Main Objectives/Steps/Tasks:

- SystemC code → DeSCAM → [properties – wrapper – C++ operations] → Vivado → VHDL operations {NORMAL APPROACH}
- SystemC code → DeSCAM → [properties – C++ operations] → edited the C++ operations → Bambu → VHDL operations → Wrote the Wrapper {Bambu APPROACH}
- The main objective was to make DeSCAM able to work with Bambu to offer flexibility in the use of DeSCAM

Main Challenges:

- Writing the wrapper
- Editing the C++ operations
- Separating the C++ code
- Had problem with variable data sizes previously
- The main problem with bambu was that it needs return port to output the port
- In the wrapper I used a register
- Avoided touching the bambu generated code
- Wrote separate properties for each C++ code at first after entering bambu
- Had the design as combinational first with bambu that's why we used register to make it sequential like the PPA
- Had an issue with wait state 2 which Lucas solved in the wrapper, the true false values and the value for z_sig

Stage 1:

- I started off with the Bus_new example, and used it to generate the C++ operations using Bambu and Vivado
- There was a huge difference in file sizes between Bambu and Vivado [10,000 - 400] lines
- The Bambu generated code was much more complicated
- I tried using the .tcl script files [needs vivado to run], .xml directives [needs Vivado] and .sdc extensions [needs FPGA board], to make the files size from Bambu smaller, but it didn't work

Stage 2:

- Changed to work on the Adding example which was simpler,
- I wrote a simple SystemC code to perform addition
- One of the main issues in the generated design was that the number of bits for port signals was variable
- Onespin had issues with the variable sized variables, it didn't accept them

- The wrapper from DeSCAM and the operations VHDL file from Bambu were not fitting together
- Had to make them fit together to make the elaboration work

Stage 3:

- Edited the SystemC file to make it simpler and generated the C++ operations again
- Having an initial value in a C++ file before using it in Bambu is recommended to avoid undefined values
- I edited the operations file .cpp before using it in Bambu, which resulted in a very different Bambu file, especially after removing the ap_int data type, a much lesser code,
- Bambu has a problem with ap_int data type
- I realized that the generated design is wrong in the first place, Bambu design, x notify does not lead to anywhere, that was because the output ports was made input in the Bambu VHDL file
- " In C the semantic is always either "IN" (for parameters passed to functions) or "OUT" (for return values of functions)."
- I can’t return multiple functions variables

Stage 4:

- I wrote 3 separate C files for each output, then I had to write the wrapper so that they are connected together
- generated 3 vhdl files for each state in the .cpp file
- I had to write properties for the separate generated bamboo files from the 3 C++ files first to help me understand how to write the wrapper
- Successfully wrote the wrapper file and the properties to prove them
- most properties conform with DeSCAM properties except the wait state of z_2_2

Stage 5:

- Added registers to change the design to sequential like the PPA
- Solved the problem with wait state 2

Code Changes:

- C++ operations
  - Separate the C++ operations code to 3 different codes, each code returns one of the used variables as an O/P
  - Removed the intermediate/temporary values in the C++ code
- VHDL Wrapper
  - Instantiation of the 3 separate components [3 VHDL files from bambu]

- Connecting the signals and the ports
- Initialized register values in case of a reset
- Added an additional if statement to solve the wait state problem