Report Computer Architecture 2021

Names: Bosmans Jeroen (R0665199), Cattoir Benoit (R0671037), Geenen Tibo (R0667002)

Implementation	*Total Cell Area (mm^2)*	*Area without data path (mm^2)**	*Critical Path (ns)*	*Maximum Operating Frequency (MHz)*	*Number of Cycles for program MULT*	*Minimal time to execute the program MULT*
Single Cycle	*242,896.1983*	*13,555.787*	*30.26*	*33.047*	*1179*	*35,676.54*
Single Cycle with Multiplication Support	*246,550.7889*	*17,203.7699*	*39.78*	*25.138*	*37*	*1,471.86*
Pipelined	*248,753.7092*	*19,414.5686*	*23.58*	*42.409*	*37*	*872.46*
Pipelined with hazard and stall logic	*248,789.7976*	*19,444.0493*	*23.54*	*42.481*	*25*	*588.5*

** Since the instruction memory and data memory are practically the same for every implementation, these areas have been subtracted from the total cell area to create a better overview.*

Questions:

- ***For the single cycle processor, which kind of instruction would stimulate the critical path found? How would you improve it without adding any pipe stage?***

  The load operation will stimulate the critical path. The reason is because it because it requires most components, from instruction memory, to register, ALU (to calculate the address), the data memory and back to the register to store the data. Mostly the memory will add a large time delay, therefore the loading time can be improved by having more / faster cache. However, this does not improve the critical path as a cache miss will still result in even longer time. In order to improve the critical path, the memories can be made smaller, but this might not improve the average performance because of the lower hit rate. Increasing the register or ALU speed will be beneficial, as they are always used.

- ***For the single cycle processor, which resources constitute most part of the gates? What is your explanation for this distribution? Is it possible to reduce gate cells number?***

  The register, because it contains plenty of flipflops, which are built out of gates. To reduce the number of gates, it would be smart to look at this register as it contains most gates. One option could be to look into a different type if register technology, if a good alternative exists. Or decreasing the register size, but this may severely influence the performance.

- ***What are the advantages of using a single cycle processor compared with more advanced implementations? Can you imagine/propose an application scenario of such cores?***

  A single cycle processor has a smaller surface area and simple design, because of the lack of registers between stages. Therefore, it is also cheaper. This can be useful for simple applications, such as a simple micro processor (dishwasher, coffee machine, rain sensor…). They need less performance and require a very low cost. For simple tasks they could even be more efficient, as a single cycle processor has less components and thus requires less energy over time (ideal for smoke detectors on batteries that have to last a long time and don't have to perform many operations, which would require many steps in a pipeline).

- ***Is the critical path affected when hardware support for multiplication is added to the single cycle processor? What is your explanation for this? Do you know any multiplier implementation that can improve timing?***

  Yes, the critical time increases. The reason is that the ALU becomes slower (since extra hardware is needed for multiplication support) and therefore also operations that use the ALU. As the load operation uses the ALU to calculate the addresses, the critical path is influenced. Implementing a separate adder to compute load/store addresses could improve the critical path, the multiplier could then potentially be pipelined.

- ***Is adding hardware support for multiplication a good choice for every microprocessor? Motivate exhaustively your answer.***

  No, you should only include a multiplier if multiplications are a significant part of the total of ALU operations. Otherwise, you needlessly slow down the other operations only to speed up a few multiplications. Also, costs increase when adding a multiplier. This is only worth it if the time reduction is useful. In simple appliance such as a coffee machine the critical path is of low importance. Also, the multiplication ALU might use more power since there is more hardware.

- ***How much larger is the pipelined implementation compared to the single cycle processor? What is the main cause for its increase? How is the critical path affected when we pass from a single cycle processor to a pipelined implementation ?***

  The area of the CPU, when not considering the data and instruction memory, increases from 17,203.7699 mm^2 to 19,414.5686 mm^2. This is mainly because of the addition of extra registers. The critical path is greatly reduced from 39.78 ns to 23.58 ns because it now only spans one stage of the processor (between two registers) instead of the whole operation.

- ***Taking into account the critical path found for the pipelined processor, how would be possible to increase the performance of the system? Would your solution significantly speed up the core? Also, what will be the new critical path?***

  Since the new critical path is only over one pipeline stage, and therefore by far faster than the one of the single-cycle processor, the frequency can be increased to only just guarantee the new critical path to be finished in time. Since the critical path is almost divided by two, the execution time can in the best case also be reduced by the same factor two. In reality however, hazards will not allow for this performance to be reached.

- ***What other microarchitecture techniques (besides pipelining) could be implemented in our microprocessor in order to improve performance? Please explain under what conditions/type of workload you will have the maximum/minimum performance.***

  - Multi-threading: The compiler becomes more complicated and extra logic is added. As a result, only if enough instructions can be shifted to increase performance, the overall performance will increase. Otherwise, the overhead will only slow down the processor without producing any time gain.
  - Parallelizing: running multiple operations in parallel, which can take two different forms: functional or data parallelism.
  - Out-of-order execution: running independent instructions while others are waiting for dependencies
  - Branch prediction: predict which branch will most likely be taken and flush if wrong.
  - Register renaming: avoid dependencies that can be solved by using a different register. As a result these instructions can be ran sooner.

  The performance depends on the program and especially on the dependencies in the program.

  - Pipelining: if you need to stall the pipeline a lot because you are waiting for data to arrive, the performance will be less.
  - Multi-threading: only a way to resolve dependencies and to already start running other code if there are too many dependencies.
  - Parallelizing: depends on the type of operations and parallelism
    - Functional: if you perform a lot of different operations on different data, this type really improves performance. You get bad performance if the software is not well adapted. Example include VLIW and superscalar.
  - Data: if there is a lot of data elements that need the same operation (e.g., image processing), performance will be good. However, if most data requires different operations, performance will be bad. Also, software needs to be well adapted. Examples include vectoring and SIMD.
  - Out-of-order execution: if a lot of independent, time intensive instructions are preceded by dependent instructions the gain is high when they are re-ordered.
  - Branch prediction: if the branches are accurately predicted the wait for the branches can be avoided. Wrong predictions will degrade the performance by flushing.

- Register renaming: if a lot of dependencies can be solved by renaming registers, the performance increase is highest.

- ***Is the addition of hardware improvements, like pipelining, correlated with higher power consumption? How can we assess if a specific modification to our processor improve or diminish the energy efficiency of the system?***
  The power consumption will increase firstly because of the higher clock frequency, and secondly because of the additional hardware. However, the pipeline implementation is generally faster and will therefore use less energy for a certain operation if the power is the same. Standard benchmarks can provide more insight, for the different task available.