York University

Department of Electrical Engineering and Computer Science

EECS 4612: Digital VLSI

Fall 2025

Term Project: Design of a 32-Bit ALU

Amir M. Sodagar

TA: Mohsen Namavar – Abel Beyene

1. Purpose

In this project, you will design a 32-bit *Arithmetic-Logic Unit* (*ALU*) based on a hierarchical structure. The design will be first simulated and then implemented in the form of a complete chip using Cadence layout design tools.

2. Objectives

By the end of this project, you will be able to use professional computer-aided VLSI design tools to (1) Simulate your written HDL code, (2) Synthesize your design, (3) Place and Route your synthesized HDL , (4) Use a pad frame to build a full chip, and (5) export your final GDS file.

3. References

1) Notes from our class
2) References introduced for our course
3) Our tutorial for RTL-GDS flow design (Cadence Tutorial)

NOTE ***Process:*** *In this lab, you will use the Cadence 45nm GPDK CMOS process provided in the lab.*

CAD Tools: *Activities in this lab, including synthesis HDL, automatic layout design, DRC check, circuit extraction, and different kinds of circuit simulations will be done using Cadence® tools.*

DESCRIPTION ***Function:*** Figure 1 shows a simplified circuit diagram of the ALU you will design in this project. In this diagram, S_3~S_0 are function select bits, *A* and *B* are 32-bit operands, and C_{in} is the carry-in bit. In addition, the circuit has two serial inputs, $D_{in,L}$ and $D_{in,R}$, which are the input bits for "shift left" and "shift right" operations, respectively. The circuit also has a 32-bit parallel output, *F*, and a carry-out bit, C_{out}. Functional description of the ALU is presented in Table 1.

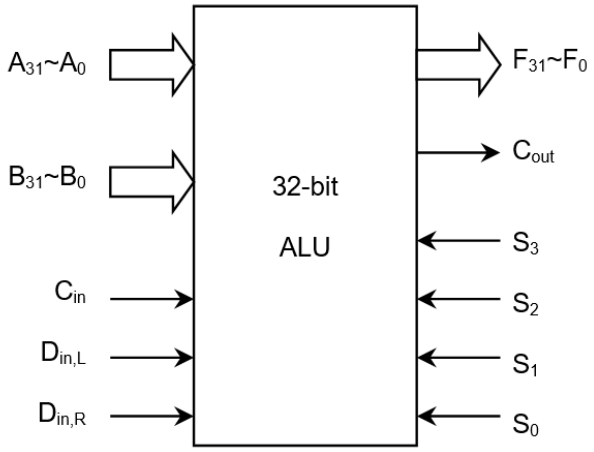

Figure 1: Input/output diagram of the target ALU

Table 1: Functional description of the ALU

Operation Select						
S_3	**S_2**	**S_1**	**S_0**	**C_{in}**	**Operation**	**Function**
0	0	0	0	0	F=A	Transfer A
0	0	0	0	1	F=A+1	Increment A
0	0	0	1	0	F=A+B	Addition
0	0	0	1	1	F=A+B+1	Add with carry
0	0	1	0	0	F=A+B'	Subtract with borrow
0	0	1	0	1	F=A+B'+1	Subtraction
0	0	1	1	0	F=A-1	Decrement A
0	0	1	1	1	F=A	Transfer A
0	1	0	0	X	F=A.B	AND
0	1	0	1	X	F=A–B	OR
0	1	1	0	X	F=A/B	XOR
0	1	1	1	X	F=A'	Complement A
1	0	X	X	X	F=shr A	Shift right A (1 bit) into F
1	1	X	X	X	F=shl A	Shift left A (1 bit) into F

Structure: The target 32-bit ALU is to be designed based on a *bit-slice* scheme. A bit-slice structure is a modular design approach used in digital circuits, especially in ALUs, processors, and DSP architectures. This structure divides a complex function into smaller, identical processing units that each handle a fixed number of bits (*e.g.*, 1-bit, 4-bit, or 8-bit). These slices are then stacked together to create a larger system capable of processing wider data words. Key features of the bit-slice scheme can be listed as follows:

Modularity – Each slice performs computations on a subset of the total word width (*e.g.*, 1 bit, 4 bits, or 8 bits out of a total of N bits).

Scalability – Multiple slices can be combined to create wider data paths (e.g., stacking four 4-bit slices for a 16-bit processor).

Reuse of Circuit Blocks – Instead of designing a large circuit from scratch, smaller units are designed and replicated.

Simplification of Design Complexity – Instead of handling an entire 32-bit or 32-bit computation at once, designers work with smaller, repeatable units.

Early microprocessors (before full-chip integration), custom ALU designs (e.g., in DSPs and FPGAs), and high-speed parallel processing architectures are examples of the applications of bit-slice structures.

In our case in this project, a 1-bit bit-slice ALU is designed as a module containing logic gates, adders, and multiplexers to handle the basic operations listed in Table 1. To build a 32-bit ALU, thirty-two such slices are chained together.

Part 1: 1-bit ALU

Simplified schematic of the 1-bit ALU you will design in this project is shown in Figure 2(a). The circuit comprises three sub-modules: an arithmetic unit, a logic unit, and a multiplexer. Internal circuits for the arithmetic and logic units are shown in Figure 2(b)&(c).

i) Write ***HDL codes*** implementing the functions of the arithmetic unit, logic unit, and the multiplexer unit, separately. Simulate the HDL design of the units and verify the operation of each block for their designated functions. Test vectors should consider all possible combinations of the associated inputs. Report the pre-synthesis simulation results (including truth-table and simulated waveforms).

ii) Use your designed sub-modules to create the 1-bit ALU (with a structural design). Simulate the circuit and verify its functionality.

iii) Synthesize the 1-bit ALU and report the resulting area and power.

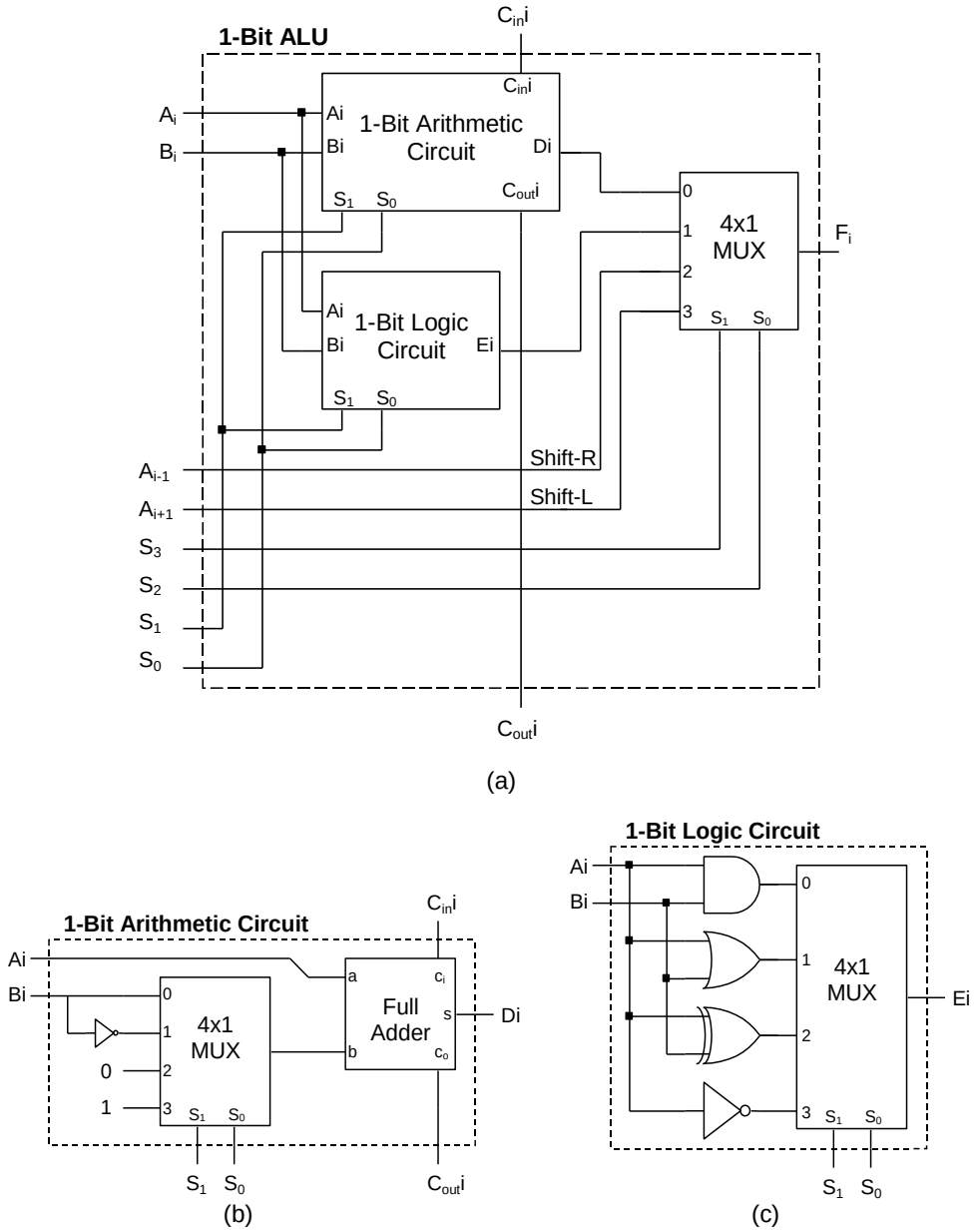

Figure 2: (a) Circuit schematic of the 1-bit ALU (b)&(c) Internal circuits for the arithmetic and logic units

iv) Generate the layout of the 1-bit ALU using the *Place and Route* (*PnR*) tool in Cadence Innovus.

v) Perform DRC and connectivity checks in Innovus.

vi) Report the estimation of the highest speed of the circuit.

PART2: 32-BIT ALU

i) **HDL Design:** Design (Write a HDL code for) a 32-bit ALU formed in two ways: (a) by instantiating thirty-two 1-bit ALU cells and properly connecting them together, and (b) by writing a HDL code for a 32-bit ALU in one single step. For this design, your code does not necessarily have to be written in a modular way. It can take the 32-bit operands as 32-bit binary numbers to model the operations in a behavioral manner.

ii) Simulate both designs in (i) and verify their functionality.

iii) Synthesize both designs and include the synthesis reports in your project report.

iv) Use Innovus to perform Place and Route separately for each one of the synthesized Verilog designs.

v) Compare the two cores in terms of area, power, and speed.

vi) Perform DRC and connectivity checks.

vii) Export a GDS file for each design.

viii) For the core that is smaller in area, use the pad frame provided (step-by-step guidelines are presented on eclass), place the core layout inside it, and generate the complete layout for your chip. Report the complete layout, total area, and maximum speed of the chip. Provide a pad configuration diagram for your chip. Export your final layout (complete chip) as a

---

Due Date: December 14, 2025 (23:59 EST)

Project Deliverables:

1. Legible and Organized PDF Report:

- **Format**: PDF only *(Word documents will not receive marks.)*
- **Contents**:
    - Simulation waveforms clearly illustrating the results.
    - Detailed coverage of all specified requirements.
    - Short and descriptive explanations accompanying each waveform and image.

2. Complete HDL and Synthesis/PnR Files:

Include all files related to your design and synthesis workflow:

- **HDL Source Files**:
    - Verilog or VHDL design files.
    - Corresponding test benches.
- **Synthesized Netlists**:
    - Clearly labeled synthesized netlist files.
- **Synthesis Reports**:
    - Comprehensive reports detailing synthesis results.
- **Innovus Timing Reports**:
    - Complete timing analysis reports exported from Innovus.
- **Innovus Exported GDS Files**:
    - Clearly named GDS files exported from Innovus.
    - A full-chip GDS file that includes the pad frame.
    - Ensure all GDS files are named clearly and distinctly to be easily identifiable.