Project 2

Multiplier – Wakerly's and Paplinski

1. Simulation of Datapath and Control Units in WSM.

The top-level structure for Paplinski multiplier can be shown with the data path and the control path. Control path is responsible to decide the change of states and hence comprises of the counter unit. The counter produces an output 'ZI' which is used to synchronize the data path. If 'ZI' is high, data path is active, and data is driven. Data path comprises of D-register, A-register, Q-register and the ALU unit. D and Q registers are used to store the multiplicand and multiplier input data and ALU unit does + or – operations required for the multiplication. 'Op' is a control data which controls which registers need to be active for the multiplication operation to complete.

2. Comparison of the 4 multipliers.

A simple multiplication operation can be performed in the circuit with different approaches. We have considered a top-to-down approach, where each bit is considered for addition operation in each stage and the final output is the product of the two inputs (As multiplication is same as repeated addition). Further, we can discuss 4 different models that can be used to describe a multiplier in VHDL.

- Firstly, we have considered a behavioral model where we have first described a component for addition, and this model is instantiated multiple times in loop. It follows a top to down approach and each stage addition is carried to the next stage and so on to get the right multiplication output. As the model is behavioral, the synthesis runs as described, however, the synthesis circuit might not be the same as the description. Also, we can see that there is no delay in the output.
- Secondly, we have considered a structural model where the same methodology as described in the first model is used, but we have described each stage with the basic logic gates. Here, we are constraining the synthesis tool to design the circuit to be exactly as the model described. Though the circuit is big and uses more components, we can still see that there are no considerable delays in the output.
- Next, we have considered another behavioral model which is considered the best way to program uses, the pre-defined library unit 'std_logic_arith' which has all the basic arithmetic operations described. Here, using this library, we can just use Y = A*B; which performs both signed or unsigned multiplication based on the ports described. The multiplication operation inside the library unit is described based on the shift-and-add

algorithm. Even in this case, the structure is almost similar to the previous two models and there are no considerable delays in the simulation.

- Lastly, we have defined our model based on the 'Word Serial Multiplier' where we have many small multipliers which are operated in several cycles to produce the multiplication output. Here, we are using just three registers to store the data bits and an ALU unit to perform +/- operation. The same circuit is used to run multiple times to produce the product (multiple addition operation). A counter is used to keep the count of number of times the operation repeats (i.e., multiplication cycles) and a flag is set to restart the operation. This is used to synchronize the data inputs. Here, though the circuit is simple, there is a huge time delay to produce the required multiplication output.

3. Analysis and Conclusion

We can see from all the different descriptions for the multiplier that for any circuit can be described in many ways based on the requirement and the hardware available. In the initial 3 cases, the gate implementation is more, thereby using more hardware. However, there was no delay in the output and the computation was faster. In case of a WSM, though we utilized minimum hardware, as then operation was performed multiple times, the result was not obtained instantaneously but took several clock cycles to generate the result.

As we have FPGAs which consist of millions of registers and logic gates, there is no need to compromise on the hardware utilization and we can try to reduce the delay in the output for fast computation. However, in case of ASIICs, where we have hardware component constraints, we will need to compromise on the computation time and use the latter design description for multiplication.