ECE4810J System-on-Chip (SoC) Design

Fall 2021

Final Project

Logistics:

- This project is a team exercise.
- Please use the discussion board on Piazza for Q&A.
- All reports and code (if available) MUST be submitted to the assignment of Canvas.
- Internet usage is allowed and encouraged. But copying code/materials from the internet is not allowed (you will get zero score and other penalties).
- No late submission is allowed for this project.
- Creativity is highly encouraged and credited for this project.

1. Project Description

In this class, we have learned various design SoC design methodologies (FPGA & ASIC) together with different flavors of design languages (Verilog, HLS, Python, etc.). In this final project, you will have opportunity to try out one or more of these design approaches/languages and experience an actual design process from concept to implementation. This is a team-based project, you will work in the same team (3 students) as the in the lab sessions.

The goal of this project to implement an application with various hardware development approaches. You have the freedom to choose one of the applications from the given pool. But at least one of the following design flows/approaches need to be used. If you implement with two or more flows, extra bonus will be applied towards your final score.

Flow Options:

- FPGA flow with Verilog + board evaluation on Arty Z7
- FPGA flow with Vivado HLS + board evaluation on Arty Z7
- PYNQ flow with Python + board evaluation on Arty Z7
- ASIC flow (with from RTL to GDS) with Verilog + STA

2. Application Pool

Application 1: Matrix Multiplication Accelerator

Matrix multiplication is a traditionally intense mathematical operation for most processors. Despite having applications in computer graphics and high-performance physics simulations, matrix multiplication operations are still relatively slow on general purpose hardware, and require significant resource investment (high memory allocations, plus at least one multiply and add per cell). We set out to design an accelerator to not only speed up the operations, but also allow offloading of the execution to free up general processor time for other tasks. As an added bonus, accelerators, because they don't contain any processing elements that aren't necessary, typically are more efficient at the task at hand, requiring less power to run than a general-purpose implementation.

For the register-based approach, we wanted to create a design that would be able to perform matrix multiplications in as few cycles as possible. To this extent, we realized that each value in the output matrix could be calculated completely independently, since each value in the output matrix only depends on one row in m1 being multiplied with one column of m2. To take advantage of this massive parallelism, we needed to store our matrices in a data structure that allowed us to write many values into the output matrix simultaneously, and we needed to be able to read many different values from our input matrices simultaneously. This was the reason we decided to implement this data structure from registers. The overall design can be seen below in Figure 1:

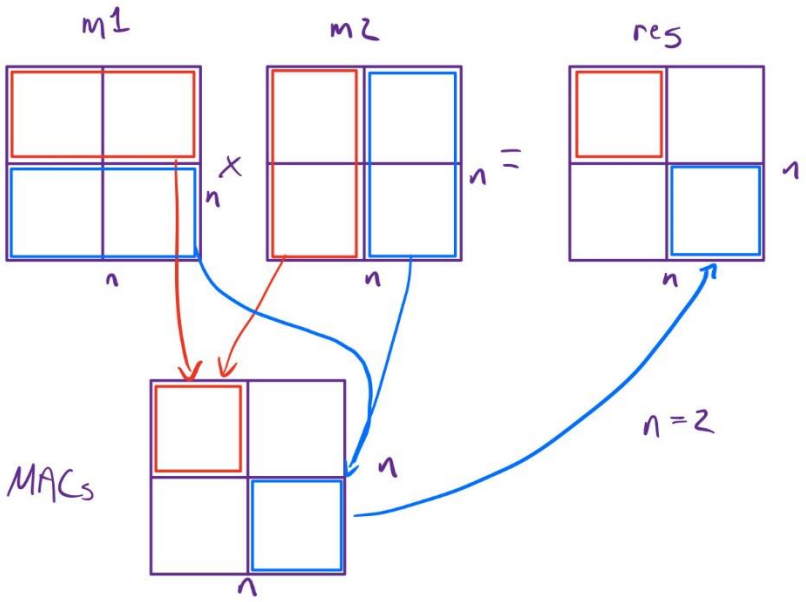


Figure 1: 2x2 Example of Register-Based Design

We hoped to achieve better 'single threaded' performance by increasing the clock frequency and pipelining the multiply-accumulations as much as possible with this scenario.

We compare both of these solutions to a simple implementation of matrix multiplication written in C. This is a straightforward $O(n^3)$ implementation that essentially performs the same operations as the hardware does, which can be seen in Figure 2. This gave us an interesting comparison because while the ARM Hard Processor System (HPS) is running at almost 10x the clock frequency of the hardware we designed, there is far more overhead for any given instruction running in C than in the hardware. For example, in the C code, there are branches for the for loops that must be predicted and resolved, there are cache misses that would require far more cycles to retrieve a value from main memory, and loads and stores cannot happen simultaneously to each of the matrices. So, if our hardware were to keep up with the HPS, it would have to be far more efficient, taking full advantage of each cycle.

```
for (i = 0; i < MATRIX_SIZE; i++) {
  for (j = 0; j < MATRIX_SIZE; j++) {
    accum = 0;
    for (k = 0; k < MATRIX_SIZE; k++) {
      accum += (m1[k][j] * m2[i][k]);
    }
    res[i][j] = accum;
  }
}
```

Figure 2: C Code Implementation of Matrix Multiply

In this work, you are expected to come up with novel ways to accelerate the matrix multiplication with hardware approach.

References:

[1] Dave N, Fleming K, King M, et al. Hardware acceleration of matrix multiplication on a xilinx fpga[C]//2007 5th IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE 2007). IEEE, 2007: 97-100.
http://csg.csail.mit.edu/pubs/bluespec/HardwareAcceleration.pdf

[2] Chen Z. Hardware Accelerator of Matrix Multiplication on FPGAs: Hardware Accelerator of Matrix Multiplication on FPGAs[J]. 2018. http://www.diva-portal.org/smash/get/diva2:1265778/FULLTEXT01.pdf

[3] Jovanović Ž, Milutinović V. FPGA accelerator for floating-point matrix multiplication[J]. IET Computers & Digital Techniques, 2012, 6(4): 249-256.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.301.3662&rep=rep1&type=pdf

Application 2: YOLO: Real-Time Object Detection

YOLO (You Only Look Once) real-time object detection algorithm is one of the most effective object detection algorithms that also encompasses many of the most innovative ideas coming out of the computer vision research community. YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm "only looks once" at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes. With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance.

You can refer to the following projects for more implementation details in terms of architecture and algorithms. But directly copying codes is strictly forbidden.

Example projects:

[1] https://github.com/dhm2013724/yolov2_xilinx_fpga
[2] https://github.com/Yu-Zhewen/Tiny_YOLO_v3_ZYNQ

References

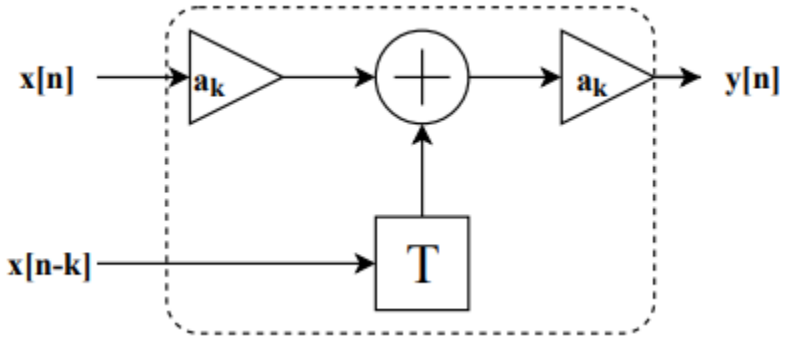
[1] https://pjreddie.com/darknet/yolo/
[2] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. https://arxiv.org/pdf/1506.02640v5.pdf
[3] https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0
[4] Sharma, Aman, Vijander Singh, and Asha Rani. "Implementation of CNN on Zynq based FPGA for Real-time Object Detection." 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2019.

Application 3: FIR Filter

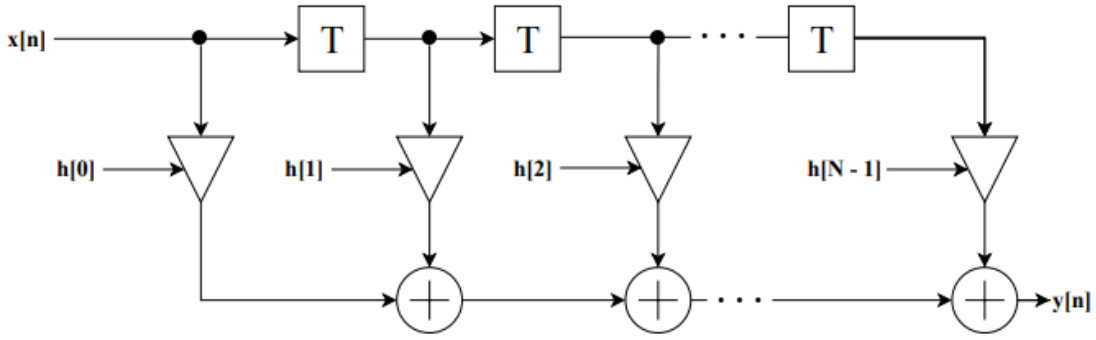
A digital filter is used to modify signal characteristics in the time and/or frequency domain. A digital signal is obtained by sampling the continuous signal in different time frames after which the signal is represented as a sequence of discrete values, unlike the analog signal which is continuous and represented as a function of time. Digital FIR filter algorithms can be described mathematically by the difference equation below or by signal flow graphs that are represented as

block diagrams. The fundamental components used in digital filters, as shown in figure 2.5, are multipliers, adders and delay elements. While difference equations are used to describe the relation between the input and output sequence and are therefore an external behavioral of the filter, signal flow graphs are more detailed and are used to exactly compute the output values y[n] and the internal values in the filter. There are different ways to construct FIR filter.

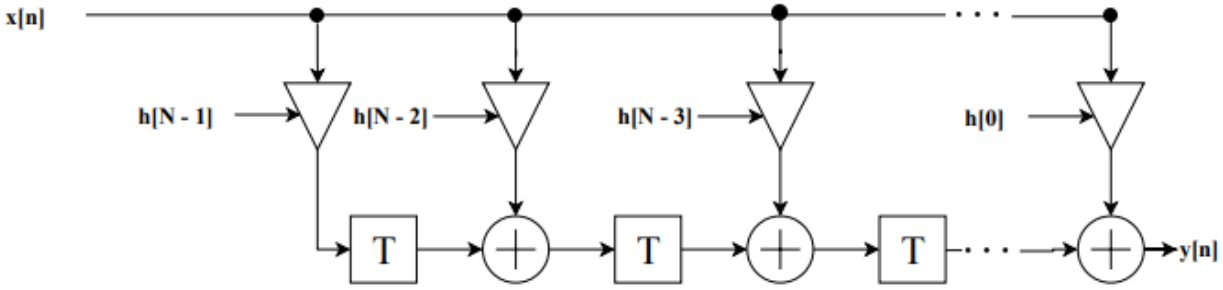
$$y[n] = \sum_{k-1}^{N} a_k \cdot x(n-k)$$



Two common ones are classified as 1) direct form structure and 2) transposed direct form structure. The direct form structure is the classical implementation of the FIR filter equation shown in section 2.2. The input samples are delayed by using delay elements in form of D-flip flops. The direct form structure depends upon so called chain adder where input data is partially computed and outputted through a chain of small adders. The chain adder technique leads to significant reduce in area utilization. The critical path of the FIR filter is the longest path of logic between two registers. Latency is the number of inserted pipelined registers between input and output. The following figure shows the direct form structure of a FIR filter.



The transposed direct form is obtained from the direct form after applying the transposition theorem. Transposition is obtained by replacing inputs with outputs and replacing outputs with inputs where the inputs are directly connected to multipliers. The critical path of transposed structure reduced significantly in comparison with direct form



As an additional feature: You can implement it with approximate computing by introducing approximate multiplier or adders. Bonus will be applied for this feature.

Reference:

[1] https://conferences.computer.org/ictapub/pdfs/ITCA2020-6EIiKprXTS23UiQ2usLpR0/114100a079/114100a079.pdf
[2] https://liu.diva-portal.org/smash/get/diva2:1256720/FULLTEXT01.pdf
[3] Lars Wanhammar and Håkan Johansson. Digital filters using Matlab. Department of Electrical Engineering, Linköping University, 2011.

Application 4: Wishbone Bus

The WISHBONE System-On-Chip (SOC) Interconnection Architecture, also knows as WISHBONE Bus, is a free, open-source standard that defines a common interface among IP Cores in a System-On-Chip. By doing so, it alleviates integration problems. That in turn, encourages IP reuse which leads to improvements in portability and reliability of the system, and results in faster time-to-market for the end users.

WISHBONE is intended as a general purpose portable interface that is independent of the underlying semiconductor technology. As such, its specification defines a set of signals and bus cycles, but does not specify any electrical information nor enforces a bus topology. The WISHBONE bus uses a MASTER/SLAVE model of communication. In this model the MASTER entities are capable of driving the bus, by generating bus cycles, whereas SLAVES entities may only use the bus when instructed/inquired by a MASTER.

Wishbone Interconnection (INTERCON)

The interconnection is the component responsible for tying all the various subcomponents/entities, MASTERS and SLAVES, together in a manner that meets all communication and timing needs. Implementation requirements:

- **Uses a shared bus topology:**

Care must be taken so that just one participant drive the bus at any time.

- **Based on multiplexers.**

One other tricky part of designing a shared bus inside a FPGA is that bus participants which are not driving the bus must neither drive a low, nor a high signal on the bus. Typically, this is achieved by setting the signal drivers to a ”High-Z” state, which means using Tri-State buffers. Since, normally, FPGAs do not have internal Tri-States buffers a switch logic based on multiplexers was implemented.

- **1 to N multiplexing (1 Masters, N Slaves)**

The facts that this is a single MASTER implementation and SLAVES may not use the bus when not addressed, the only care that must be taken it to not overlap the SLAVES memory maps.

- **Partial address decoding**

Each slave present in the bus has a corresponding entry in the memory map. Each of this entries consists of a base address (BASEADDR) and a size (SIZE). The SIZE corresponds to the effective number of addresses occupied by the slave, starting at the BASEADDR. In other words, this SIZE correspond to the maximum OFFSET.

The decoding process is done in two steps: The first step is done by comparators (one for each slave) implemented in this interconnection block. These comparators use the base address to select the slave or not. They do this by masking out the OFFSET bits of the address and then comparing base addresses. The masking out of the OFFSET bits is achieved using the SIZE. The second step is done by a fine decoder inside the selected slave. It will receive just the OFFSET and verify if it is valid.

Using this decoding scheme has the drawback of imposing that the number of addresses used by the slaves must be a power of two, even when not all addresses are used. Hence, memory addresses can be waste.

Reference:

[1] https://arxiv.org/ftp/arxiv/papers/1205/1205.1860.pdf
[2] https://github.com/boschmitt/wishbone
[3] https://opencores.org/howto/wishbone

3. Deliverable

- Group Deliverables:
  - A well-presented project report that has the problem definition, your implementation details, any novelty, your evaluation methodology, detailed simulation results and evaluation results, and future work. No format requirement, but IEEE proceeding format is highly preferred.
  - Package your implementation code, test benches and other necessary files into one zip file. If you use ASIC flow, please also include a copy of the tcl script that you have made changes on.
  - Project demo & presentation – A project presentation session will be held. A 5-min demo session + 15-presentation + 5-min question for each group. Slides need to be uploaded as part of the deliverable. **Each student needs to present their part. All need to participate the presentation.**
- Individual Deliverables
  - Complete the peer-evaluation form appended in the end of this document.

4. Timeline

This is a 4-week project assignment; the intent is to allow you to plan and execute a significant, **open-ended** design exploration and mapping. You will not achieve the implementation goal or the course learning goals by trying to do this in one week. We give you timeline to help provide some structure, but the milestones are minimal and doing the minimum to hit the milestone each week will be insufficient to get you where you need to be at the end. We are giving you flexibility in planning and ordering rather than lock-step specifying exactly what you need to do each week. Also, please note that a group/team project is a great experience for exercising your ability for collaborating with peers. Thus, everyone needs to be contributed to make the project a success. Please watch out the following timeline.

- By **November 17th (Wednesday)**, please indicate your intention for the target application (from the application pool) and planned methodology by filling the following form. **Note that you won't be able to change topics after Nov. 19th unless approved by the instructor. You can adjust your planned methodology though.**
  https://sjtu.feishu.cn/sheets/shtcnBP61XqeTugdwWf6zDHQfPb
- On **December 3rd (Friday)**, during the lab session, please talk to the instructor about your progress and any difficulties.
- On **December 10th (Friday), during the lab session, please present your project.**
- By **December 15th**, please submit your report, final slides along with the code.

5. Grading Policy

Factor	Percentage
Functionality Correctness	40%
Evaluation Methodology & Results	15%
Presentation & Demo	15%
Report	10%
Source Code	20%

- Late submission will result in 0 point for the source code part and deduction of 10% per day in the other part until all 100% is deducted.

Bonus:

- For creative implementation ideas, you will get extra bonus (from 5% to 20%) towards your final grade.
- For those who implement with two or more methodologies, and both have valid results. You will get extra bonus (from 15% to 30%) towards your final grade.
- For excellent implementations, you will get opportunity to publish your results at top conferences/journals. The instructor will help throughout the process.

6. Recommendations

- Pick the application based your interests, availability and your bandwidth. You will need to take the ownership of your topic once you make the selection.
- Dedicated lab sessions will be assigned for helping you finish the project, but you can't rely on only doing it during the lab. Otherwise, it is not feasible to finish it within a short time.
- If you get stuck, please talk immediately as we don't have much time.
- Start as soon as possible, and plan carefully.
- If indeed it is hard to finish all parts, please define a reasonable milestone and talk to the instructor about it. Don't make final decision on your own.
- All team members need to contribute, this is very important!

ECE4810J System-on-Chip (SoC) Design

Fall 2021

Final Project Peer Evaluation Form

Each team member is required to provide a peer evaluation for the team effort of the lab. The score of the peer evaluation should be integers ranging between 0 to 5, inclusively, with 5 indicating the biggest contribution. A score should be given to each team member including yourself according the team member's contribution based on your observation. A brief description of specific contribution of each team member should also be provided. Note this form needs to be finished without discussing with your teammate, if all forms are found the same, then you have to justify how each member makes exactly the same contribution (through demonstration to the instructor).

Name	Level of contributions (0 – 5)	Description of contributions
(yourself)		
Team member 1		
Team member 2		
Team member 3		

Your lab grade is calculated based on the following:
Individual_Average = (sum of all team member's marks) / (size of the student team)
Group_Average = sum of Individual_Average of all team members / size of student team
Individual_Difference = Individual_Average / Group_Average – 1.0
Using the calculated Individual_Difference, we find a factor from the following lookup table.

Individual_Difference	Factor	Individual_Difference	Factor
>=0 and < +10%	1.0	> -10% and < 0	1.0
>= +10% and < +20%	1.1	> -20% and <= -10%	0.9
>= +20% and < +30%	1.2	> -30% and <= -20%	0.8
>= +30%	1.3	<= -30%	0.7

Your final lab grade is calculated by:
Final grade of lab = points for team effort * factor

Note that the final grade of the project will not exceed the maximum points assigned to the project. If the peer-evaluation form is not submitted, then your individual_difference will be based on the available data (without yours) only, and your final score will be:
Final grade of the project = points for team effort * factor * 95%