Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Prof. Archana Chidanandan

# Term Project Requirements

## Learning Objectives

In the process of completing the term project, students will develop their abilities to:

- Apply the principle of abstraction in analysis and design problems.
- Specify, design, test, and document instruction set architectures and their hardware implementations, taking into consideration key computer organization design principles.
- Work effectively as members of a team.

## Completion Requirement

Because of the central role that the term project plays in satisfying the learning objectives of the course, **students must complete the term project to the satisfaction of the instructor in order to pass the course**.

## General Description

As a team, you will design a "miniscule instruction set" general purpose processor that can execute programs stored in an external memory. You will also model your design, test it, debug it, assess its performance, and possibly implement it on a Field Programmable Gate Array (FPGA) microchip. Throughout the project, you will maintain current documentation of your design, as well as an ongoing record of your design process.

The project is organized into regularly scheduled milestones that will guide you through the design process. At each milestone, you will submit your current design documentation and design process journal. Your final report will be due during final exam week, at which time you will also give a presentation about your project. You will also maintain a website that provides easy access to the current versions of all the required documentation.

This is a big project, and some of the later milestones represent significantly more effort than some of the earlier milestones, so you should try to work ahead when possible. You will also meet periodically with the instructor to answer questions about your project and discuss your progress.

## Teams

As many students as possible will work in teams of four. You are required to use Angel to submit your project documentation. The files associated with each milestone will be turned in to a folder called M<N>, where N is the number of the Milestone. Your webpage for the project will also be posted through Angel.

## Requirements

The requirements for your processor are as listed below:

Your processor must be capable of executing programs stored in an external memory, with which it communicates using a 16-bit address bus and a 16-bit data bus. Further more, it should support:

- Interrupts from two input devices,
- Reading from a 4-bit input port,
- Moving data between general purpose memory and a special purpose 16-bit register (the "display register"), and
- Writing the contents of the display register to a 16-bit output port.

The instruction set

- must be capable of performing general computations. For example, it must support each of the following with a variety of operand types: addition, subtraction, and conditional branching.
- must support parameterized and nested procedures.

Here are some features that are encouraged, but not required:

- Support for recursive procedures.
- Support for interrupts from five input devices.
- Expanding the input port to 8 bits.
- Expanding the output port (and display register) to 32 bits.

This is not an exhaustive list. You are encouraged to incorporate other features too, and not be limited by the above list.

In order to demonstrate that your processor has met the requirements, you must implement a program that computes the relatively prime value for a number using the Euclid algorithm, as well as performing associated input and output. Specifically, the main program must consist of initialization followed by an infinite loop, and the program must respond to interrupts from two input devices. The interrupt mechanism facilitates the communication between the processor and the outside world.

- When an interrupt from the first device occurs,
    - The contents of the display register are shifted left by four bits,

- The least significant four bits of the display register are replaced by the data from the 4-bit input port, and

- The contents of the display register are written to the output port.

- When an interrupt from the second device occurs, the main program using the Euclid's algorithm (written as a procedure), is applied to find a positive integer that is relatively prime to the input number i.e. the positive integer that is currently stored in the display register[1] (if the display register currently contains zero, then it should remain unchanged).

**The Euclid's algorithm**

For two numbers to be relatively prime, their greatest common divisor(gcd) must be one i.e. they must have no common divisors other than one. The Euclid's algorithm is used to determine the gcd of two numbers. The following is a high-level language specification for a program that your processor must be capable of executing. The program determines the relatively prime value to an input number using the Euclid's algorithm.

```
// To determine m that is relatively prime to n.
// The "main" program must perform the following functionality.

        m = 2;
        while( gcd( n, m ) != 1 ) {           // n is the input from the
                                              // outside world
                m = m + 1;
        }
// The following method determines the Greatest Common Divisor of a and b
// using Euclid's algorithm.
public static int gcd( int a, int b ) {

        int temp;

        while( b > 0 ) {
                if( b > a ) {
                        temp = a;
                        a = b;
                        b = temp;
                }
                else {
                        a = a - b;
                }
        }

        return a;
}
```

[1]One application of the processors developed in this term project might be in a device similar to the SecureID cards that the Department of Defense High Performance Computing Program uses to enhance computer security.

*Note: There are many ways to determine a relatively prime value to a given number. This particular specification is chosen to test the performance of your processor.*

The algorithm and an applet for the above are available on the class web-site.

*As you design your processor, keep the following in mind:*

- *Your project will be evaluated primarily on the basis of how well it demonstrates that you have satisfied the learning objectives of the course. The operative word is "demonstrates," and the mechanism for that demonstration is your documentation. Be thorough, clear, and concise.*

- *Your project will be evaluated (to a lesser extent) on the basis of performance and size. Following the design principles presented by the Patterson and Hennessy textbook will help you make good tradeoffs.*

- *Your project will also be evaluated (to an even lesser extent) on the basis of interestingness, as determined by your peers.*

## Documentation

The documentation is the most important part of the project. This is how you communicate your ideas to the rest of world. It is, therefore, important to document your design process and your design. For this project, you will do it through 3 documents and a webpage.

- The first part of the documentation, the ***design document***, describes the design in sufficient detail for an outsider to completely understand it. More specific suggestions along these lines are included in the Milestones section below. This part of the documentation should be continuously ***updated*** to reflect the current status of the design.

- The second part of the documentation records the process by which the current design was developed, in a ***design process journal***. An outsider should be able to read this document and know the design options that have been considered, the design decisions that have been made, and the rationale for those decisions. This part of the documentation should be continuously ***appended*** to reflect the complete history of the design.

- You will also turn in a ***memo*** for each milestone, indicating the ***current status*** of the project.

- A turnin folder will be created for each group on Angel under CSSE 232.

- The turnin folder should contain the following at a minimum:

    - Folders called M<N> which contain the design process journal, design document and memo for the corresponding milestone.

- default.htm file. This file should have easy-to-access links to the three documents that are submitted for each milestone. As you proceed through the project, the webpage should be updated to point to all the milestones and the corresponding documents.

*Notes:*

- *Hand-written, scanned notes are not acceptable.*
- *Neat and legible hand-drawn, scanned diagrams are acceptable.*
- *MS Word has templates for creating Memos etc. You may use them or get an idea on how to format a memo from them.*

## Milestones

The project is organized into milestones that will guide you through a top-down design process. For each milestone, you will specify your processor at the next lower level of abstraction, verify that your new specification satisfies the requirements imposed by the previous level, and state the requirements for the next level of specification. For example, the first part of milestone one essentially requires you to

1. Write an assembly language programmer's manual for your processor, and
2. Use your assembly language to write the program described in the Requirements section above.

By doing so, you are

1. Specifying your processor at the assembly language level of abstraction, and
2. Verifying that your assembly language specification meets the project requirements.

### Milestone 1: Assembly Language and Machine Language Specifications

#### Part 1: Assembly Language Specifications

Your first step is to describe everything that a user needs to know in order to write an assembly language program for your processor. You should then use your assembly language to write the program to determine the relatively prime values, while satisfying the requirements as listed in the Requirements section above. This program serves as a test to ensure that your instruction set is implemented correctly. However, remember that your processor should be designed to handle other similar programs. Therefore, there will probably be some instructions that are not tested by the program. You should write at least one other program that will test the remaining instructions also. This will also demonstrate that your processor is capable of executing more than just the program to determine the relatively prime values.

**Part 2: Machine Language Specifications**

Your next step should be to describe everything that a user needs to know to write an assembler for your processor, i.e. to translate statements from your assembly language to your machine language. Along with this, you should translate the assembly language programs that were written in Part 1 into machine language.[2]

You may have to modify your assembly language specifications, while working on the machine language specifications. Be sure to discuss these changes and their rationale in the design process journal.

Submit the following for this milestone:

1. Design document that includes the following information.

- A description of the registers available to the assembly language programmer and those reserved for any specific purpose.
- An unambiguous English description of the syntax and semantics of each instruction (see pages A-49 through A-81 on the CD accompanying your book).
- An explanation of any register conventions, especially relating to procedure calls (see page A-22 on the CD accompanying your book).
- An unambiguous English description of each machine language instruction format and its semantics (see page 63 of your book).
- The rule for translating each assembly language instruction into machine language.
- Example assembly language programs demonstrating that your instruction set satisfies the requirements as stated in the Requirements section.
- Machine language translations of the programs written for testing.

2. Design process journal which describes your thought processes and rationale behind your choices and decisions.

3. Memo indicating the current status of your processor design.

4. A webpage that links to the three documents listed above. Please ensure that any image files you include, at this time or anytime in the future, are in a format that does not occupy too much space.

[2] At this stage, assume that your program will occupy the first few bytes of memory and that the program and data will be stored in contiguous locations in memory.

## Milestone 2: Register Transfer Language Specification

Now that your assembly language and machine language specifications are complete, you should describe how you plan to implement your instruction set at the register transfer level (RTL). Your first step in this process should be decomposing each instruction (or set of related instructions) into a sequence of sets of RTL statements, in which each set can be performed simultaneously within a single clock cycle, and the delays for the clock cycles are balanced to the extent possible. Then list and describe the components that are required to implement the RTL statements (e.g. "SE is a combinational logic unit that sign-extends its 8-bit input to produce its 16-bit output"). Make sure that each component is only used once on each clock cycle, and that the descriptions of hardware registers indicate whether or not they are updated on every clock cycle.

It is important not to neglect performance and area for this milestone, because your RTL specification directly impacts the clock period, CPI, and area of your implementation.

You have to test the RTL descriptions to ensure that you have not extended the clock cycle unnecessarily at any step and that all operations that can occur simultaneously are being accommodated into one clock cycle. Generate a test plan to test your RTL description and the correct specification of your components.

You may modify your assembly language and machine language specifications for this milestone. In fact, time spent fixing the instruction set architecture for this milestone will save you lots of time later.

For this milestone, you will submit the following:

1. An updated design document, that includes the following:

- A list of input signals, output signals, and control signals for each component, including the number of bits in each signal.

- An unambiguous English description of each component in terms of its input, output, and control signals.

- An RTL description of each instruction or set of related instructions.

- Descriptions of the tests necessary to verify the correct implementation of your RTL description.

- Any changes made to the Assembly language and Machine language specifications.

2. An updated design process journal, including your thought processes and rationale behind your decisions.

3. A memo indicating the current status of your design.

4. A webpage that contains a link to Milestone 1 documents (un-modified from the previous submission) and a link to Milestone 2 that points to the updated design documents, design process journal and the current memo.

You will eventually model, test, and debug your complete processor using the Xilinx ISE 6.1i software suite. It is not required for this milestone, but because the complete process will be time consuming, it is suggested that you start now by modeling, testing, and debugging your individual components in isolation.

## Milestone 3: Datapath Design and Component Specification

### Part 1: Datapath Design

The RTL description above contains all of the information about how the data flows through your processor. Your next step is to use that information to draw a block diagram of your datapath. Obviously, the block diagram should include the components identified for the previous milestone. It should also show the paths through which the data flows. Where a component's input can come from more than one source, the diagram should show a multiplexer to select between the sources. The diagram should also show the control signals for the components, including the multiplexers, as well as the control units that will generate the control signals, and the inputs to those control units. Once the block diagram is complete, you should list all of the control signals and describe them. Finally, you should specify additional tests to verify the correct implementation of your datapath.

Since the datapath will invariably involve a large number of components, inputs, outputs and busses, it is important to have an exhaustive test plan to test the datapath. You will realize that there are three levels of testing. Testing each component of the datapath(unit testing) and then testing the datapath as it is being built(integration testing) and finally system testing i.e. testing the entire datapath.

**Integration testing:** A good approach to integration testing involves isolating pieces of the datapath and testing them individually. Of course, choosing the pieces and deciding how to test them requires some creativity, and almost certainly some "extra" components. A testable prototype should therefore have extra components that are used to test different parts and states of the design. For this milestone, you should also develop an integration plan that would aid in the testing of the datapath.

*Tips for integration testing: These are only suggestions. You are encouraged to devise your own test plan.*

- *Let the control unit be the last module to be integrated into the testable prototype.*
- *Let the memory module and the Instruction Register be the next to last modules to be integrated into the testable prototype.*

You may modify your register transfer language specification, during the datapath design and the control design phase, but not your assembly language or machine language specifications (unless you obtain instructor approval).

## Part 2: Component Specification

At this point, you have a complete list of components that you will need to build your datapath. For this milestone, you will also specify the design of each of your components at the level of detail necessary to include them in your Xilinx model.

You must build all your components using the Xilinx Schematic tool. (The control unit which you will design in the next milestone, may be designed using Verilog or StateCad.). Memory can designed in Verilog, using the Schematic Entry tool or using Coregen IP.

For this part of the milestone, in the design document, you must specify how you choose to implement each component and provide sufficient detail about the implementation, so that the reader of the document can reproduce each component in Xilinx using your descriptions. You may assume that the reader knows how to use Xilinx. You do not have to include any schematics in your design report.

**Unit Testing:** Every component must be tested. Most combinational logic components will be small enough to test exhaustively. Test cases for sequential logic components should ensure that they satisfy the state transition and output behavior required to implement the RTL specifications (including timing considerations). For this milestone, include a test plan to test each component.

For this milestone, submit the following:

1. An updated design document that includes the following

- A complete but uncluttered block diagram of the datapath. Neatly hand drawn datapaths are perfectly acceptable.
- An unambiguous English description of each control signal.
- Descriptions of the tests necessary to verify the correct implementation of your datapath.
- Integration plan for your components in the datapath.
- A narrative explanation of how the components, datapath, and control units are implemented in Xilinx.
- Descriptions of the tests necessary to verify the correct implementation of your components.
- Changes made to the RTL descriptions.

2. An updated design process journal.
3. A memo indicating the current status of your design.

4. The webpage that will have links to previous milestones' documents (un-modified from previous submissions), and with a link to the current milestone and all relevant files as stated above.

## Milestone 4: Control Unit Design and Component Implementation

### Part 1: Control Unit Design

Your next step in the design process is to specify the control of your processor. The details of this step will depend on which parts of the control you decide to implement using combinational logic, a finite state machine (FSM), and microcode. For any combinational units, you should provide a truth table relating the inputs to the outputs – see Figure 5.13 (page 302) in your textbook. For any FSMs, you should provide a state transition diagram (or table) – see Figure 5.38 (page 339). For any microcode units, you should describe the microinstruction format – see Figure 5.7.2 on the CD (page 5.7-5) – and provide a micro-program – see Figure 5.7.3 on the CD (page 5.7-10). However you choose to implement the control units, you should also specify how you plan to test their correct implementations.

*Tip: Add a reset state(in the state diagram) or reset instruction (in the micro-program) to initialize your processor to the appropriate values.*

You may modify your register transfer language specification, during the datapath design and the control design phase, but not your assembly language or machine language specifications (unless you obtain instructor approval).

### Part 2: Component Implementation

In the second part of this milestone, you should complete the independent modeling, testing, and debugging of all your components, except the control unit. Build each component as per your plan in the previous milestone and use your test plan to test the device comprehensively.

As always, remember to consider how your project will be evaluated. Maintaining good documentation remains the most important consideration, but creativity at this stage can still influence performance, area, and especially interestingness.

For this milestone, submit the following:

1. An updated design document that includes the following

- The specifications of the control units, as described above.
- Descriptions of the tests necessary to verify the correct implementation of your control units.
- An updated list of the control signals.
- An electronic version of your current Xilinx models for all the devices(except the control unit). Include all of the test-benches necessary to implement your test plan.

- Changes made to the RTL descriptions.
- Changes to the integration plan.

2. An updated design process journal.
3. A memo indicating the current status of your design.
4. The webpage that will have links to previous milestones' documents (un-modified from previous submissions), and with a link to the current milestone and all relevant files as stated above.

It is suggested that you begin modeling, testing, and debugging your control unit in Xilinx.

## Milestone 5: Component Integration and Datapath Testing

It is very important to have a well-planned approach to testing and verification, especially for the datapath. A systematic integration and testing plan will save you untold hours. At this stage, now that your components are implemented and tested, you will construct the entire datapath, using the testing and integration plans devised in Milestone 3. You must also complete the implementation, testing and debugging of the control unit, for this milestone.

*Tips on testing are repeated here: These are only suggestions. You are encouraged to devise your own test plan.*

- *Let the control unit be the last module to be integrated into the testable prototype.*
- *Let the memory module and the Instruction Register be the next to last modules to be integrated into the testable prototype.*

At this point, your datapath need not be completely working. However, it should be completely assembled and read for final system testing.

**System testing**: You must generate a test plan to test your assembled datapath.

As always, remember to consider how your project will be evaluated. Maintaining good documentation remains the most important consideration, but creativity at this stage can still influence performance, area, and especially interestingness.

You may modify your register transfer language specification during this phase, as well as the datapath and the control specifications, but not your assembly language or machine language specifications (unless you obtain instructor approval).

For this milestone, submit the following:

1. The design document that is updated to include the following:

- An updated integration plan, highlighting any changes from the plan in Milestone 3.

- A system test plan to test your assembled datapath.

- Any modifications to the RTL, datapath or control unit design.

2. The updated design process journal.

3. An electronic version of your current Xilinx model for this milestone. It should be a complete model of your processor, including all of the test-benches necessary to implement your integration test plan.

4. A current memo.

5. The webpage that includes links to all previous milestones' documents (un-modified from previous submissions) and a link to the current milestone and all relevant documents as listed above.

Maintaining good documentation is still the most important consideration in evaluating your project, but keep in mind that the implementations of your components directly impact the performance and area of your design. There are also opportunities to affect the interestingness.

You may modify your register transfer language, datapath, control, or component specifications, but not your assembly language or machine language specifications (unless you obtain instructor approval).

## Final project and documentation

For this milestone, your Xilinx model should be completely working. You may modify your register transfer language, datapath, control, or component specifications, but not your assembly language or machine language specifications (unless you obtain instructor approval). Be sure to update the design documentation, and include the changes and their rationale in the final design report.

Determine the final results for your project:

1. By simulating its execution, determine the number of clock cycles required to determine the relatively prime value to an input value provided by the instructor.

2. By examining the program, determine the instruction count for the section of the program that determines the relatively prime value.

3. From the appropriate Xilinx reports, determine the size of your processor (in gates, excluding the external memory) and the minimum clock period (including the external memory).

**On the due-date, submit the following:**

- Your final report (see below for details).

- An electronic copy of your final Xilinx model **(This has an earlier deadline. See schedule for actual date.)**

- A copy of briefing slides of the presentation.

- Reflections: each team member should review your design process journal and then write a 1-page summary reflection. It should address the following questions:

  - What is the most valuable thing you learned about working in teams?
  - What is the most valuable technical material you learned?
  - How could you have been a more effective member of your team?
  - What was your most significant contribution to the project?
  - (optional) Do you have any suggestions for future projects?

  Each team member must write their own reflection (it is not a team effort). Take this reflection seriously! Comments should be brief, but thoughtful. The reflection forms a significant portion of each individual's project grade.

## Final Documentation

Your final report will be a substantial document. It should include a cover page, a table of contents, an executive summary, an introduction, the body, a conclusion, and appendices. The body should ***summarize and discuss*** your instruction set design, your implementation, your Xilinx model, your testing methodology, and your final results. The appendices should include your complete design documentation, design process journal, and test results. Except for the appendices, the report should be written as if the reader is familiar with computer architecture, but not with this project.

## Presentations

Each team will present their project to the class. Your presentation should include an overview of your design, highlighting unique or interesting features. Each team member must contribute in a substantial way. Presentations will be peer reviewed. Additional details will be announced.