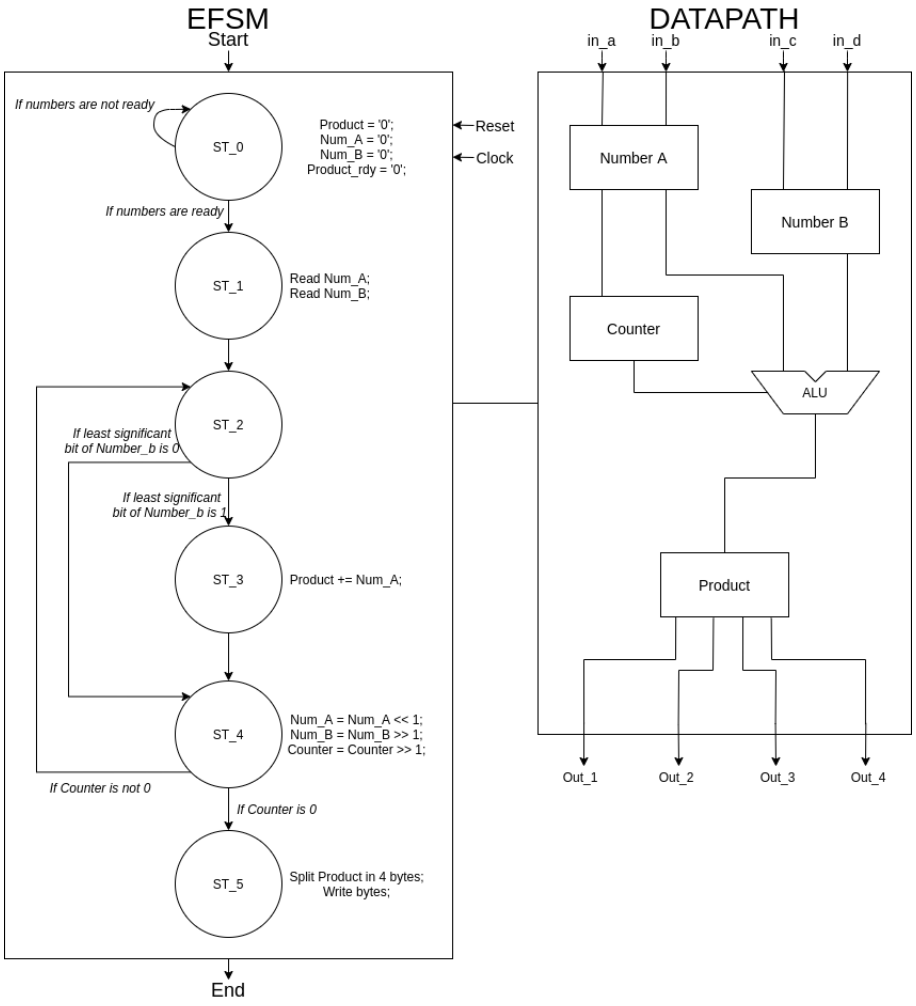
Report 02

Filippo Nevi - VR458510

October 2020

1 RTL

1.1 Diagram



1.2 Results visualization

The values returned from the `mult_RTL` module are correct, but they get approximated when printed to the terminal by the function `get_float_value` that is called in the testbench: this function converts a 32-bit *integer* value to a *float* with 16 bits of integer part and 16 bits of rational part.

It prints the numbers correctly with few fractional bits, but with more than 8 bits it starts approximating the values.

2 TLM

2.1 Approach and motivations

I've chosen to implement this algorithm using the **Loosely-Timed** approach because I think the system doesn't need more than two synchronization points: *request* and *response*, since there aren't any other significant situations to be modeled.

Once the initiator starts the request it will wait the response from the target, because I assume the results are needed for the computation and can't go any further without it.

3 Computation times

In order to compare the computation times, I decided to run both the implementations 10 times, using the command `time`.

The average running time of the RTL executable is $0.013s$, and the average time of the TLM part is $0.004s$. These results were **expected**, because the first implementation is more focused on timing accuracy and events, while the main goal of the second one is to test the functionality implementation.