Filing Structure)
- Prepare ModelSim project. (§Setting up ModelSim) – FP.mpf
- Prepare a Quartus Project. (§Setting up Quartus) – FP.qpf

Refer to Project Documentation Guidelines-350-xxxx.pdf

- Prepare the Activity Log file.(§Activity Logs)
- Prepare an Engineering Notebook.(§Engineering Notebooks)

2) Baseline Device: (using a Cyclone IV FPGA)

An 64-bit Ripple adder implemented on a Cyclone IV FPGA (EP4C-E-115-F29-C7) will be used as a standard circuit representing the scale of the design, referred to as the baseline design.

- The cost, and performance of the baseline design will be used to normalize the respective metrics of all candidate implementations.

Write a structural VHDL description for an N-bit ripple adder,
with an architecture named "Baseline"

Baseline: Create a structural description that explicitly instantiates full adders, or iteratively generate the single bit carry and sum expressions, or use array operations.

- Check that your code compiles with both Quartus and ModelSim/Questa.
- Check that your code can be loaded by ModelSim/Questa and synthesised by Quartus.
- Add comprehensive comments to the sourcecode.

The Baseline Device is defined as
a 64-bit structural ripple adder implemented on a Cyclone IV FPGA (EP4C-E-115-F29-C7).

- Alternatively, you may select a different circuit, implemented on a Cyclone IV FPGA as a baseline design. Briefly justify your reasons for selecting this circuit.

For this baseline system – DO NOT USE the "+" operator.

Update the log file and your notebook.

General Tasks: Design Candidates

3a) Design Candidates - topological considerations:

➢ Write multiple VHDL architectures, for each of the three main subsystems;

- Check that your code compiles with both Quartus and ModelSim/Questa.
- Check that your code can be loaded by ModelSim/Questa and synthesised by Quartus.
- Add comprehensive comments to the sourcecode.

➢ Avoid redundant definitions of similar architectures. Partition the overall project into useful components and manage an organized hierarchy using VHDL configurations that bind Entity/ Architectures to components.

✓ Arithmetic Subsystem: include architectures that exploit a variety of design principles; for example, the Carry-Select principle, the pre-fix Look-Ahead concept, the Carry-Skip principle.

✓ Logic Subsystem: This subsystem is primitive, thus one topology is sufficient.

✓ Shift Subsystem: include architectures that exploit a variety of design principles; for example, optimizing the length of a barrel-shifter, combining multiple shift operations into a single barrel-shifter, combining 64-bit/32-bit operations into a single barrel-shifter.

➢ Establish a naming convention to easily identify the architectures, & configurations.

✓ Avoid creating excessively long names. (some people consider '_' akin to the dodo)
✓ Try using short sequences of fixed length, ie Rip64, Csa64, Bka64 ...
✓ Use digits to indicate numeric information such as operand-width, barrel-shifter MUX sequence etc.

Warning: The VHDL "**sra**" operator is obsolete and most likely will not work correctly. You may wish to use the ieee functions, RESIZE(), SHIFT_LEFT() & SHIFT_RIGHT(), instead of explicitly manipulating array slices.

Update the log file and your notebook.

3a) Design Candidates - implementation considerations:

- Each design candidate uses specific topologies, a specific hierarchical structure and is finally implemented on a specific target FPGA device.
  - ✓ You should create at least four design topologies => eight design candidates.
  - ✓ The baseline device not considered as a design candidates; it is simply used as a normalization device for comparing measured quantities.

- Use Quartus to Synthesise an implementation for each topology/hierarchy using

  1) a Cyclone IV FPGA and

  2) an ARRIA II FPGA.

  You should select a version that fits your design and avoids inclusion of unnecessary additional hardware resources. You may wish to consider (as a weak recommendation) (EP4C-E-115-F29-C7, and EP2A-GX-45-D-F29-C6)

- Update your naming convention to include identifiers that indicate the target FPGA. (maybe append the characters C4 and A2)

- For each design candidate, rename the timing models produced by Quartus.
  - ✓ Store these files logically in your project file hierarchy.
  - ✓ You may also wish to force Quartus to name the architecture of its timing models. Refer to SW-Install-Setup-350-xxxx.pdf - §EDA Netlist Writer Settings

- Create individual configurations and scripts to distinguish between functional and timing VHDL models. The naming convention will apply to Your VHDL sourcecode filenames, Your VHDL architecture names, timing-model filenames, Configuration names, Script filenames, Simulation-Output Transcript filenames. (and possibly also timing model architecture names)

Update the log file and your notebook.

General Tasks: Testbench Design

3d) Testbench for the DUT:

- Write a single testbench for the Execution Unit. The testbench should
  - ✓ Refer to the specifications for general requirements.
- Create Test Vectors:
  - ✓ Refer to the specifications for general requirements.
- The DUT should be instantiated as a component. Create configurations and scripts to manage simulation experiments.
  - ✓ Refer to the specifications for general requirements.
- Include useful information in a well formatted wave window. Index, propagation delays etc.
  - ✓ Use separate scripts to efficiently maintain organized wave windows.
- Add extensive comments to the testbench so that anyone can understand the code in the future.

Update the log file and your notebook.

General Tasks: **Iterative Experiments**

4) Synthesis & Simulation of the DUT:

- Synthesize all design candidates.
  - ✓ Record all information from Quartus, that relates to the cost of the implemented device.
  - ✓ Rename and store the timing models.
  - ✓ Review RTL and post-fit netlists to ensure that your VHDL coding methods have successfully produced your desired topological features.

- Using a ModelSim/Questa project, run both, a functional simulation and a timing simulation for each design candidate.
  - ✓ Setup a well formatted wave window and save the script.
  - ✓ You may wish to add specific signals that are unique to a given topology for diagnosics.

- Create multiple scripts; each for design candidate, both a functional simulation and also a timing simulation. You may wish to update your naming convention to use prefixes "**FS**" and "**TS**" to the script filenames, to distinguish between the two types of simulation runs.

Each script should,

- ✓ name a unique transcript file,
- ✓ turn on/off re-direction to this transcript file when you wish to capture the output,
- ✓ compile all relevant sourcecode, including the configuration,
- ✓ start a simulation of the appropriate configuration,
- ✓ setup an appropriate wave window, and
- ✓ run the simulation.

Update the log file and your notebook.

General Tasks: Analysis & Optimization

5) Analysis of Cost & Performance:

- Using the cost results from your experiment, create a summary table of costs.
  - ✓ Include your initial predictions in measurement units, (LE).
  - ✓ Include the actual raw measurements produced by Quartus.
  - ✓ Include processed results, ie conversions between FPGA units and normalized values.

- Using the measured delays of the Baseline device, estimate a bound on the propagation delay the LUTs in both a Cyclone IV and an ARRIA II. (operating in normal mode)

- Analyse each candidate an make a manual estimate of the worst-case delays.
  - ✓ record the predicted & actual measured worst-case delays.
  - ✓ Create a concise table of performance to easily compare, predicted results, measured results and derived parameters for each design candidate.

6) Optimization Experiments:

- If your experimental process is implemented efficiently, it should not be too difficult to repeat all measurements and collect (timing & cost) results for different values of N. Furthermore, you could experiment with individual subsystems by replacing the other subsystems with dummy components. Worst-Case Test Vectors would need to be re-constructed. If test vector filenames include the vector size, the single testbench may be modified to use a generic parameter to construct a test vector filename and read the appropriate test vector file.
  - It may be the case that an optimal cost susbsystem depends on the value of N.
  - It may also be that an optimal performance susbsystem depends on the value of N.
  - Record the cost and performance for various circuit sizes, in a convenient table.
  - You may also wish to plot a graph to find break-point values of N.

- Using your results from your experiments, devise some simple experiments that combine topologies to find subsystems optimized for cost or performance. Identify the bottle-necks in your design. Using the identified bottle-necks optimize for a secondary parameter.

- Construct a design candidate that optimizes the cost-performance ratio.
  - ✓ Create final summary table that conveniently provides cost, performance and cost-performance ratio.

- You will need to document all of your design attempts, including design candidates that were not optimal.

Update the log file and your notebook.

Deliverables: Files to be Submitted for Assessment

FP: Full Documentation Requirements

Your project documentation will contain many files. These files are to be zipped into a single archive called "FP-Gxx-350-1257.zip"

- The Archive file should contain **ONE** root folder named **FP**.
- Each group is to submit one archive file.
- The archive must be structured as specified in the section "Filing Structure" in the Project Documentation Guidelines.

This project requires **TWO** written documents;

- A summary report, being a business style letter that documents your R&D activities, and
- A design project report, that documents technical details specific to the project.

Refer to the section (Project Documentation Guidelines-350-xxxx.pdf§Reports)

- **FP** requires that you submit complete final documents.

The Archive Contents:

1) Submit an archive file

- ✓ A root project directory with all sub-directories forming the specified hierarchy.
- ✓ All required files should be filed in the correct sub-directories.

The file structure is defined in "Project Documentation Guidelines-350-xxxx.pdf §Filing Structure"

- ✓ **Delete all unnecessary files**. Marks will be subtracted if you include useless files such as database files generated by Quartus and ModelSim/Questa

2) Submit an completed Project Report, placed in the Documentation directory.

- ✓ Filename = "FP-Report-Gxx-350-1257.pdf"
- ✓ Cover Page.
- ✓ Table of Contents – with page numbers.
- ✓ Refer to "Project Documentation Guidelines-350-xxxx.pdf§A Design Project Report"

3) Submit an completed Summary Report, placed in the Documentation directory.

- ✓ Filename = "FP-Summary-Gxx-350-1257.pdf"
- ✓ No Cover Page.
- ✓ No Table of Contents.
- ✓ At most two pages.(§A Summary Report)

4) The log files for each group member, placed in the Documentation directory.(§Activity Logs)

- ✓ Completed entries with activities associated with the **FP** deadline.
- ✓ The same log file is used for DP1.0, DP1.1, DP1.2, DP2.0, DP2.1 & FP.
- ✓ The workbook contains separate worksheets for each stage of Ensc 350 project work.

5) Sourcecode:(§Documenting VHDL Sourcecode)

- ✓ Machine-readable Text files containing all original sourcecode, filed in the appropriate sub-directories.
- ✓ Human-readable PDF Appendices containing files sourcecode listings with syntax highlighting, filed in the same location as the project report.

6) Testvectors:

- ✓ Machine-readable Text files, containing the test vectors used to produce the results given in the report, filed in the appropriate sub-directory.

7) Execution Scripts:

- ✓ Machine-readable Text files, containing scripts that you created to produce results from ModelSim (and/or Quartus).

8) ModelSim Transcripts: (Timing simulation results)

- ✓ Human-readable Text files, containing the output from the simulation runs.
- ✓ The scripts and VHDL sourcecode should be created to generate comprehensive, well-organised transcripts.
- ✓ A human reader should be able to easily and quickly be able to find useful information.

9) Quartus Output Reports:(§Documenting Quartus Output Reports)

- ✓ A single Human-readable Text file, created from Quartus summary files.
- ✓ Merge the individual **summary files** for each design candidate into a single file.
- ✓ Insert Section Titles that identify the information relevant to each design candidate.
- ✓ Do Not include other reports generated by Quartus.

10) Quartus and ModelSim Project Files:

- ✓ Include the **.QPF**, **.QSF** and **.MPF** files containing Quartus/ModelSim configurations.
- ✓ Do not include any other superfluous settings files.

11) Document Name Summary Table:

- ✓ Filename = "FP-FileList-Gxx-350-1257.pdf"
- ✓ A one-page PDF document containing a table that lists the mapping of Directories/Sub-Directories to Filenames.
- ✓ Add a brief description of file content.

In-Lab Assessment:

FP - Project Assessment Procedure:

A 20-30 minute assessment will take place in the lab TIME.

All group members should be present for the demonstration.

- You should sign-up for a demonstration time using Canvas.

During the demonstration you must be prepared to quickly and efficiently show that you have accomplished all task.

Demonstration Activities:

- ✓ (10 minutes) Review of the Adder source code & the testbench source code. (10)
- ✓ (5 minutes) A demonstration of
  - valid synthesis.
  - functional simulation from a script &
  - timing simulation from a script.
- ✓ (5 minutes) A question period where individual members will answer posed questions.