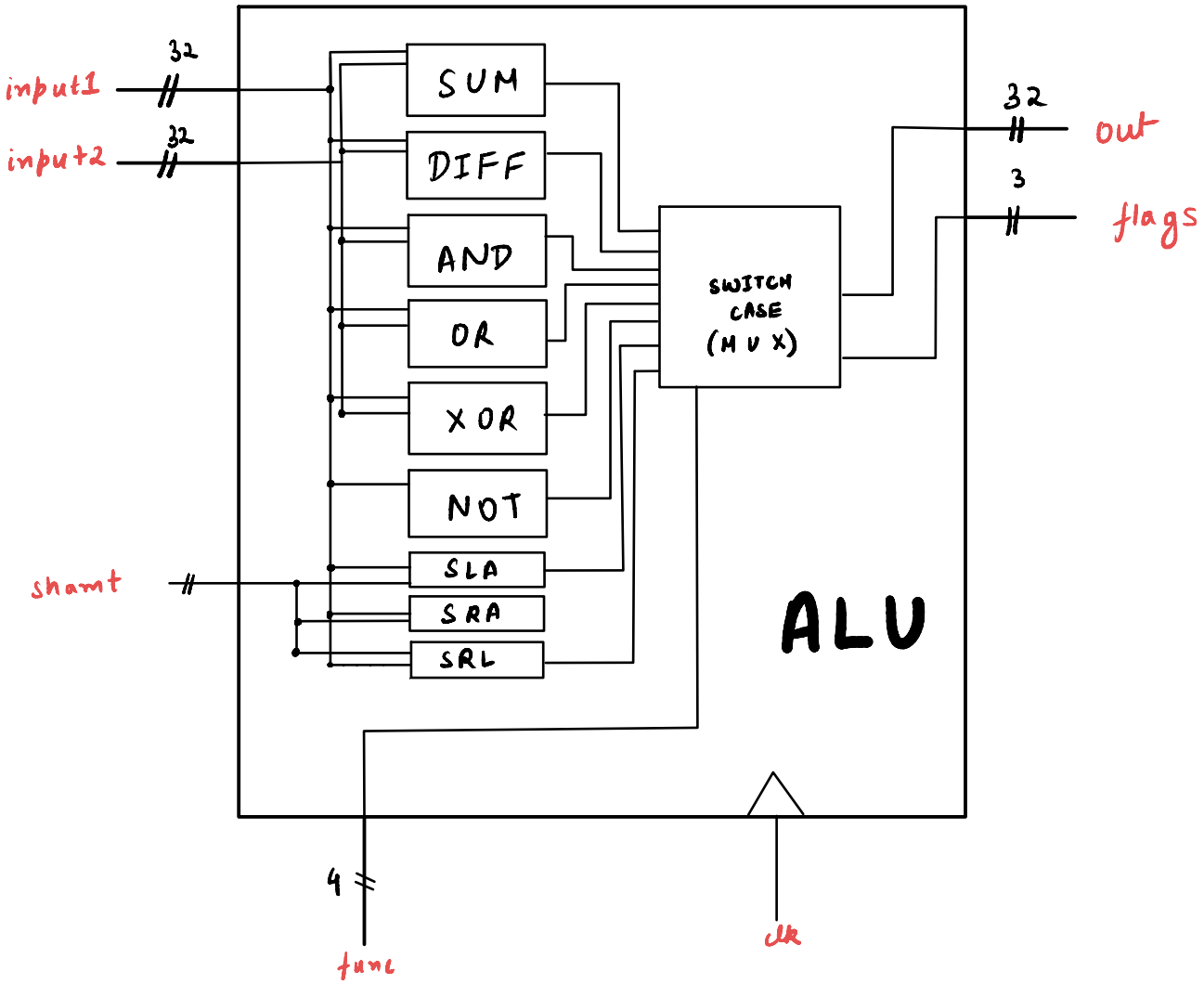
Verilog Assignment 7 ALU Design
Tanishq Prasad (21CS30054)
Krish Khimasia (21CS10037)
Group No. 17

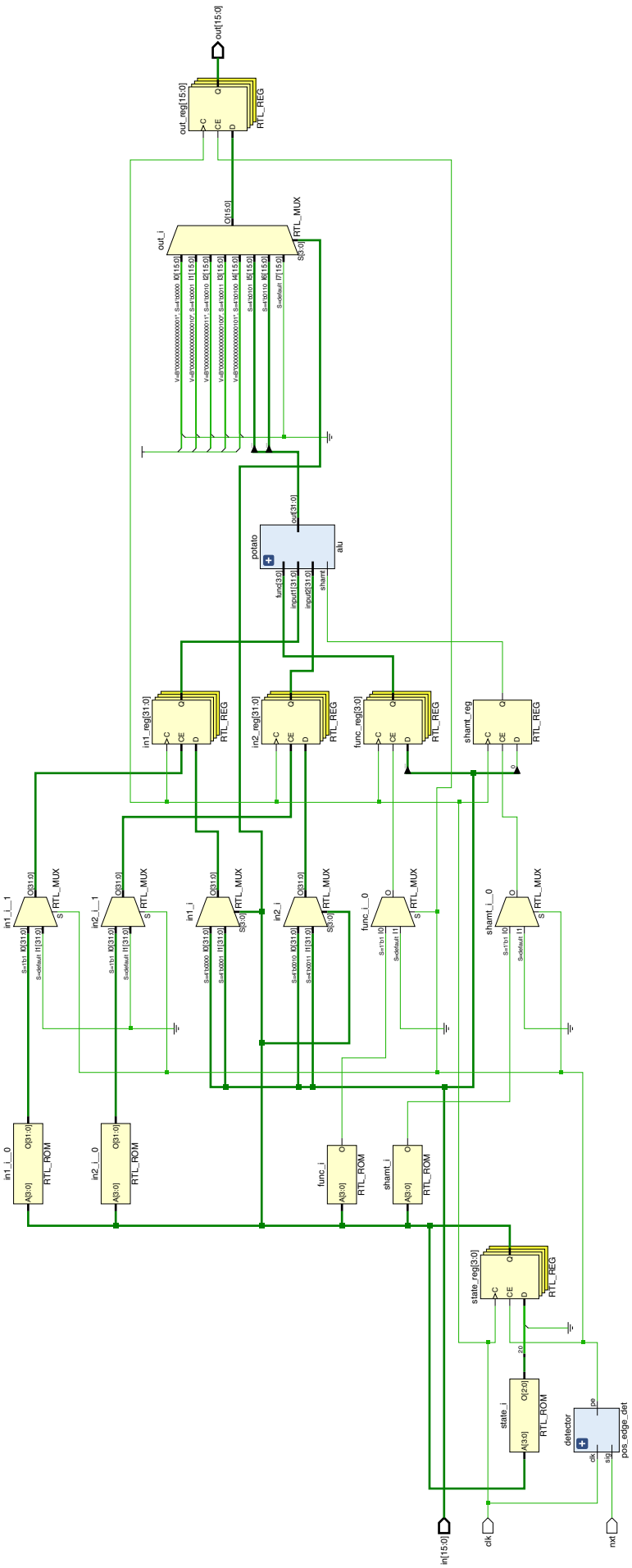
ALU Design:

Our ALU module has 32-bit input1 (for the first input), 32-bit input2 (for the second input), 4-bit func (for choice of operation) and 1-bit shamt (for shift amount) and clk (for the clock). All the inputs are wires. Output is stored in the 32-bit out, which is a register. There is a 3-bit flags register whose zeroth bit shows whether input1 is zero, the first bit shows the sign for input1, and the second bit shows carry or borrow for the SUM and DIFF operations, respectively.

We have implemented a 4-bit CLA in a module called CLA_4bit_augmented. Now we calculate the look-ahead carry in the LookaheadCarryUnit and called 4 instances of CLA_4bit_augmented to make a 16-bit CLA with look-ahead carry as a module named CLA_16bit_withLCU.

- SUM: We called two instances of the CLA_16bit_withLCU cla1 and cla2 and concatenated them in a ripple fashion to calculate the sum of the two 32-bit inputs.
- DIFF: We took the complement of input2 and stored it in not_input2. We called two instances of the CLA_16bit_withLCU cla3 and cla4 and concatenated them in a ripple fashion to calculate the sum of the two 32-bit inputs. We set the c_in for cla3 as 1 to turn the 1’s complement of input2 to the 2’s complement of input2.
- AND: Computes input1 & input2; That is the bitwise-AND for the inputs.
- OR: Computes input1 | input2; That is the bitwise-OR for the inputs.
- XOR: Computes input1 ^ input2; That is the bitwise-XOR for the inputs.
- NOT: Computes ~input1; That is the bitwise-NOT for the input.
- SLA: Computes input1 << shamt; That is left-shift input1 by shamt.
- SRA: Computes input1 >> shamt; That is right-shift input1 by shamt logically.
- SRL: Computes input1 >>> shamt; That is right-shift input1 by shamt arithmetically (that is conserving the sign).


input1
32
input2
32
SUM
DIFF
AND
OR
XOR
NOT
SLA
SRA
SRL
shamt
SWITCH CASE (MUX)
32
out
3
flags
ALU
4
func
clk
CONTROL LINE


out[15:0]
out_reg[15:0]
RTL_REG
O[15:0]
out_i
RTL_MUX
S[5:0]
potato
alu
out[31:0]
func[3:0]
input1[31:0]
input2[31:0]
shamt
in1_reg[31:0]
in2_reg[31:0]
func_reg[3:0]
shamt_reg
in1_i__1
in2_i__1
in1_i
in2_i
func_i__0
shamt_i__0
in1_i__0
in2_i__0
func_i
shamt_i
RTL_ROM
A[5:0]
state_reg[3:0]
state_i
O[2:0]
detector
pe
pos_edge_det
clk
sig
in[15:0]
clk
nxt