IN3160 Oblig 7

Chris René Lerner

I am not delivering timing summary reports, since it was stated in discourse that it was not required.

Task 1

Adding two binary numbers end up with a maximum of 1 overflow bit. Take the worst case as an example where every bit gets shifted one place to the right:

$$\begin{array}{rr} & 1111\ 1111\ 1111\ 1111 \\ + & 1111\ 1111\ 1111\ 1111 \\ \hline = & 1\ 1111\ 1111\ 1111\ 1110 \end{array} \tag{1}$$

Therefore, the answer is 17 bits.

Task 2

Multiplying the two numbers ends up with a less obvious amount of bits. Lets take the worst case, do a multiplication in base 10, and converting to binary.

$$(1111\ 1111\ 1111\ 1111)_2 = (65\ 535)_{10} \tag{2}$$

$$(65\ 535)_{10} \cdot (65\ 535)_{10} = (4\ 294\ 836\ 225)_{10} \tag{3}$$

$$(4\ 294\ 836\ 225)_{10} = (FFFE0001)_{16} = (1111\ 1111\ 1111\ 1110\ 0000\ 0000\ 0000\ 0001)_2 \tag{4}$$

The end result of the worst case multiplication is 32 bits.

Task 3

Say all four numbers are FFFF. a + b results in 17 bits. c + d results in 17 bits.

$$\begin{array}{rr} & 1\ 1111\ 1111\ 1111\ 1110 \\ \hline + & 1\ 1111\ 1111\ 1111\ 1110 \\ \hline = & 11\ 1111\ 1111\ 1111\ 1100 \end{array} \tag{5}$$

The addition of these two 17 bit sums will then result in 3FFFC which is 18 bits.

Task 4

We also assume that all the numbers are FFFF, including e. We then end up with multiplying 18 bits with 16 bits. The multiplication goes like this:

$$(3\ FFFC)_{16} \cdot (FFFF)_{16} = (262\ 140)_{10} \cdot (65\ 535)_{10}$$

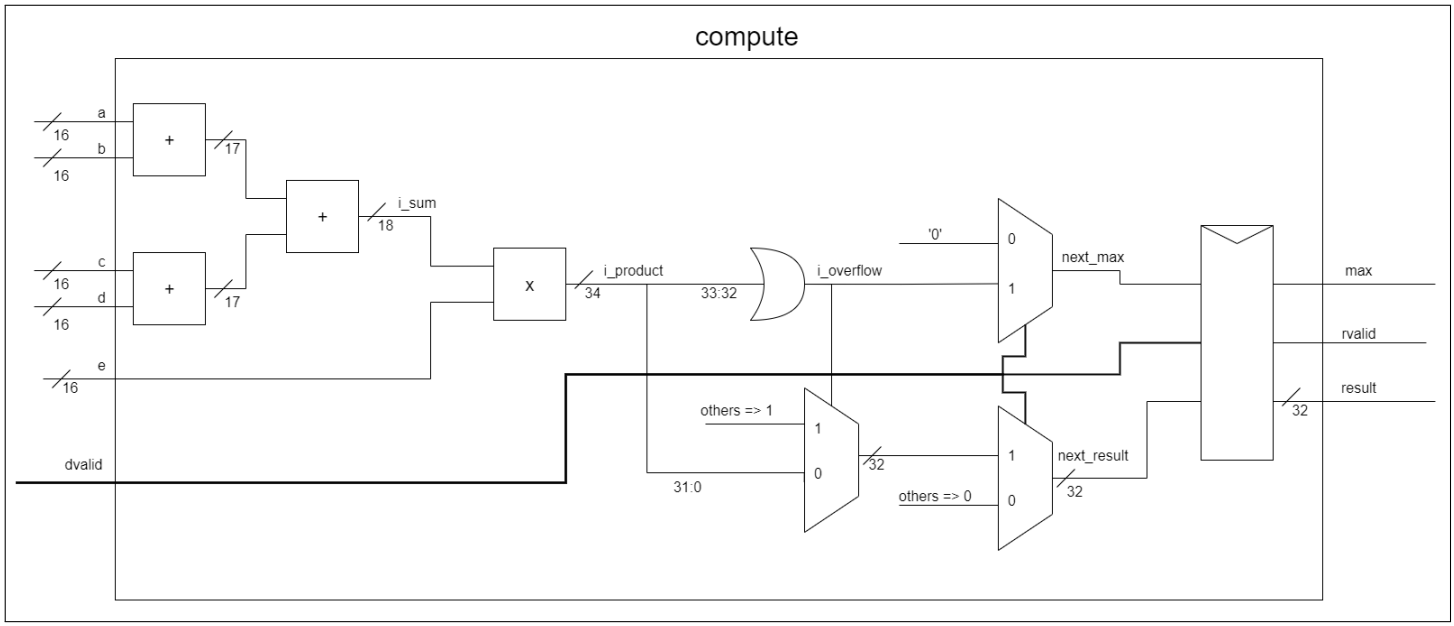
$$= (17\ 179\ 344\ 900)_{10} = (3\ FFF8\ 0004)_{16}$$

$$= (11\ 1111\ 1111\ 1111\ 1000\ 0000\ 0000\ 0000\ 0100)_2 \tag{6}$$

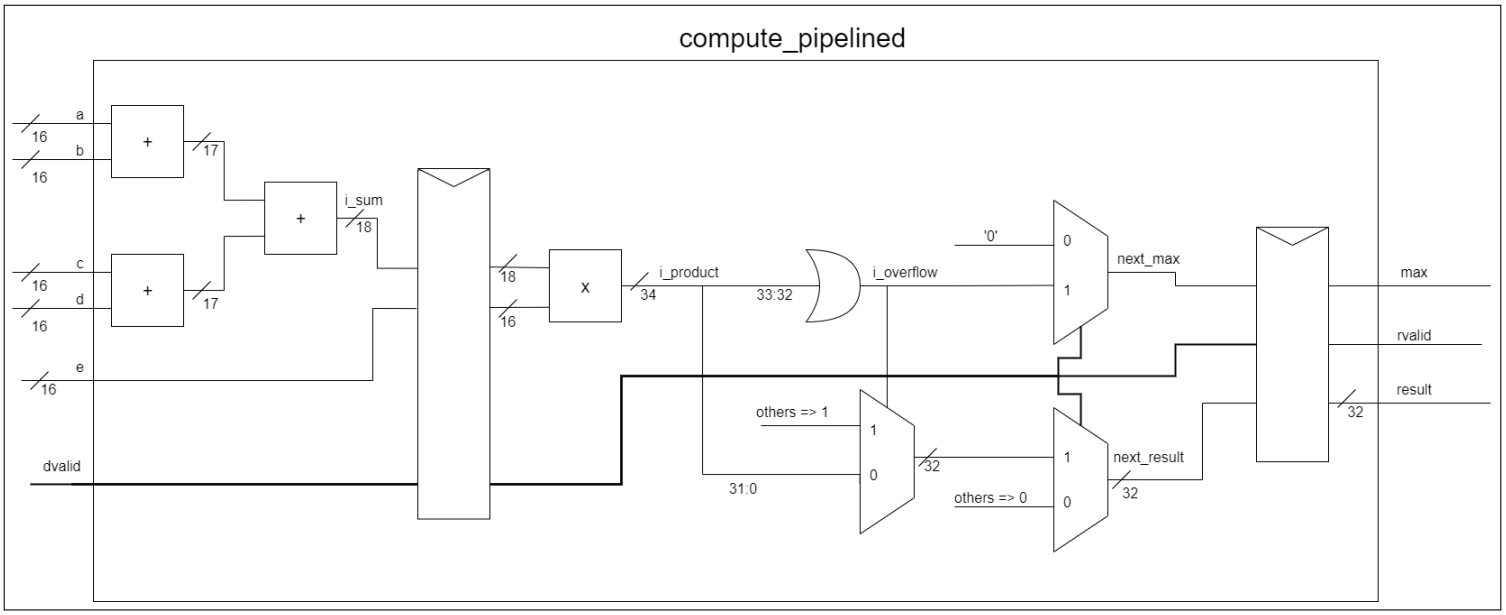
The resulting binary number is 34 bits long.

Task 5

a)


compute
a
16
b
16
+
17
+
i_sum
18
c
16
d
16
+
17
e
16
x
i_product
34
33:32
i_overflow
'0'
0
1
next_max
max
rvalid
result
32
dvalid
others => 1
1
0
32
31:0
others => 0
1
0
next_result
32

b)



Task 7

Since there is 1 flip-flop per flip-flopped bit, we have 32 flip-flops for result, 1 flip-flop for rvalid, and 1 flip-flop for max, with a total of 34 flip-flops for the module compute. I have not used the synthesized design to get this information.

Task 8

We have 32 flip-flops for result, 1 flip-flop for rvalid, and 1 flip-flop for max, just like with the module compute. In addition, there are the flip-flops used for the pipeline. 18 flip-flops for i_sum, 16 flip-flops for e, and 1 flip-flop for dvalid. With a total of 34 + 18 + 16 + 1 = 69 flip-flops for the module compute_pipelined. I have not used the synthesized design to get this information.