Answers ICAR Laboratory : PicoBlaze Lab 5

Task 1

- Q: If each instruction takes two cycles to execute the time to process this data is given by:

  Processing time = Number of Instructions × 2 × Clock period

  If the typical clock speed for the PicoBlaze processor is 50MHz how long will it take to execute this program?

- A: IC = 3 + 11×16 + 1 = 180
  Processing time = 180 × 2 × 20ns = 0.0000072 = 7.2us

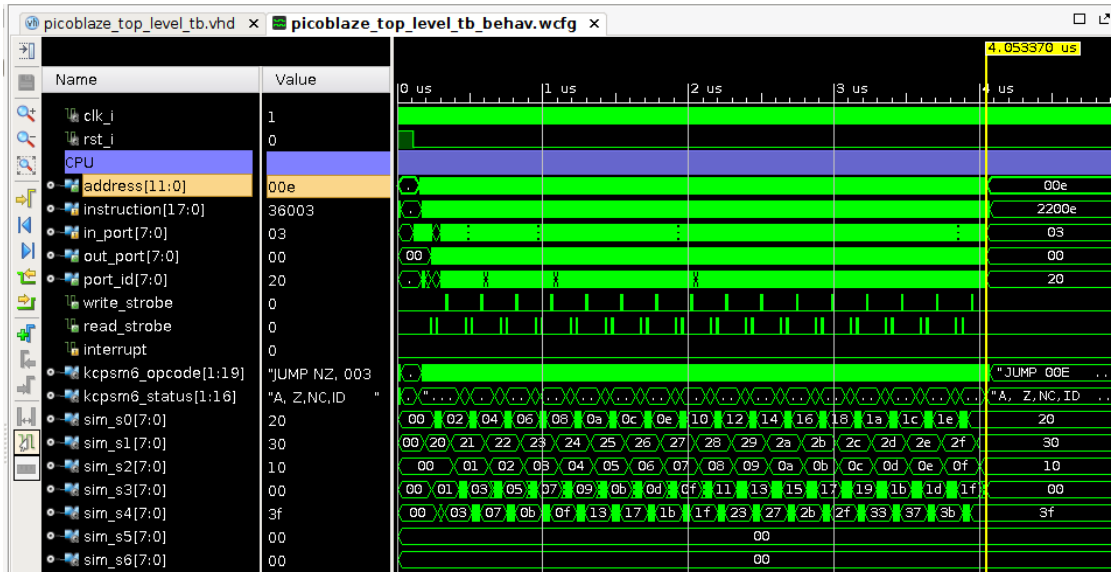
  Note, the processor's clock speed will be different for each system, determined by the hardware architecture i.e. the critical path delay, and the technologies used to implement it i.e. the switching speed of the transistors used.

Task 2

- `PicoBlaze_VHDL_SISD.zip` device utilization summary (baseline used later). Note, the number of slices and clock speed may vary a little owing to changes in Vivado's optimisation algorithms i.e. small variations in how the VHDL model is mapper onto the FPGA hardware.
  - Number of Slices: 141
  - Number of FF : 98
  - Maximum Frequency: 91MHz

Task 3

- Q: How long did it take to execute this program?
- A: 4.05 us

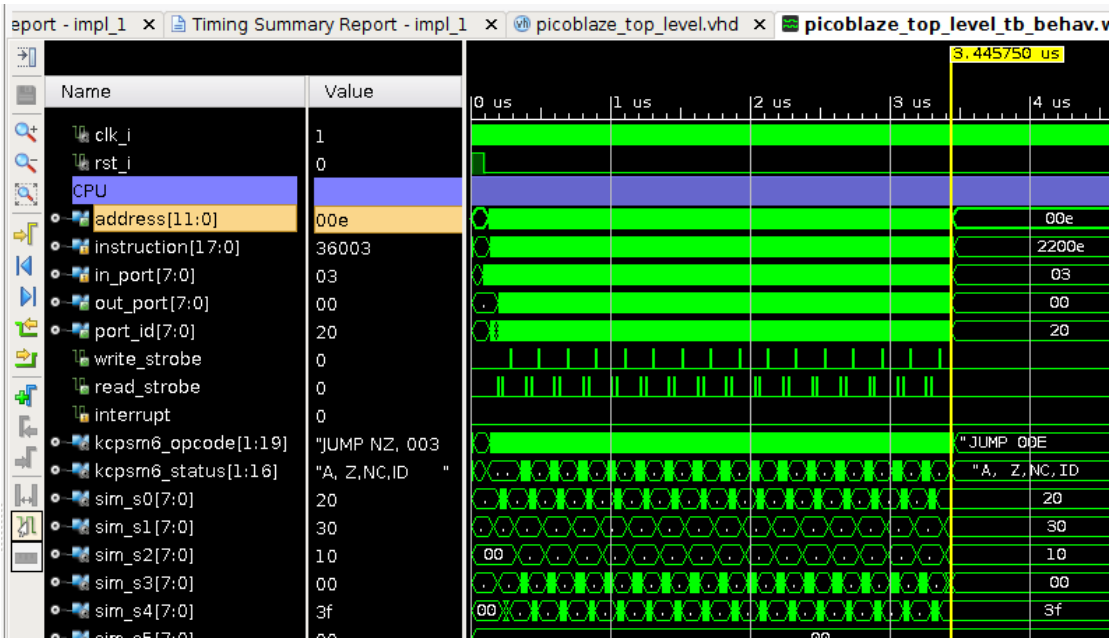


Task 4

- `PicoBlaze_VHDL_MISD.zip` device utilization summary (baseline used later)
  - Number of Slices: 142
  - Number of FF : 114
  - Minimum period: 107MHz

  Owing to how the hardware is implemented in the FPGA i.e. the fundamental building blocks (slices) contain spare 'functionality' therefore, only 1 'additional' hardware block is required to implement the pipeline architecture. However, you can see that 16 extra flip-flops were required.

- Q: How long did it take to execute this program?
- A: 3.4us. Same program, faster clock speed, allowing more instructions to be executed per second.



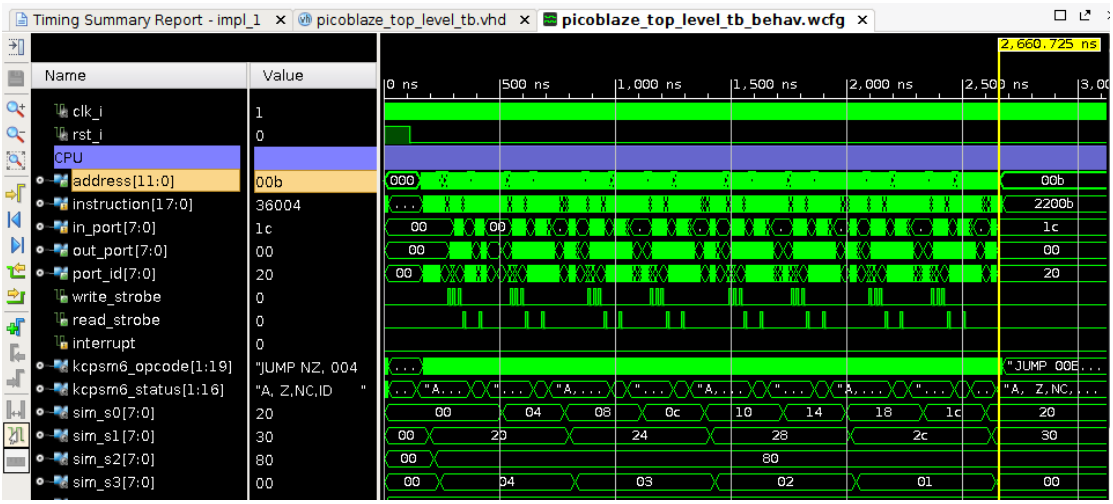
Task 5

- `PicoBlaze_VHDL_SIMD.zip` device utilization summary (baseline used later)
  - Number of Slices: 288
  - Number of FF : 162
  - Minimum period: 88 MHz

Extra hardware required to implement the co-processor, this has increased the critical path delay, reducing the maximum clocking speed i.e. complex hardware means signals may have to travel further. However, as we can now access 4 data value on each memory access and perform two additions at the same time, overall processing performance is increase (even if the max clock speed is reduced).

- Q: How long did it take to execute this program?

- A: 2.6us. From the waveform diagram below, it takes approximately 0.27us to process each pair of data valve, however, the addition only requires a few nanoseconds. The main bottleneck is in transferring data and polling to see if the co-processor is finished i.e. time it takes for the processor to talk to the co-processor. If this functionality is moved into the processor then it would be an order of magnitude faster.



Task 7

- Q: Using the simulator and taking into account these additional coding overheads estimate how long it will take to process this data.
- A: If it is assumed the clock speed period is approx 10ns (100MHz), the processing performance could be expected to be approximately twice as fast as a single processor i.e. two processing cores
  - IC = 6 + 15 × 16 + 1 = 247
  - Processing time = (247 × 2 × 10ns) / 2 = = 2.5us

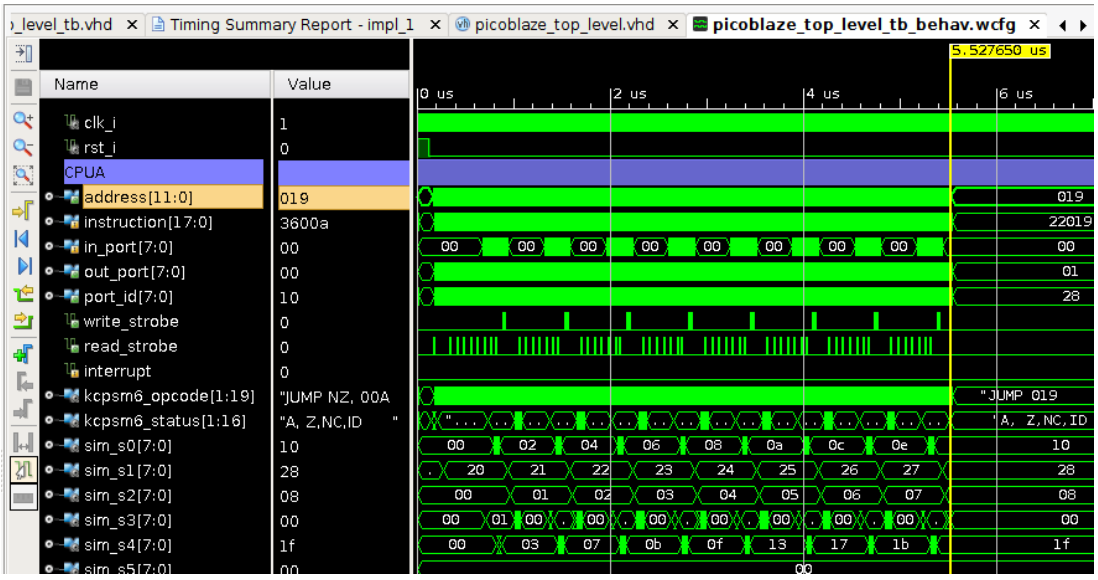
Task 8

- `PicoBlaze_VHDL_parallel.zip`, device utilization summary (baseline used later)
  - Number of Slices: 327
  - Number of FF : 216
  - Minimum period: 84 MHz

  Not unexpectedly need more hardware (slices) as we now have two processors, and due to increased critical path the maximum clock speed is reduced a little.

- Q: How long did it take to execute this program?
- A: 5.5us. This is slower than a single non-pipelined processor. The reason for this is that the two processors use the same memory i.e. they prevent each other from operating in parallel. Therefore, due to the way the code accesses memory and delays associated with arbitration, the program will run more slowly. This problem could be removed if we used dual port RAM i.e.

memory with two separate sets of address, control and data buses, allowing simultaneous / parallel memory transactions.



Task 9

- Q: Calculate the relative speedup of each architecture compared to a non-pipelined architecture.
- A: MISD = 4.05 / 3.45 = 1.17 increased performance
  SIMD = 4.05 / 2.66 = 1.52 increased performance
  MIMD = 4.05 / 5.53 = 0.73 reduced performance

- Q: Compare the relative hardware requirements of each architecture, which gave the best hardware / performance return for the selected application.
- A: Even if the MIMD architecture could double the processing performance, best case results would be in the order of 2.25ns. MISD processing performance is 3.5ns with no increase in hardware. SIMD processing performance is 2.7ns, but needs twice as much hardware. Which is best? No real answer, very much dependent on what processing performance you need and the behaviour of the code i.e. instructions used and memory access required.
- Q: Consider what types of applications would be best suited for the pipelined and parallel architectures. What are the possible advantages and disadvantages of these architectures?
- A: Pipelined: large basic blocks i.e. lots of sequential instructions allowing the processing pipeline to remain full, increasing performance. Programmes that require a lot of branches, IO, Interrupts, memory accesses could reduce processing performance as the processing pipeline may have to stall until data / instructions are available (we will look at this in later labs). Parallel: true parallel architecture, supports parallel software architectures i.e. multi-processing not multi-tasking. Allows independent programs (processes) to run in parallel, however, significant time may be wasted synchronising access to shared resources and shared data.