Embedded Systems Design

First Assignment

First Assignment

- First Assignment composed by 3 main parts:
  - RTL Modeling
    - VHDL/Verilog
    - SystemC
  - RTL Synthesis
    - VHDL/Verilog
  - High-Level Synthesis
    - Choose between
      - C simple floating point multiplication
      - SystemC or C algorithm HLS

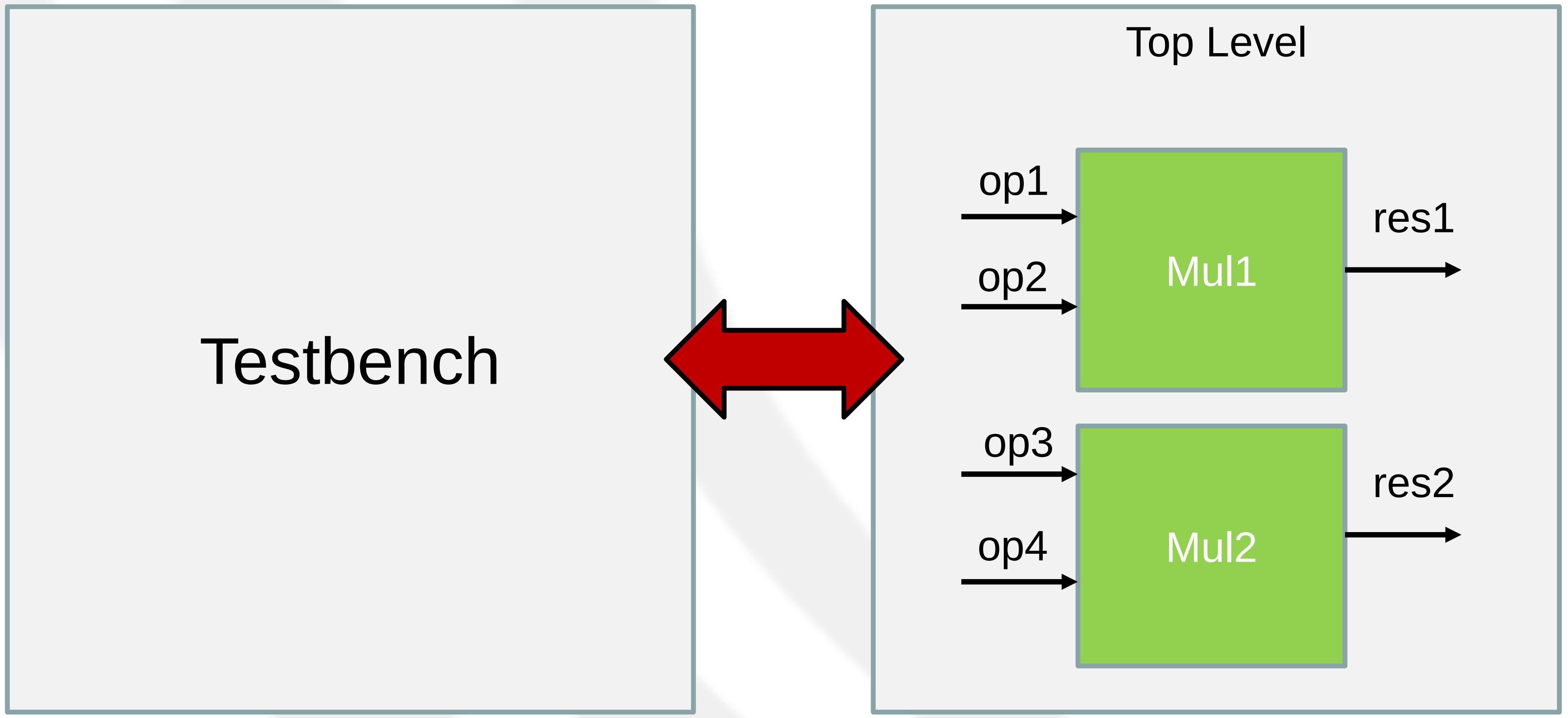
Delivery Deadline

Wednesday, 8th January 23:59

Embedded Systems Design - Assignment

1.1 - RTL Modeling

Floating-point multiplications HW accelerator (Single Precision)



Floating-point multiplications HW accelerator

- The HW accelerator, **developed in VHDL/Verilog and SystemC**, must perform two floating point single precision multiplications.
  - One multiplier in Verilog
  - One multiplier in VHDL
  - You can choose the language for the Top Level
- Input: the four operands necessary for the two multiplication
  - 32 bit floating point
- Output: the results of the two multiplication
  - 32 bit floating point
- **Pay attention on the total number of FPGA ports (125)**
- The component will implement an handshake protocol
- The component will receive the four input serialized
  - One or two operators per clock cycle
- The component will return the two results serialized
  - One result per clock cycle
- The implementation must be hierarchical
  - The two multiplication will be computed by two different sub-components
  - Define (and report) the EFSM of the "top-level" component
  - Define (and report) the EFSM of the sub-components

Floating Point Multiplication

- IEEE 754 single-precision binary floating-point format:
  - Sign bit: 1 bit
  - Exponent width: 8 bits
    - Offset-binary representation: offset 127
    - Exponents with special meanings (M will be the mantissa):
      - 0x00: signed zero (if $M = 0$) and subnormals (if $M \neq 0$)
      - 0xff: infinity (if $M = 0$) and NaNs (if $M \neq 0$)
      - It shouldn't be a problem for the Multiplication!
  - Significand precision: 24 bits (23 explicitly stored)
- Given a biased esponent e and a 23-bit fraction, the real value is:

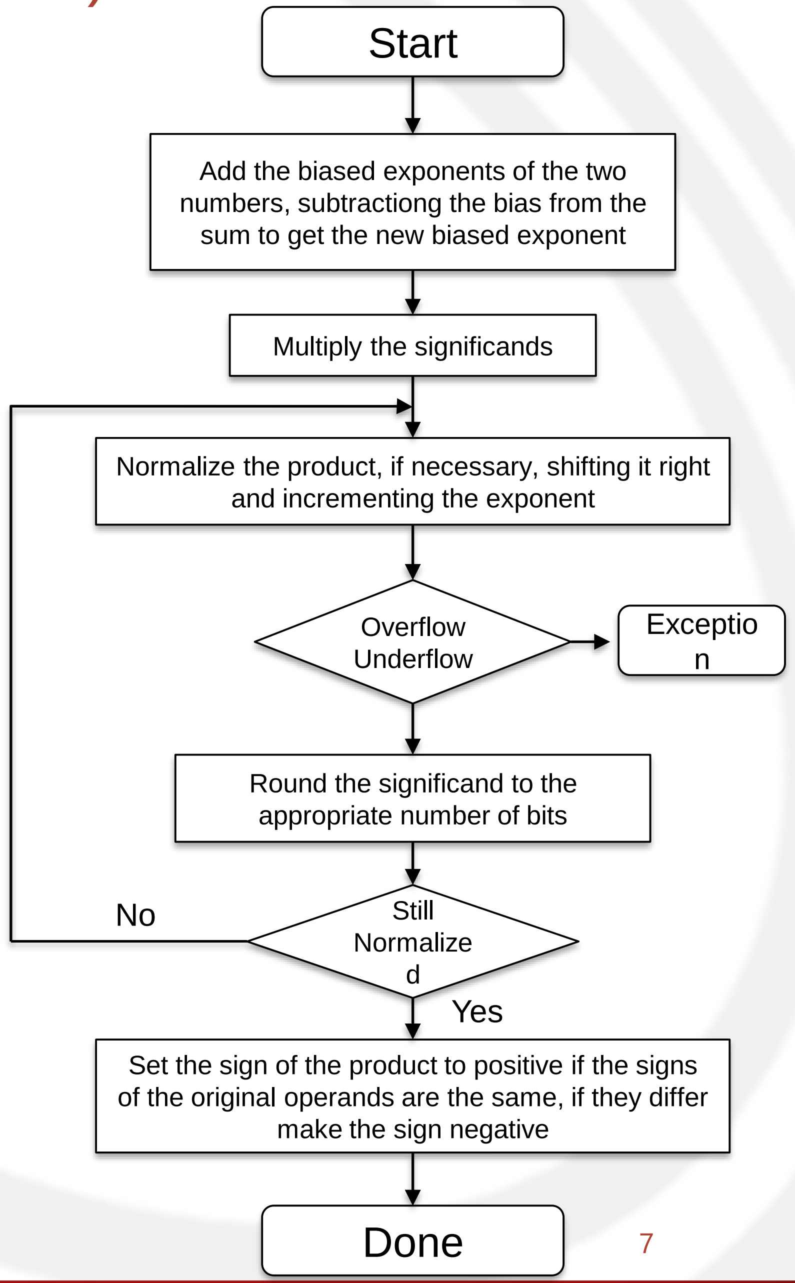
$$(-1)^{sign} * (1 + \sum_{i=1}^{23} \left(b_{23-i} * 2^{-i}\right)) * 2^{e-127}$$

- Little-endian representation

https://binaryconvert.com/convert_float.html?decimal=050046053

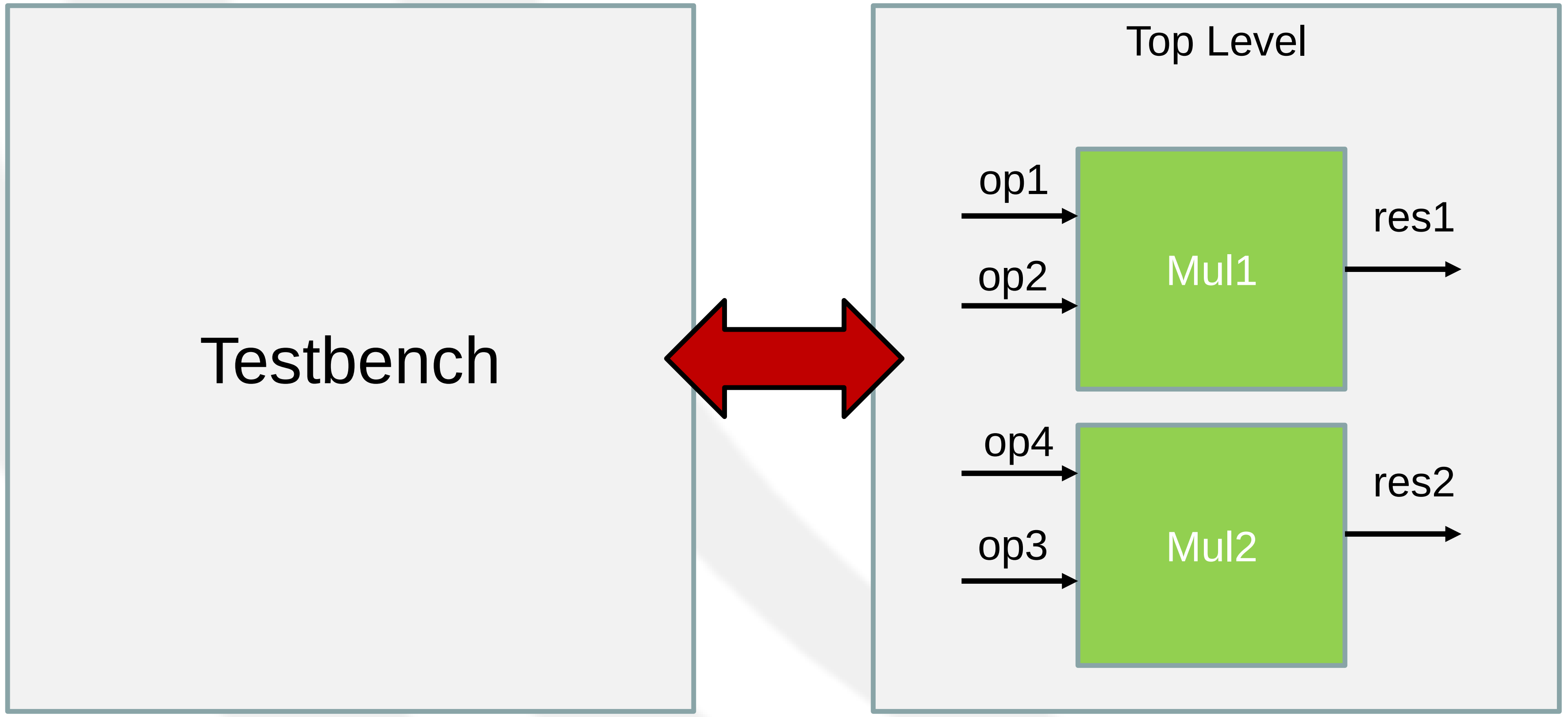
Floating Point Multiplication (Cont.d)

- Implement the function using the depicted algorithm
  - A lot of documentation and information available on the web!
    - IEEE 754 multiplication
    - Floating point multiplication
- Implementation with handshaking protocol
  - Ready flag
    - Inputs are ready, start computation
  - Done flag
    - Outputs are ready, read the results
- Test the implementation with a testbench executing a reasonable number of multiplication (~ 20)
  - Inputs can also be generate randomly
  - The testbench compares VHDL and Verilog results


Start
Add the biased exponents of the two numbers, subtractiong the bias from the sum to get the new biased exponent
Multiply the significands
Normalize the product, if necessary, shifting it right and incrementing the exponent
Overflow Underflow
Exceptio n
Round the significand to the appropriate number of bits
No
Still Normalize d
Yes
Set the sign of the product to positive if the signs of the original operands are the same, if they differ make the sign negative
Done

RTL Modeling

- Design the multiplier in two different version:
  - VHDL/Verilog
  - SystemC

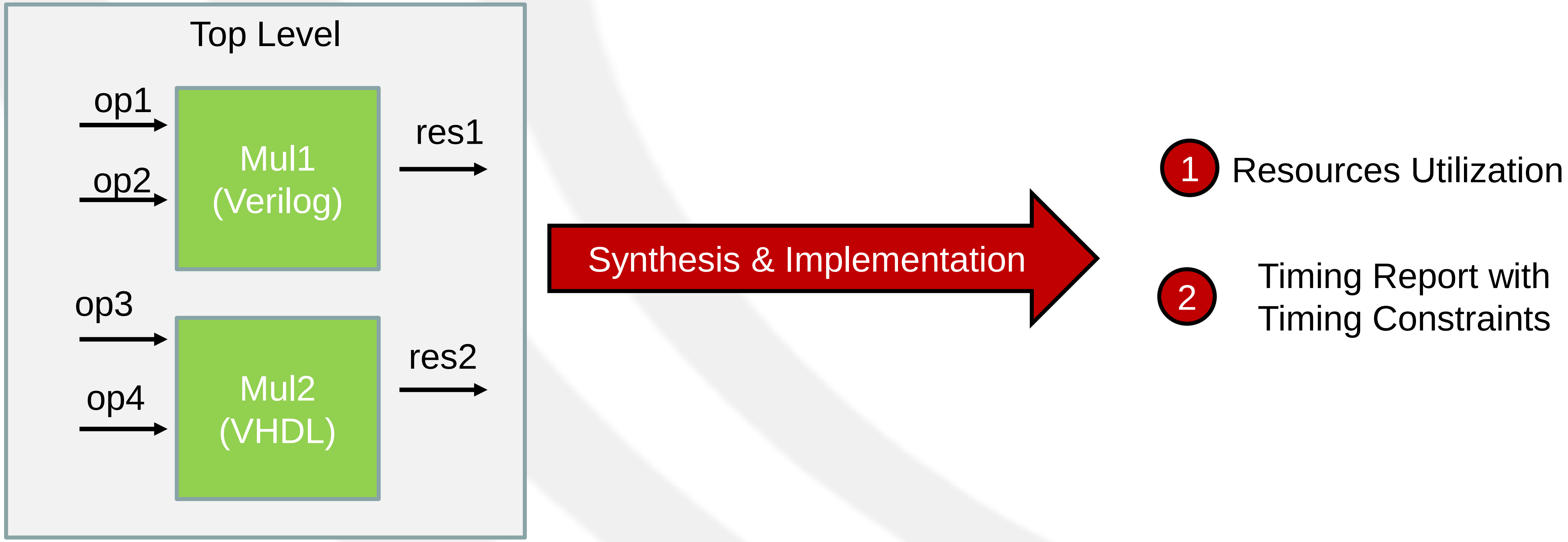


Embedded Systems Design - Assignment

1.2 – RTL Synthesis

RTL Synthesis

- Synthesis of the Floating point multiplication (Verilog and VHDL)
  - Synthesis of the entire design (two multiplier and top level)
- Find the maximum clock speed of your design
  - Add timing constraint before implementation



Embedded Systems Design - Assignment

1.3 – High-Level Synthesis

HLS Synthesis

- Perform High-level Synthesis of Floating point multiplication
- You can choose between:
  - Simple C float multiplications
  - SystemC
    - Use synthesizable SystemC if you use SystemC for HLS
  - C-algorithm of your design
    - C-Algorithm with SystemC datatypes (variable are allowed here)
    - C-Algorithm with C datatypes
      - Primitive C types are recommended: uintX_t (X=8,16,32,64) with bitwise operation (&, <<,>>)

Final Report

- Implement the assigned design at RTL
  - You must use only synthesizable VHDL/Verilog
- What the report should contains for this lesson:
  - Interface definition
  - FSMD and EFSM of the device behavior
    - For the FSMD schema, it is enough to show the connections between Controller and Datapath (not the internal implementation schema)
    - The EFSM in the report has to be consistent with the implementation and the FSMD schema!
      - Remember: define the EFSM and FSMD before implementing the design!
  - Organization of the processes
    - 2 or 3 processes, divide controller from datapath
    - Signals for internal communication
    - Describe process dependencies and sensitivity lists
    - Describe which coding style you used to model the processes
    - Report waveforms of a single multiplier and the top level
  - Describe the main features of the design
    - How do you organize and use the internal signals/wire/reg?
    - What about the latency? Fixed or variable?
  - Justify ALL your choices:
    if something is not on the specification, then write it on the report!

Final Report

- Synthesis Results
  - What are the results after the Synthesis and Implementation?
    - Report resource utilization
    - Perform post-Functional synthesis simulation to verify the correctness of your design
  - What is the maximum speed of the design (clock period/frequency)?
    - Report WNS,TNS, WHS,THS
  - What is the time required to compute an operation in the worst case (max latency)?
- Report and comment the obtained results

- HLS Results
  - Compare the results with Synthesis results of the RTL design
  - Latency and Resource Utilization

Use the template on the moodle platform!

Final Structure of the Assignment

- VR123456_Stefano_Centomo_01/
  - Report/
    - VR123456_Stefano_Centomo_01.pdf
  - Solutions/
    - RTL/
      - VHDL_verilog/
      - SystemC/
        - » bin, include, Makefile, obj, src
      - cpp/ (if you decide to use cpp for HLS)
        - » bin, include, Makefile, obj, src

- **Everything within VR123456_Stefano_Centomo_01.tar.bz2**

Source Code delivery

- For SystemC assignment create the following directories:
  - *src/* containing the source code files
  - *include/* containing the header files
  - *obj/* used to store the object files generated by the compilation
  - *bin/* to store the executable file

- If any parameter is required by the executable
  - Create a README file
  - Handle the parse line for the parameters and print the usage

- Makefile
  - **Use a Environment Variables called *SYSTEMC_HOME***

- ***IF IT DOES NOT COMPILE ON MY COMPUTER, THE ASSIGNMENT WILL BE DISCARDED AUTOMATICALLY!***
  - ***Same configuration of the laboratory Computers! Check your implementation on it!***

Assignments not properly delivered will not be corrected, and the report <u>will not be considered</u>!

Consegne che non rispettano le regole descritte sopra <u>non verranno corrette</u> e il report verrà considerato <u>non consegnato</u>!