CMPE295B(LM) Spring 2022 Assignment 1

Project Status Update and Task Planning

Stewart Lach, Chuby Okafor, Khoa Nguyen

1 Background Information

A prime number is a natural number greater than 1 that is not a product of two smaller natural numbers. In other words, a prime number is a natural number that has exactly two distinct natural number divisors: the number 1 and itself. Prime numbers are of central importance to number theory but also have many applications to other areas in mathematics, and other fields including public-key cryptography, computing (in checksums, hash tables, and pseudorandom number generators), quantum mechanics, and evolutionary biology. The sieve of Eratosthenes is an algorithm in mathematics for finding all prime numbers up to any given limit.
The sieve of Eratosthenes can be expressed in pseudocode, as follows:

```
algorithm Sieve of Eratosthenes is:
    input: an integer n > 1.
    output: all prime numbers from 2 through n.

    let A be an array of Boolean values, indexed by integers from 2 to n, initially all set to
        true.

    for i = 2, 3, 4, ..., not exceeding (square root of n), do
        if A[i] is true
            for j = i^2, i^2+i, i^2+2i, i^2+3i, ..., not exceeding n, do
                A[j] := false

    return all i such that A[i] is true.
```

2 Description of Topic and Objectives

The topic that we have selected for the master project is the hardware acceleration of the Sieve of Eratosthenes using a main processor and a set of processing elements. Four different hardware acceleration computational methods will be explored and analyzed through actual implementation:

1. The Sieve of Eratosthenes in its original and un-optimized form.

2. The Sieve of Eratosthenes using an optimization method that is based on the wheel factorization using the basis 2, which is the list consisting of the first prime number.

3. The Sieve of Eratosthenes using an optimization method that is based on the wheel factorization using the basis 2, 3, which is the list consisting of the first two prime numbers.

4. The Sieve of Eratosthenes using an optimization method that is based on the wheel factorization using the basis 2, 3, 5, which is the list consisting of the first three prime numbers.

The performance, defined as the time it takes to carry out the entire computation, will be measured, analyzed, and then compared for each of the above four hardware acceleration computational methods, and the software computational method without any hardware acceleration (with software running on the same main processor).

3 Overview of Design and Implementation

We will implement the above-mentioned hardware acceleration computational methods using the Digilent Zybo Z7-20 Development Board and the Xilinx Vivado Design Suite (version 2017.4). The diagram below shows the overview of the system architecture.

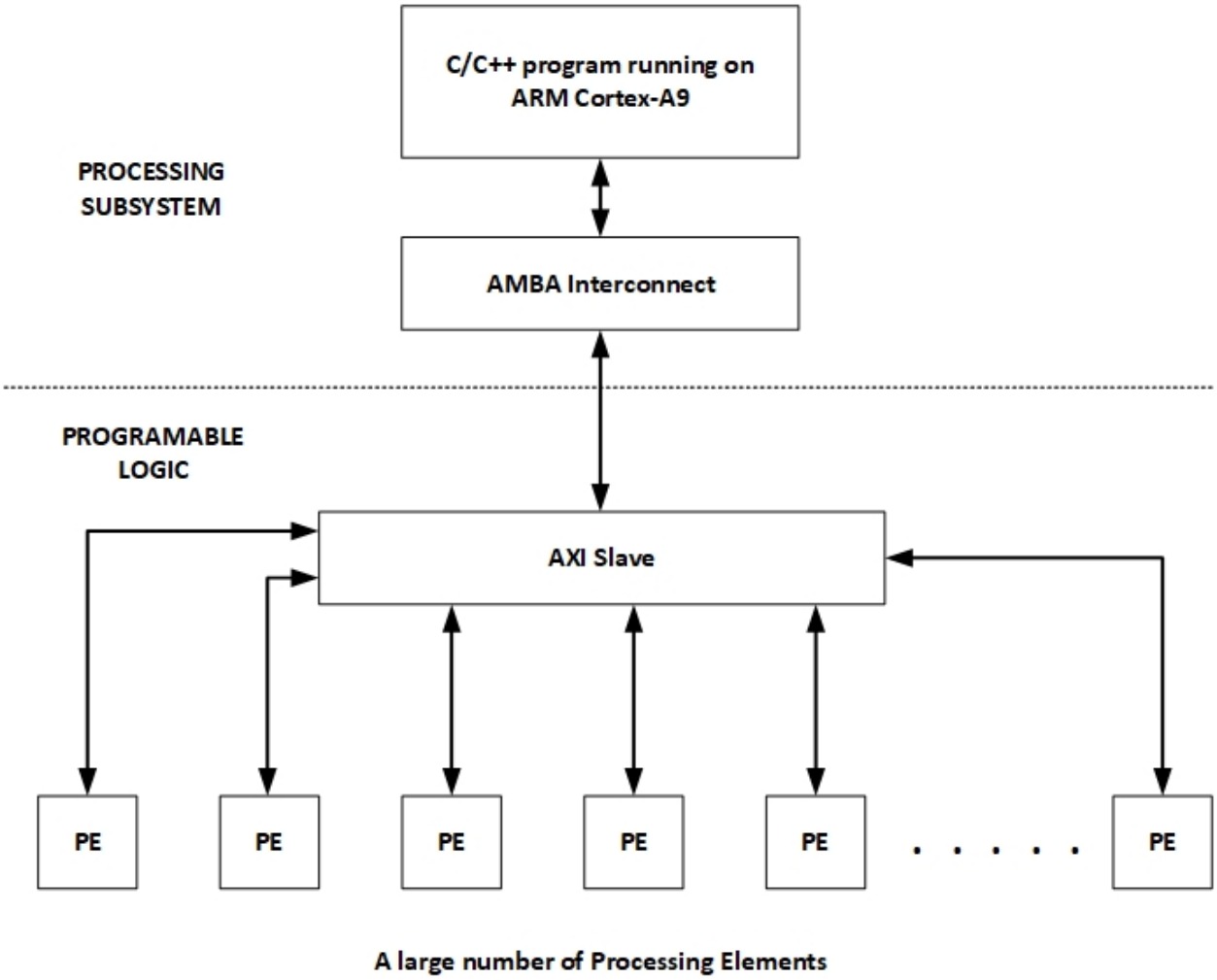


Figure 1: Overview of system architecture

The main processor is the ARM Cortex-A9 on the processing system (PS) of the Xilinx Zynq-7000 platform FPGA device. The set of processing elements (the PE's) resides on the programmable logic (PL) of the device. The PE's are instantiated from the same RTL design module. Each PE instantiates a Block RAM (Xilinx IP) memory instance of size 1024 x 32 bits, for the use of storing the Boolean values as described in the sieve of Eratosthenes algorithm.

The ARM processor communicates with the PE's through the AMBA interconnect on the PS side using an AXI slave module on the PL side. The PE's are memory-mapped to the ARM memory space using this AXI slave module. Using the PE's memory-mapped registers, the ARM processor can send commands to each PE and can query the status of each PE. The diagram below shows the overview of the processing element architecture.

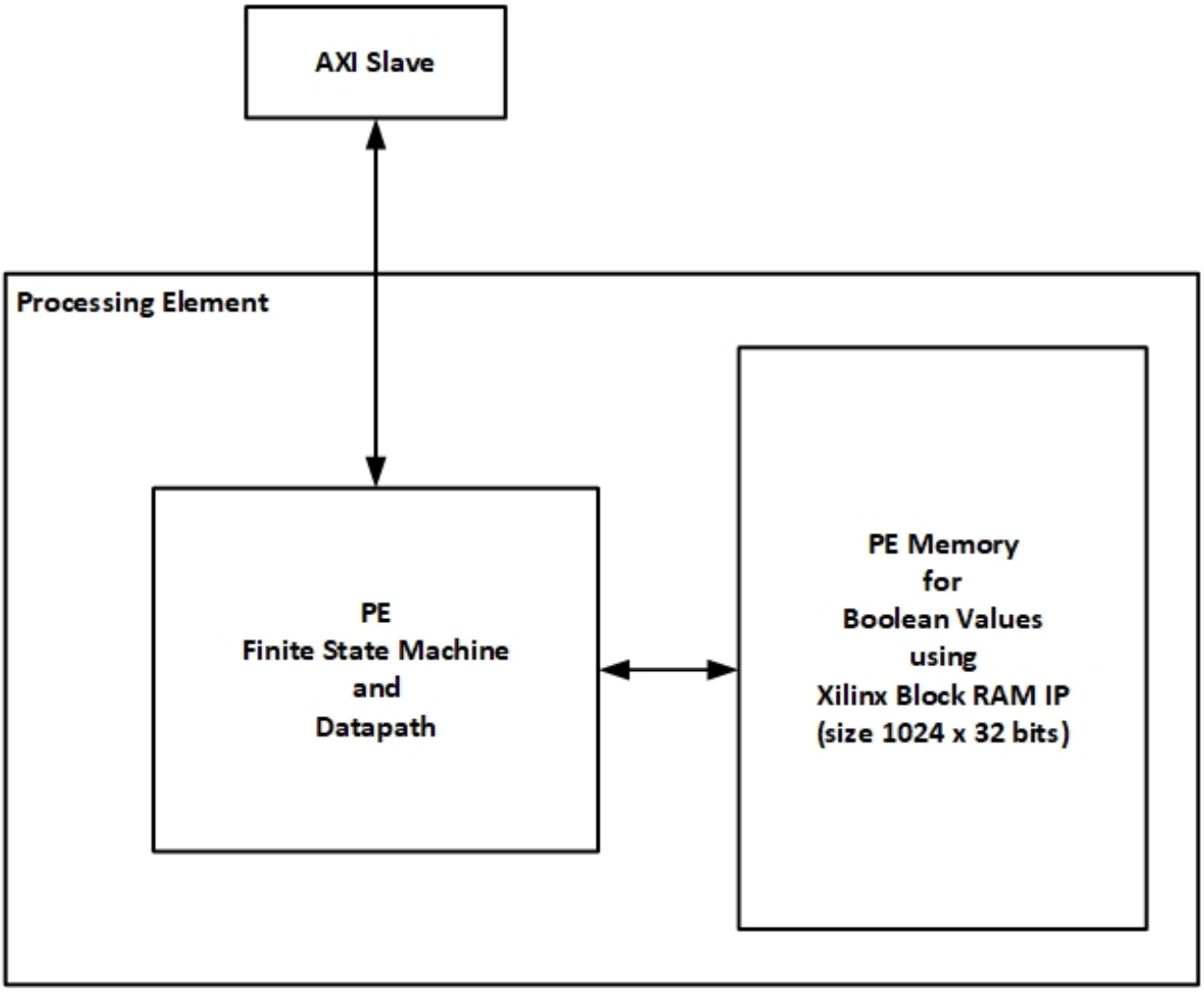


Figure 2: Overview of the processing element architecture

The commands that the ARM processor can send to each of the PE are as follows:

1. **CLEAR MEMORY:** this command causes the PE to clear its entire memory to 0. This command has no input argument. This command is sent once to each PE at the

start of the entire computation.

2. **COMPUTE:** this command causes the PE to set the appropriate Boolean values in its memory as dictated by the input arguments. The input arguments are the `STARTING OFFSET (15 bits)`, the `INCREMENTING STEP (12 bits)`, and the `MAX OFFSET (15 bits)`. This command is used extensively throughout the course of the entire computation from start to finish, which consists of many computational iterations. In each computational iteration, the ARM processor sends different COMPUTE commands to the different PE's. The COMPUTE commands are executed concurrently by the PE's. It is the concurrent execution of the PE's that is the key to the overall reduction of the time it takes to carry out the required computing task.

3. **READ MEMORY:** the ARM processor uses this command to retrieve the results of the computation. At the end of the entire computation, the ARM processor sends a series of READ MEMORY commands to the PE's. The READ MEMORY command has one argument, which is the `READ OFFSET (10 bits)`.

4. **WRITE MEMORY:** this command will be not available in the final design. This command is only available initially during design bring-up and debug. Using this command, the ARM processor can write data to the PE's memory. This command has two arguments, which are the `WRITE OFFSET (10 bits)`, and the `WRITE DATA (32 bits)`.

4 Status

We completed the system architecture design. We also completed the architecture and design the PE. We have started PE's core-level verification.

5 Upcoming Tasks

1. Complete the PE's core-level verification.

2. Integration of the PE's, and form the complete system.

3. System-level verification.

4. System implementation (synthesis and bit-stream generation) and system-level validation.

5. Performance measurement, analysis, and comparison.

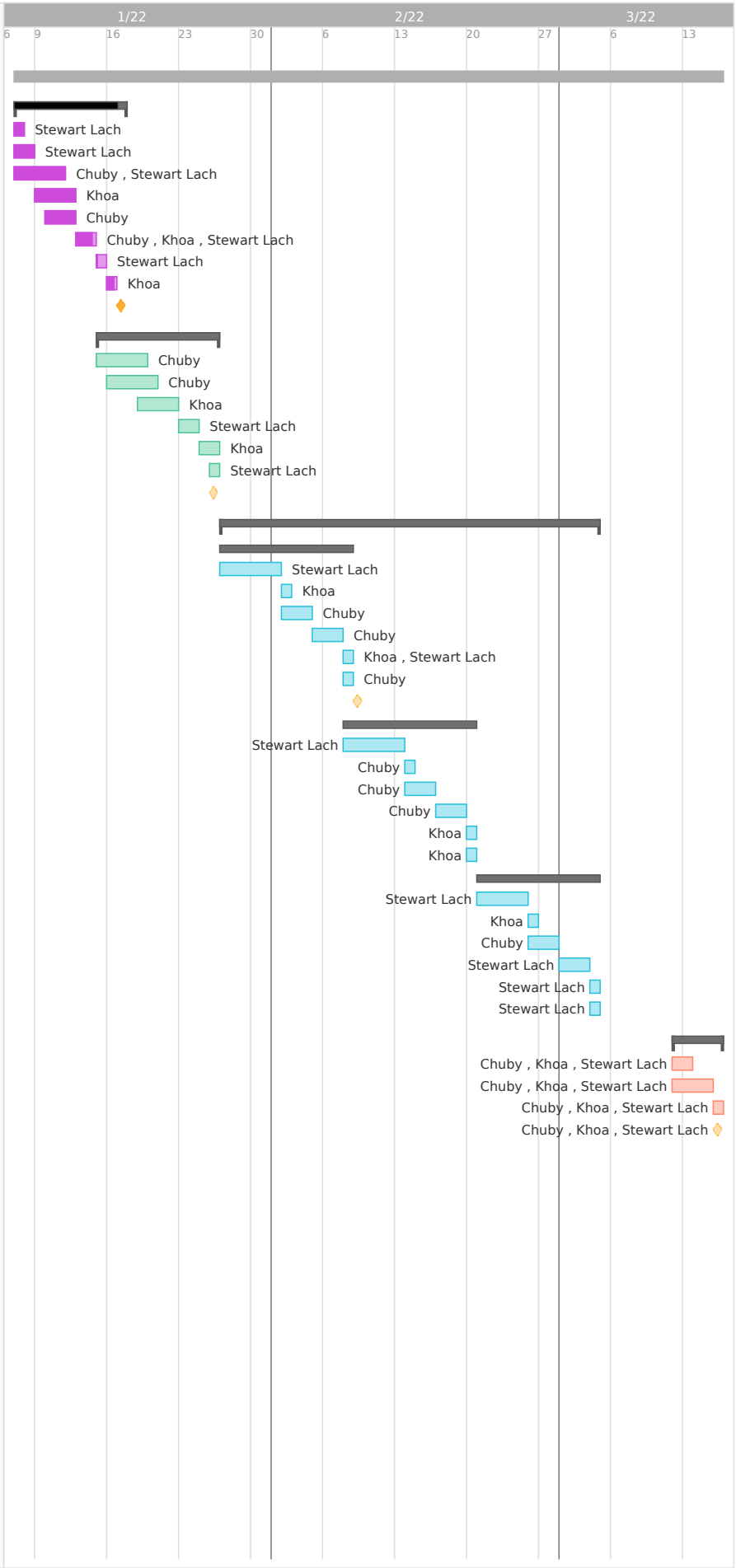
6. Preparation of the final presentation and the final report.

6 Issues Encountered

- The physical distance between the AXI slave module and a PE may be large, so must make sure to properly register all core-level input and output signals, to ensure timing closure during design implementation phase. This activity, registering of the signals, must be analyzed and carried out carefully to balance with the need for speedy signal propagation throughout the design.
- Since there is a large number of PE's, the integration of the PE's needs to be carefully automated to ensure completion in a timely manner, and at the same time ensure the correctness of a huge number of interconnections between the AXI slave module and a large number of PE's. This will also ensure system-level verification will proceed more smoothly.
- The large number of PE's will also significantly increase the rate of utilization of resources on the FPGA, and will put great pressure on the physical design implementation phase. Synthesis runtime will likely be quite longer. Careful FPGA resource management, and careful design timing considerations (through the use of proper core-level signal registering) during design phase are therefore of utmost importance. Following good design practices during design phase will help to reduce the number of synthesis iterations, and synthesis runtime, in the physical design implementation phase.

7 Timetable

See attached gantt chart.



Sieve of Eratosthenes using...	**0h**	**20%**
Single PE proof of concept	**0h**	**92%**
Investigate & research sieve implem...	0	100%
ARM segmented sieve algorithm code	0	100%
Implement ARM code for PE read and...	0	100%
Processing element hardware accele...	0	100%
Develop test bench to verify PE	0	100%
Integration of PE and ARM	0	80%
Single PE validation for small N	0	10%
Document and measure single PE pe...	0	70%
Assignment 1: Project Status Update...	0	100%
Multiple PE on RAM constraint	**0h**	**0%**
Investigate using fixed vs dynamic n...	0	0%
Investigate Zybo RAM constraints	0	0%
Implementation of multiple PEs	0	0%
Integrate multiple PE and ARM	0	0%
Validation of parallel PE for large N v...	0	0%
Document and measure parallel PE ...	0	0%
Assignment 2: Project Progress Upda...	0	0%
Optimization of PE using wheel fac...	**0h**	**0%**
Optimized seive algorithm using...	**0h**	**0%**
Modify PE to support wheel factori...	0	0%
Rerun test bench to verify optimiz...	0	0%
Investigate impacts to Zybo constr...	0	0%
Integrate optimized parallel PE and...	0	0%
Validation of optimized parallel PE ...	0	0%
Document and measure optimized...	0	0%
Assignment 3: Project Progress Upda...	0	0%
Optimized seive algorithm using...	**0h**	**0%**
Modify PE to support wheel factori...	0	0%
Rerun test bench to verify optimiz...	0	0%
Investigate impacts to Zybo constr...	0	0%
Integrate optimized parallel PE and...	0	0%
Validation of optimized parallel PE ...	0	0%
Document and measure optimized...	0	0%
Optimized seive algorithm using...	**0h**	**0%**
Modify PE to support wheel factori...	0	0%
Rerun test bench to verify optimiz...	0	0%
Investigate impacts to Zybo constr...	0	0%
Integrate optimized parallel PE and...	0	0%
Validation of optimized parallel PE ...	0	0%
Document and measure optimized...	0	0%
Final delivery	**0h**	**0%**
Create final report	0	0%
Create final presentation	0	0%
Project presentation	0	0%
Assignment 4: Project Final Report	0	0%