TSEA26 Tutorial 2. Arithmetic Logic Unit

Frans Skarman

November 17, 2021

More lab info

Suggested schedule

- **Session 1–2**: Lab 1
- **Session 3–5**: Lab 2
- **Session 6–8**: Lab 3
- **Session 9–10**: Lab 4

Remote work

- `ssh -YC ssh.edu.liu.se` or thinlinc
- Then `ssh -YC muxenX-0YY` (select muxen 1 or 2 + computer)

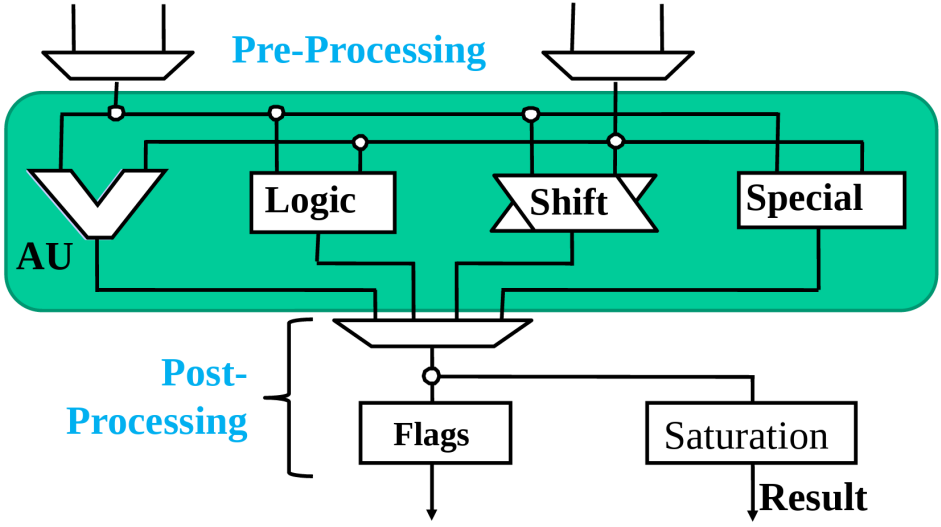
Arithmetic and Logic Unit (ALU)

- Key component in a processor datapath
- Usually receives all operands from register file or immediate values
- Latency 1 clock cycle
- Usually 1 guard bit

Key components of an ALU

- Arithmetic unit (`Add`, `Sub`, `Min`, etc.)
- Logic unit (`And`, `OR`, etc.)
- Shifter (`lsr`, `asl`, etc.)

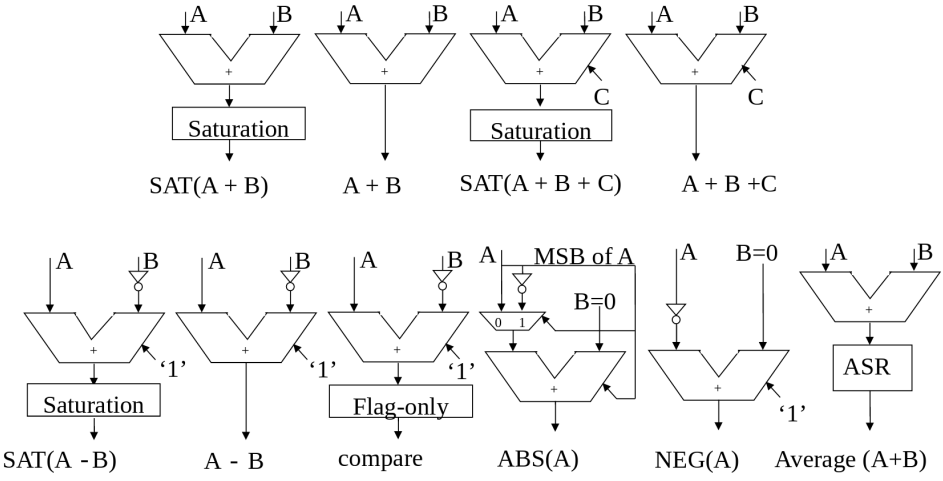
ALU overview



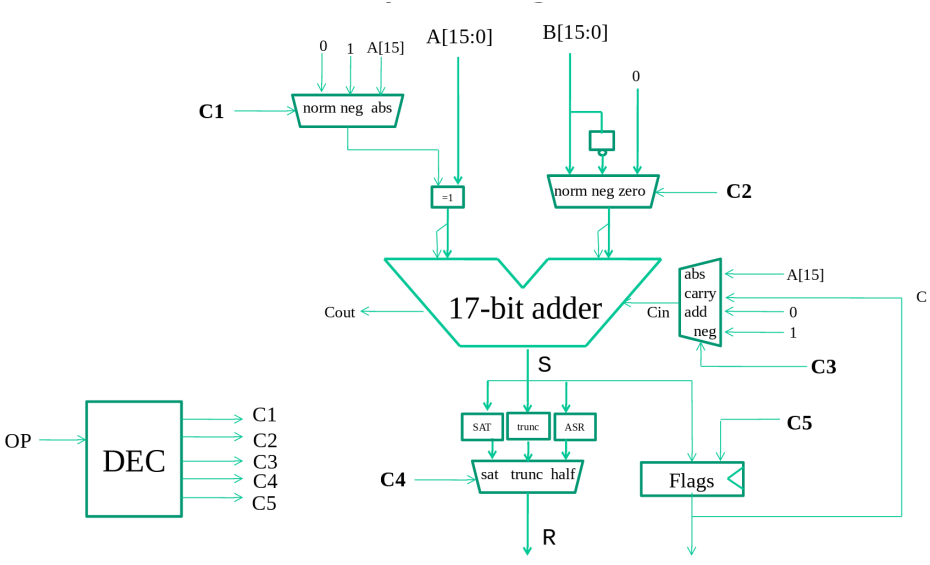
Example: Design a small ALU

OP	Operation	Saturation	Update flags
0	$A + B$	yes	yes
1	$A + B$	no	yes
2	$A + B + C_{in}$	yes	yes
3	$A + B + C_{in}$	no	yes
4	$A - B$	yes	yes
5	$A - B$	no	yes
6	compare A and B	yes	yes
7	$\|A\|$	–	yes
8	$-A$	–	yes
9	$(A + B)/2$	–	yes
10	*NOP*	–	no

Required computations

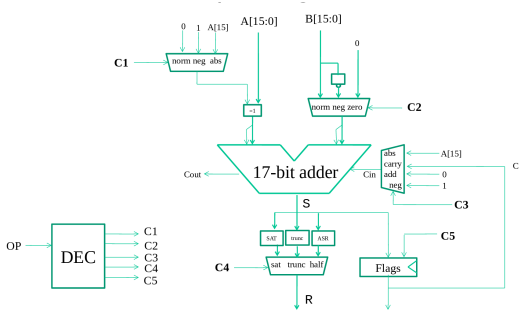


Combined hardware



Control signals

OP	C1	C2	C3	C4	C5
0	norm	norm	add	sat	1
1	norm	norm	add	trunc	1
2	norm	norm	carry	sat	1
3	norm	norm	carry	trunc	1
4	norm	neg	neg	trunc	1
5	norm	neg	neg	sat	1
6	norm	neg	neg	-	1
7	abs	zero	abs	trunc	1
8	neg	zero	neg	trunc	1
9	norm	norm	neg	half	1
10	-	-	-	-	0

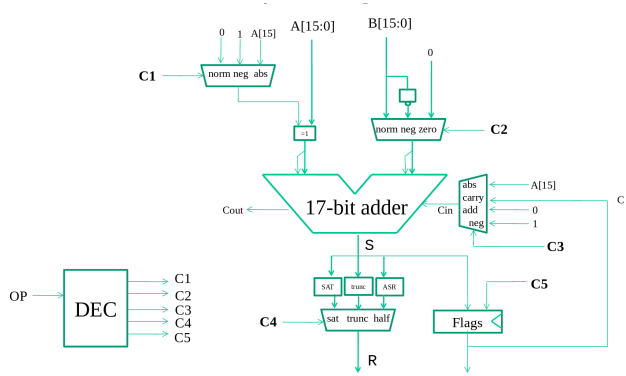


Black boxes

```
// Flags
always @(posedge clk) begin
    if (c5) begin
        C <= Cout;
        Z <= R == 0;
        N <= R[15];
        V <= (S[16] != S[15]);
    end
end

assign R = S[16:1] // ASR
assign R = S[15:0] // Trunc

// sat
always @(*) begin
   if (S[16]==S[15])
        R <= S[15:0];
   else if (S[16]==0)
        R <= 16'h7fff;
   else
        R <= 16'h8000;
end
```



About Lab 2

- Implement (parts of) ALU and MAC unit for senior
- Use Verilog or VHDL
- Skeleton files provided in `lab2-3`
- All info in lab manual
  - Chapter 0 covers useful commands etc.
  - Chapter 2 covers lab instructions
  - **Read through this carefully**

About Lab 2

Files to write:

- Hardware (VHDL/Verilog)
  - `saturation.vhd`
  - `mac_dp.vhd`
  - `adder_ctrl.vhd`
  - `min_max_ctrl.vhd`
  - Or corresponding verilog
- Software (Assembly)
  - `saturation.asm`
  - `rounding_vector.asm`
  - `alu_test.asm`

Lab 2 Workflow

1. Run SW in `srsim` to get reference output
2. Run SW in simulated HW using `vsim`
3. Compare output
4. Check coverage. Are you testing all HW?

Steps 1–3 are handled automatically by `Makefile` (I think)

Lab 2 - Hints

It is very important that your (synthesizable) code in Lab 2 does not contain any latches

When maximizing your coverage, make sure that you actually output the result of what you are testing

The following code will get coverage for both the mulss and muluu instruction but will not actually verify that the result is correct of mulss

```
mulss r0,r1,r2 ; Test mulss
muluu r0,r1,r2 ; Test muluu
call outputacr_to_testport
```

Lab 2 – Synthesis

- To synthesize your design, you need to login to
- `only-da.ad.liu.se` (for license reasons)
- Information on lab web page
- Make sure you use the latest version of lab `2-3-files` (or you may need to change)
- The latest lab manual contains updated information

Exercises

Exercises 1, 2, 3, and 4 from ALU exercises