Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Profs. Larry Merkle and 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, including a demonstration. You will also maintain a website that provides easy access to the current versions of your design and design process journal, as well as your final report and your presentation.

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. Meetings normally are scheduled for 25 minutes.

Teams

As many students as possible will work in teams of five. The remainder will work in teams of four. Students may select their own teams provided that they do so no later than the date

indicated on the course schedule.[1] An e-mail message from one of the members to the instructor is sufficient, provided that the other members also receive the message.

You are required to use CVS to submit your project documentation. The files associated with each milestone must be tagged as specified below. Additional details regarding required CVS usage will be discussed in class. You are encouraged to use CVS to manage your work throughout the process, not just when each milestone is due.

Requirements

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. 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. Furthermore, the instruction set must support

- Parameterized and nested procedures,
- 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 following 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.

In order to demonstrate that your processor has met the requirements, you must implement a program that computes relatively prime integers, 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:

[1] A good team is one in which the members have similar goals regarding the outcome of the project and have the ability to work well together. The latter may depend on personalities, schedules, and physical proximity.

- 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, Euclid's algorithm is used inside of a loop to find a positive integer that is relatively prime to the positive integer currently stored in the display register[2] (if the display register currently contains zero, then it should remain unchanged).

A high-level language specification of this program will be provided on the course website. All other design decisions are up to you.

Documentation

The documentation is the most important part of the project. It has two primary parts, which document the design and the design process, respectively.

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 (See details in the Milestones section below).

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

[2] Euclid's algorithm determines the greatest common denominator (GCD) of two integers. Note that the computation of relatively prime integers is one of the steps in the generation of RSA encryption keys. Thus, 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.

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.

At each milestone (except the last), you will submit: [3]

- A memo summarizing the current status of your project, including:
  - A brief description of your activity, successes, and difficulties since the previous milestone.
  - A summary of your design decisions and the rationale for them.
  - A clear statement of the degree to which you believe you have satisfied the progress requirements for the milestone.
- A copy of your current design documentation.
- A copy of your design process journal.

You should submit these documents by committing them to your CVS repository (see below) with the tag `Milestone <N>` before the deadline for the milestone. The remainder of this section describes the individual milestones in detail.

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 (which should also be enough to write a simple compiler). You should then use your assembly language to write the procedures listed in the Requirements section above, along with main programs to invoke them. These programs will eventually serve as useful tests to ensure that your instruction set is implemented correctly, but they may not thoroughly test your instruction set. Thus, you should also specify additional test programs. At this point your design documentation should include:

[3] See the course schedule for the due dates of the milestones.

- A description of the registers available to the assembly language programmer.
- An unambiguous English description of the syntax and semantics of each instruction (see pages A-55 through A-75 of your book).
- An explanation of any register conventions, especially relating to procedure calls (see page A-23 of you book).
- Example assembly language programs demonstrating that your instruction set is capable of implementing the procedures listed in the Requirements section.

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.[4] It is possible that you will need to specify additional test programs (for example, there may be machine language instructions that do not have assembly language equivalents). In addition to the items identified previously for this milestone, your design documentation should also include:

- An unambiguous English description of each machine language instruction format and its semantics (see page 118 of your book).
- The rule for translating each assembly language instruction into machine language.
- Machine language translations of the programs written in Part 1.
- Descriptions of any additional tests that are necessary to verify the correct implementation of your instruction set.

As you design your instruction set and the machine language, 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.

[4] To ensure compatibility with code that the instructor will write for the final milestone, starting addresses and other guidelines for memory usage will be posted on the course website.

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. You probably will still not quite get it completely "right"

Register Transfer Language Specification

Once 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. Finally, specify tests to verify the correct implementation of your components and your RTL description.

You must have at least one test case for each RTL instruction. A test case consists of a description of a series of operations to be performed at each cycle of an instruction's execution and the expected outcomes. The operations must determine the contents of each relevant register and also check the availability and use of the required component. It might help to think in terms of which instructions you should test first, so that they could be used to test other commands and so on.

In addition to the previously identified items, your design documentation should now include:

- 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.

Remember to consider how your project will be evaluated. By now, you should be in the habit of maintaining good documentation. 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 may modify your assembly language and machine language specifications for this milestone. In fact, time spent getting the instruction set architecture for this milestone will save you lots of time later (said another way, cutting corners at this point will cost you lots of time later). Be sure to update the design documentation, and include the changes and their rationale in the design process journal.

After this milestone, the instructor will write an assembly language program for each team's processor that evaluates the correctness and the performance of the required procedures. Consequently, ***after this milestone, you may only modify your assembly language or machine language specifications by approval of the instructor.***

You will eventually model, test, and debug your complete processor using the Xilinx ISE 5.2i 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.

Datapath and Control Design

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.

As stated earlier, here too, there must be at least one test case for each instruction. Tests should ensure that there is a datapath in the design for each instruction. The availability of each component at the required time also has to be tested. Test cases should check for the availability of the inputs to each component. Furthermore, the control signals that control the components and also the inputs to the components need to be verified.

The additions to your design documentation for this milestone should include:

- A complete but uncluttered block diagram of the datapath. Neatly hand drawn datapaths are perfectly acceptable, but Xilinx schematics are not.
- An unambiguous English description of each control signal.
- Descriptions of the tests necessary to verify the correct implementation of your datapath.

It is suggested that you continue to make progress in developing the Xilinx model of your processor by connecting your components as required for your datapath. Leave the control signals as inputs for now, and use them to model, test, and debug your datapath.

Part 2 : Control 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 5.15 in your textbook. For any FSMs, you should provide a state transition diagram (or table) – see Figure 5.42. For any microcode units, you should describe the microinstruction format – see Figure 5.45 – and provide a microprogram – see Figure 5.46. However you choose to implement the control units, you should also specify how you plan to test their correct implementations.

Again, every instruction must be tested. Essentially, test cases should ensure that all cycles of an instruction are represented in the state machine and that the truth tables have an entry for every input/output combination for all combinational units. If you have microcode units, each micro-instruction has to be tested and it has to be verified that all control signals are being activated and in the correct order.

The additions to your design documentation for this milestone should include:

- The specifications of the control units, as described above.
- Descriptions of the tests necessary to verify the correct implementation of your control units.

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 the datapath design and the control design phase, but not your assembly language or machine language specifications (unless you obtain instructor approval). Be sure to update the design documentation, and include any changes to the RTL design and the rationale for these changes in the design process journal.

It is suggested that you model, test, and debug your control units in Xilinx.

Component Specification

The final step in your design process is to specify the design of each of your components at the level of detail necessary to include them in your Xilinx model. Some components (e.g. the ALU) will require specification at the gate level, but others (e.g. the PC) can be specified in terms of higher level units that are pre-defined within Xilinx (e.g. registers).

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, you should make the following additions to your design documentation:

- The specifications of the components, as described above.
- Descriptions of the tests necessary to verify the correct implementation of your components.

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). Be sure to update the design documentation, and include any changes and their rationale in the design process journal.

It is ***strongly*** suggested that you model, test, and debug your components in Xilinx.

Testable prototype

As stated above, you will use the Xilinx ISE 5.2i software suite to model, test, and debug your design. If you have not already done so, you should complete the independent modeling, testing, and debugging of your components, datapath, and control, as suggested in the milestones above. It is very important to have a well-planned approach to testing and verification, especially for the datapath. The "big bang" approach requires very little thought, because it doesn't require you to include any "extra" components. However, it is very unlikely to provide any useful information, so don't use it. Better approaches involve 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. The additions to your design documentation for this milestone should include:

- A narrative explanation of how the components, datapath, and control units are implemented in Xilinx.

- Descriptions of the tests designed to verify the implementation.

In addition to the design documentation and design process journal, you must submit an electronic version of your current Xilinx model for this milestone. It should be a complete model of your processor, including all of the testbenches necessary to implement your test plan. You should have successfully tested each of the components in isolation. Your status memo should report the outcomes of each of the tests, the actions planned in response to any unsuccessful tests, and your plan for integrating the components. A systematic integration and testing plan will save you untold hours.

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). Be sure to update the design documentation, and include the changes and their rationale in the design process journal.

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 design process journal.

Shortly before the milestone is due, the instructor will provide you with a Xilinx model for an external memory containing the final testing program. The program will consist of a main program that invokes the implementations of the required procedures that you submitted as part of your design documentation for the RTL milestone (unless the instructor approved a later change to your instruction set architecture). Integrate the memory component with a working Xilinx model of your processor.

Determine the final results for your project:

- By simulating its execution, determine the number of clock cycles required for the final testing program.
- By examining the program, determine the instruction count for the final testing program.
- 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).
- Use the cycle count, the instruction count, and the minimum clock period to calculate the average CPI and the ideal execution time for the final testing program.

For this milestone, submit the following:

- Your final report (see below for details).
- An electronic copy of your final Xilinx model
- 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.

Website

You should maintain a website in the folder

`/class/csse/csse232/turnin/<section>/<TeamNumber>/project`

All of the files for the website should be stored within that folder or a subfolder thereof.

At a minimum, this website should provide easy access to your current design and design process journal, the snapshots of those documents that you submit at each milestone, your final report, and your presentation. You may include anything appropriate that you would like (e.g. scanned images, HTML, Xilinx files)

Having your project archived in a website will make it easy for both you and your instructor to show off your excellent work.