KING'S College LONDON

Hardware Design Lab

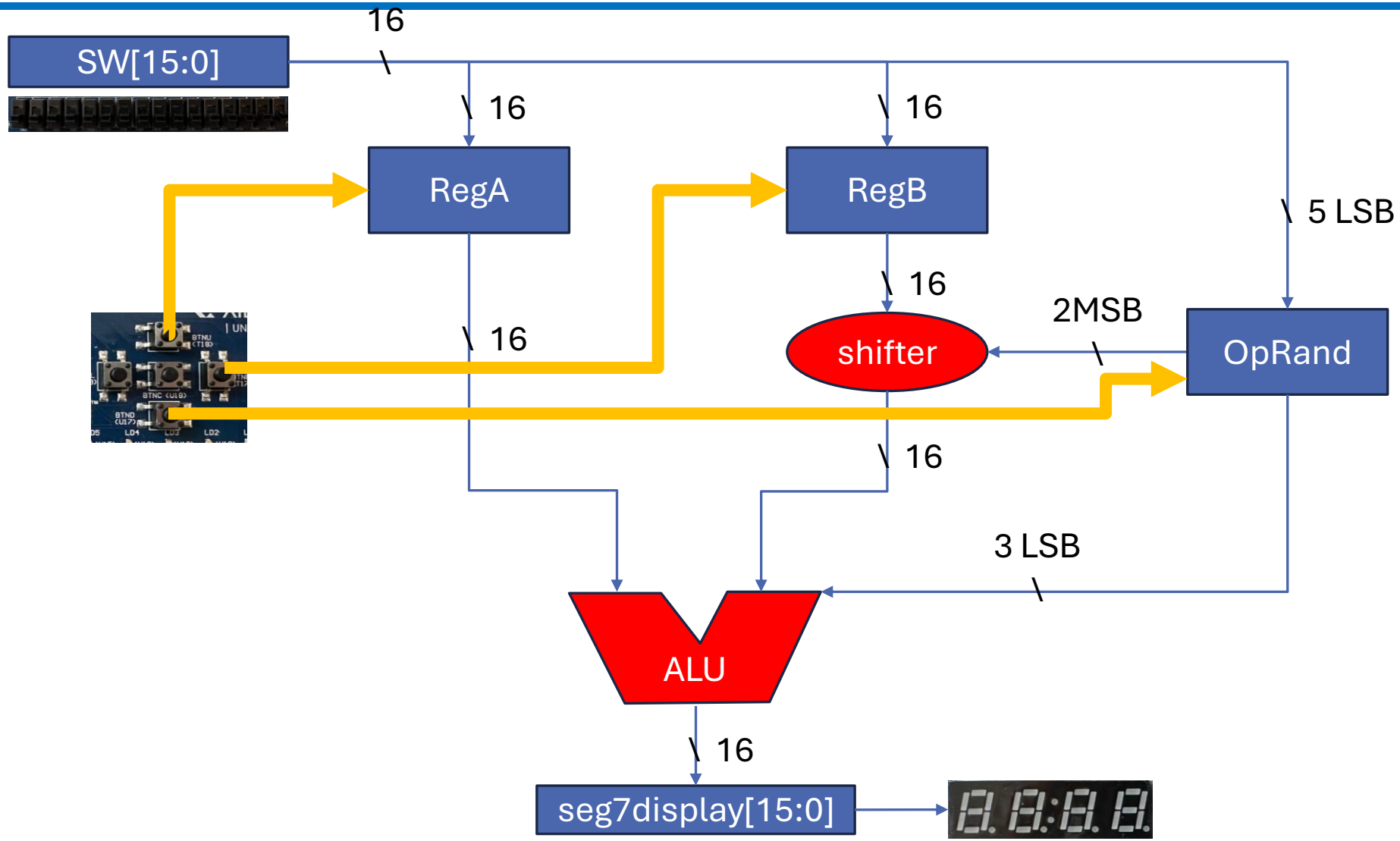
Task1: ALU Design

Instructor: Dr. Haiyu Mao
TA: Zihao Pu
Ali Alsarraf
Stephen Johannesson

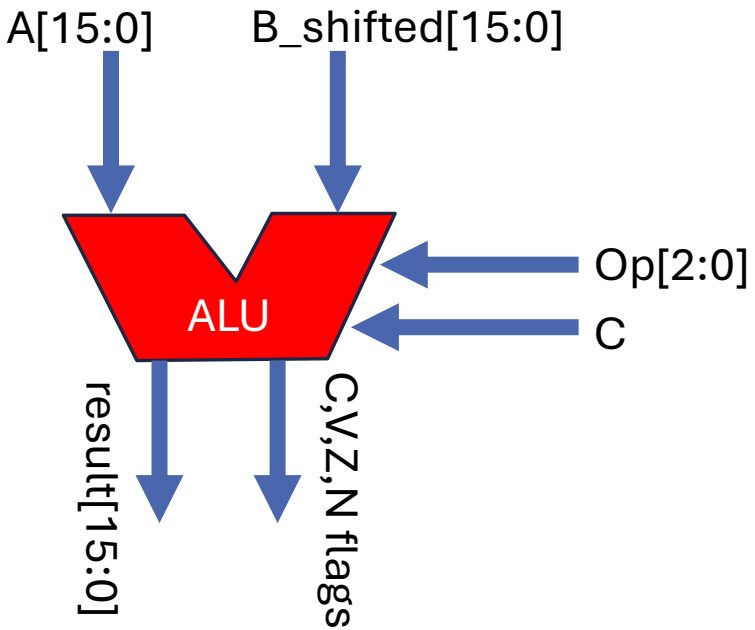
Nov.25, 2025

Gratefully acknowledge:
Prof. Onur Mutlu (ETH)
Dr. Yair Linn (TRIUMF Canada)
Prof. Tor Aamodt (UBC)

Task 1: Structure



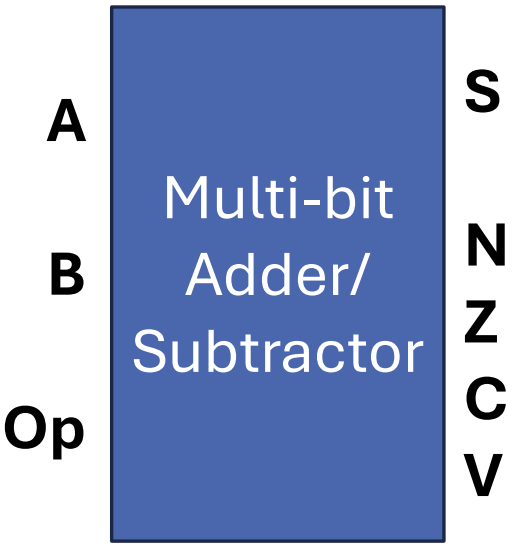
Task1: ALU Design and Opcode



ADD	000	A + B_shifted	Update Z, N, C, V
SUB	001	A - B_shifted	Update Z, N, C, V
ADDC	010	A + B_shifted + C	Update Z, N, C, V
SUBC	011	A - B_shifted - Borrow	Update Z, N, C, V
AND	100	A & B_shifted	Update Z, N
ANDBB	101	A & (~(B_shifted)	Update Z, N
OR	110	A | B_shifted	Update Z, N
ORBB	111	A | (~(B_shifted)	Update Z, N

ALU Status Flags: **N,Z,C,V**

Sign	Name	Usage	Source
N	Negative	1 if result is negative, 0 if not	S_n
Z	Zero	1 if result is 0, 0 if not	$\lvert \boldsymbol{S}$
C	Carry	1 if there is a carry-out; 0 if not	C_n
V	oVerflow	1 if there is overflow; 0 if not	$C_n \oplus C_{n+1}$



ALU: Idea of oVerflow

- e.g 8-bit number: value is between -128 to + 127. Calculate 70+80， result is negative! -> overflow happens.

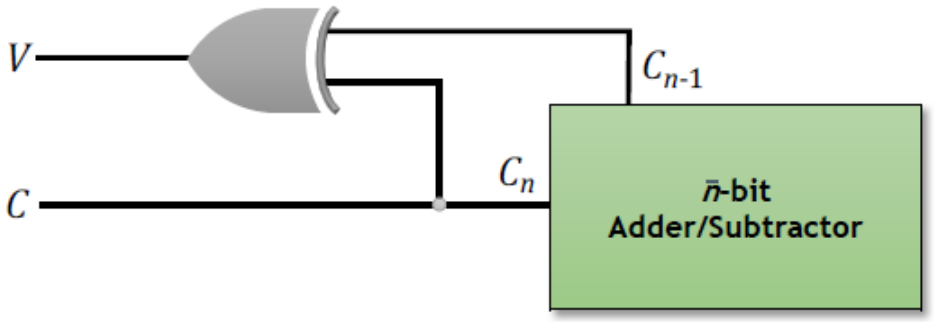
- For unsigned addition:
- $C = 0$: No overflow
- $C = 1$: Overflow

$$
\begin{array}{r}
0\ 1\ 0\ 0\ 0\ 1\ 1\ 0 \\
+\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0 \\
\hline
1\ 0\ 0\ 1\ 0\ 1\ 1\ 0
\end{array}
$$

- Defne V(Overflow): for **signed** addition/subtraction:
- $V = 0$: No overflow
- $V = 1$: Overflow
- Question: how to design V flag?

ALU: oVerflow circuit deign

- When $C_n \neq C_{n+1}$, overflow exists



- Other way to design oVerflow:
  - For adding: if both inputs are positive, but result is negative; or if both inputs are negative, but result is positive.
  - For subtracting: if A is positive, B is negative, but result is negative; or if A is negative, B is positive, but result is positive.

as	bs	cis	qs	cos	ovf	comment
0	0	0	0	0	0	Both inputs positive, both carrys 0, no overflow
0	0	1	1	0	1	Both inputs positive, carry in 1, overflow
0	1	0	1	0	0	Input signs different, carry in 0, no overflow
0	1	1	0	1	0	Input signs different, carry in 1, no overflow
1	1	0	0	1	1	Both inputs negative, carry in 0, overflow
1	1	1	1	1	0	Both inputs negative, carry in 1, no overflow

Table 10.3: Cases for inputs and carry into sign bit of adder to detect overflow. Columns show sign bit of a and b (as and bs) carry into and out of sign bit (cis and cos) and output of sign bit (qs). Overflow only occurs if the carrys into and out of the sign bit are different.

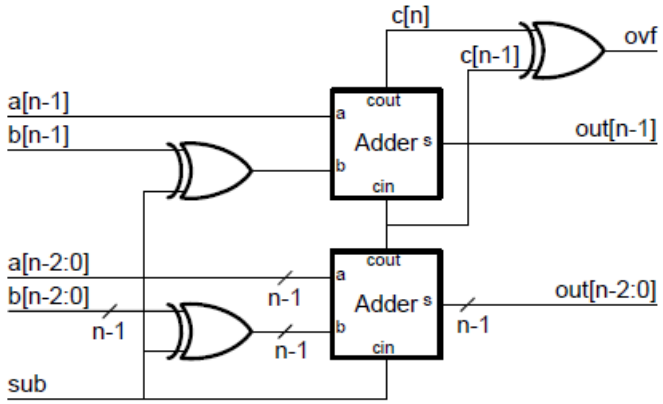
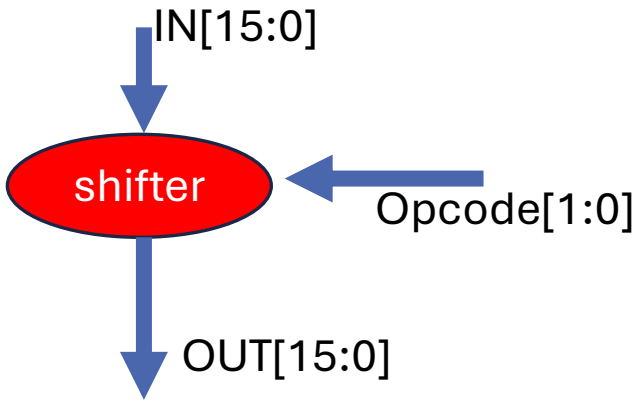


Figure 10.12: A 2's complement add/subtract unit with overflow detection based on carry in and out of the last bit.

Shifter: Opcodes

Shift Code (S)	Operation	Description
00	Pass-through	Out = B (No change)
01	Logical Left	Out = B << 1 (Multiply by 2)
10	Logical Right	Out = B >> 1 (Unsigned Divide by 2)
11	Arithmetic Right	Out = B >>> 1 (Signed Divide by 2)




```
e.g.    11110000    0xf0
LSL:    11100000    0xe0
LSR:    01111000    0x78
ASR:    11111000    0xf7
```

Task 1: Tasks

- You will need to implement the **ALU** and **Shifter** in Verilog/SystemVerilog.
- You can use the provided demo project and fill your own code.
- You need to write your own testbench to prove it works.
- You should download the project to FPGA and show it works.

TASK 1: ALU

ADD: A+B

ADDC: A+B+Cin

SUB: A+$\bar{B}$+1

SUBC: A+$\bar{B}$+Cin

00	ADD: Cin = 0
01	SUB: Cin = 1
10	ADDC: Cin = C
11	SUBC: Cin = $\bar{C}$

	2	1	0				
A+B+0	0	0	0	A&B	1	0	0
A+$\bar{B}$+1	0	0	1	A&$\bar{B}$	1	0	1
A+B+Cin	0	1	0	A|B	1	1	0
A+$\bar{B}$+Cin	0	1	1	A|$\bar{B}$	1	1	1