



How to Write Design Report 1

Mya Poe¹ and Keith Winstein²

¹ MIT Program in Writing and Humanistic Studies

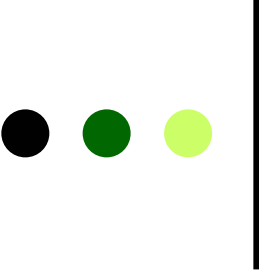
² CSAIL

March 2005



Why are you here today?

1. Proposal → report
2. Understand how a computer designer “thinks through” a design problem.
3. Understand what we look for when grading your DP1.
 - Proposal is not the report!
 - “A” on proposal may not = “A” on report



When thinking about this project at the meta-level . .

- o **Tech Audience:** Engineers building BEDs & writing drivers
- o Purpose: Help the **Implementers**
- o **Persuasive:** Why is your design better? Does it work?

-
- √ Explain why you made decisions
 - √ Acknowledge design trade-offs
 - √ Write for readers who skim (stand-alone visuals)
 - √ Write for readers who do not read sequentially



Steps in the DP1 Writing Process

1. Read comments on your proposal
2. Re-read the assignment
3. Prioritize issues + get peer feedback
4. Write
5. Clarify and refine report -- peer review!
6. Proofread

Step #1

Read comments on your proposal

- What information was missing or unclear?
- What was good?
- Can you build off existing design or do you need to “start from the ground up”?

You wrote:

When a BED wants to send data to the Beta, it sends an interrupt. The Beta asks all of the BEDs if they sent the interrupt. When it finds the right BED, that BED puts 4 bytes on the Memory Read Data lines. Because we can transmit 4 bytes in one cycle, the bus can achieve 40 MB/sec.

TA responded:

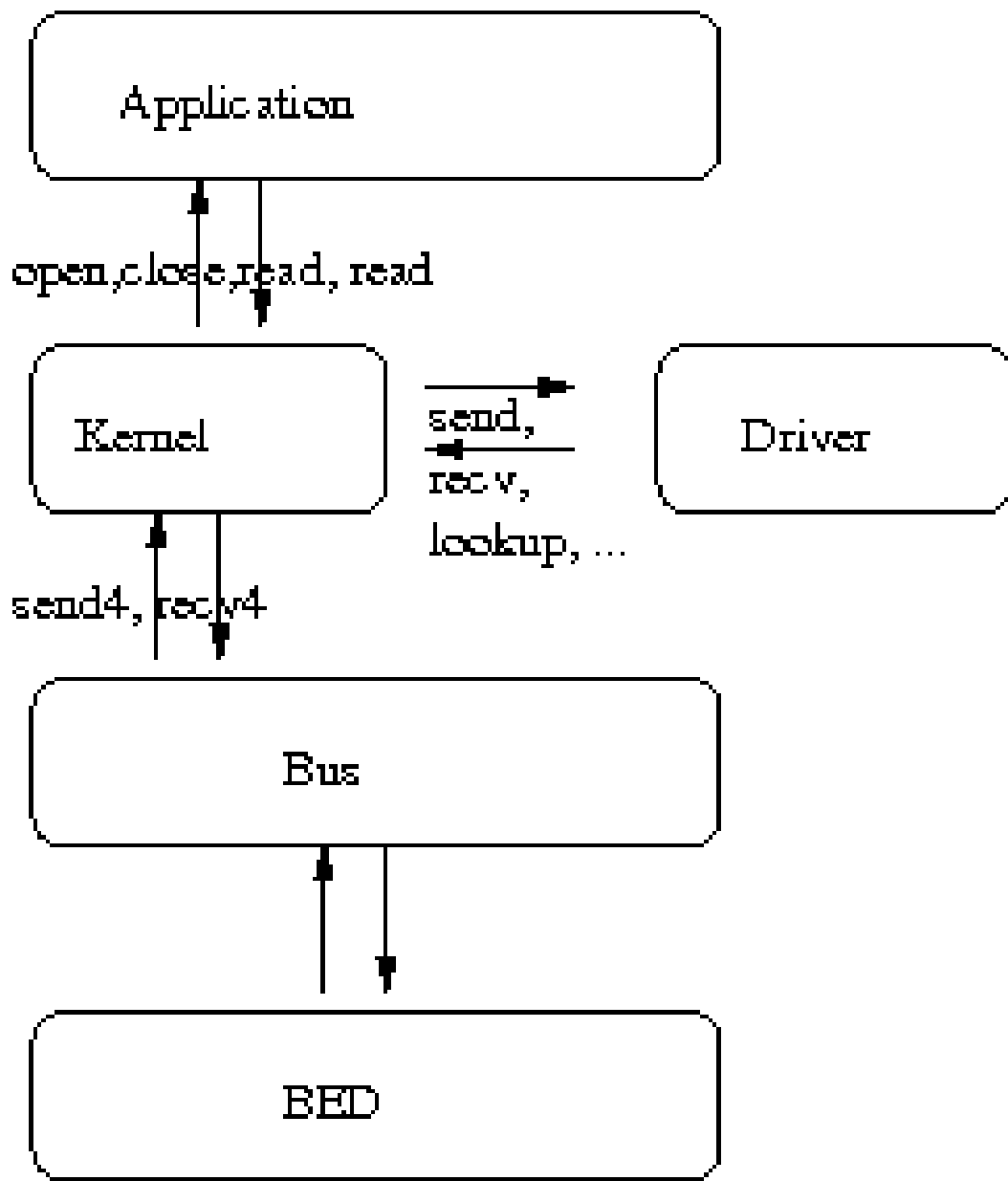
What do you mean by “sends an interrupt,” “asks” and “finds the right BED”? What actually happens? It seems like it takes 32 cycles just to send 4 bytes. How is this 40 MB/s?

Better:

"When a BED wants to send data to the Beta, it raises the IRQ line. In its interrupt handler, the BED first loads the list of connected BED addresses from its data memory into registers. Then, the BED puts the address of each of the connected BEDs on its MA lines in turn. If an addressed BED does not have data to send, it replies with a zero on the Memory Read Data lines. If an addressed BED does have data to send, it replies with the data on the Memory Read Data lines. It could take up to 16 cycles for the Beta to load the list of connected BEDs and another 16 cycles for the Beta to find the right BED. Because it takes up to 32 cycles to send 4 bytes, my bus may be as slow as 1.25 megabytes/sec."

Best:

- (1) What if a BED actually wants to send a zero as its data?
- (2) Design a bus that is capable of sending 30 megabytes/sec of payload data, and explain how it can.
- (3) Be clear about what the kernel is doing and what the device driver has to do for itself.



Step #2

Re-read the assignment to find what information is missing

Proposal did not address all aspects of assignment:

- What's missing?
- What about format? Document specs?
- FAQ: Check daily

Compile list of issues :
comments + assignment + FAQ

Recommendations:

1. Consider your audience: "What would I need if I were them?"
2. Explain exactly which bits of memory persist over reboot and which over unplugging. Most people use rebooting to "clean up" problems that occur. Does your design defeat this?
3. Consider using a **conceptual diagram** (the layers and interface between them) as well as a **physical diagram** (the components and wires between them).
4. Always be specific! What wires go where? What bits are put on them, by whom, in which cycles?
5. Use multiple techniques to explain your design:
 - Specify what each part does under all circumstances.
 - Use narratives to show what all parts do in particular cases (plugging, initializing a new BED, reading data, rebooting).

Pitfalls:

1. Not demonstrating 30 megabytes/sec payload throughput.
2. Not explaining how to use the data memory.
3. Not explaining who assigns "type" identifiers. How do you prevent against collisions? What if two different manufacturers want to use the same driver?
4. Not giving a clear interface between the device driver and the kernel. Recommendation: have the kernel provide just a few functions (e.g., `send_data_on_bus()`) for the device driver to use. Explain what the arguments to these functions are, and what the kernel does when they are called.
5. Not specifying the connector or slot. How many pins does it have? What do the pins do? Which functions are performed by the bus hardware and which by the BED?

Step #3

Identify priorities for your design

1. Does this bus use a fixed number of separate slots (like PCI)? Or is the topology unrestricted (like Ethernet)? Or something in between? How many pins does the connector use, and what are they? How fast is it?
2. How are addresses assigned? What do addresses look like?
3. What happens when a factory-fresh BED is inserted? What happens when an old BED from this machine is reinserted? What about an initialized BED from another machine? What happens when the machine is rebooted?
4. How are `/dev` names mapped to BEDs? If I'm writing a device driver, what function calls or interfaces do I need to use to talk to the device? What function calls or interface do I need to talk to the application?

Step #4a

The Design Introduction overviews your design goals and approach

- State design purpose
- List specific design considerations
- State your approach to the problem

Example Template

1.0 Design Overview

The goal of this design is to provide . . . We accomplish this goal by

Example

The goal of this design is to connect external devices (BEDs) to the Beta. We accomplish this goal by replacing the Beta's data memory with a *Keith-bus*. The bus has 32 physical slots and is responsible for routing data to the appropriate BED. The *Keith-bus* provides 35 megabytes of sustained throughput per second.

Step #4b

The Design Description details your design approach

- Organize by topic:
 - 2.1 Physical Connection
 - 2.2 Identifying BEDs
 - 2.3 Initializing New BEDs
 - 2.4 Software Interface
- Use subsections to show hierarchy of ideas
- Tell readers what & why you made choices
- Weave in discussion of design trade-offs

Example

2.3 Web Server Process Architecture

The web server uses a SPED architecture for its simplicity and performance. Because the system exclusively uses RAM for data-storage, there was no need to worry about kernel support for asynchronous disk I/O or other disk-related drawbacks that SPED may have. On the other hand, a SPED design makes it easy to implement the web server functionality needed for this system.

Courtesy of Vincent Yeung

Develop description from general to specific

2.3 Web Server Process Architecture

The web server uses a SPED architecture for its simplicity and performance. Because the system exclusively uses RAM for data-storage, there was no need to worry about kernel support for asynchronous disk I/O or other disk-related drawbacks that SPED may have. On the other hand, a SPED design makes it easy to implement the web server functionality needed for this system.

Topic sentence
conveys purpose of ¶



What &
why of
design
decisions
explained



Use figures & pseudo-code to illustrate concepts

The `delete_protection_domain` works analogously to `grow_protection_domain`. When a process is deleted from the mote, all the data that followed that process in memory is “compacted”, or shifted up, creating one large block of allocated memory instead of two smaller separated blocks. For example, if a process with seven pages of allocated memory is destroyed, the OS shifts all the other data in memory up by seven pages, as shown in Figure 8. The pseudocode below details the operation of the `delete_protection_domain` procedure.

```
boolean destroy_protection_domain([in]int PID)
{
    int removed_pages <= bound_table[PID] + 1;

    // update the base_table values and the EOMpointer
    EOMpointer <= EOMpointer - removed_pages;

    for (p = 0; p < max_processes; p++)
    {
        if (base_table[p] > base_table[PID])
            base_table[p] = base_table[p] - removed_pages;
    }
}
```

Figure 8: Pseudocode for the `destroy_protection_domain` method. Note that there is no need to explicitly reset the values stored for the base and bound of the destroyed PID. When a new process is given that PID, the old values will be overwritten anyways.

Use figures & pseudo-code to illustrate concepts

The `delete_protection_domain` works analogously to `grow_protection_domain`. When a process is deleted from the mote, all the data that followed that process in memory is “compacted”, or shifted up, creating one large block of allocated memory instead of two smaller separated blocks. For example, if a process with seven pages of allocated memory is destroyed, the OS shifts all the other data in memory up by seven pages, as shown in Figure 8. The pseudocode below details the operation of the `delete_protection_domain` procedure.

```
boolean destroy_protection_domain([in]int PID)
{
    int removed_pages <= bound_table[PID] + 1;

    // update the base_table values and the EOMpointer
    EOMpointer <= EOMpointer - removed_pages;

    for (p = 0; p < max_processes; p++)
    {
        if (base_table[p] > base_table[PID])
            base_table[p] = base_table[p] - removed_pages;
    }
}
```

Figure 8: Pseudocode for the `destroy_protection_domain` method. Note that there is no need to explicitly reset the values stored for the base and bound of the destroyed PID. When a new process is given that PID, the old values will be overwritten anyways.

Step #4c

Write 1 ¶ Conclusion

- Summarize design problems you solved,
- Identify problems in your design, &
- Justify why your design does not address these problems

Example Template

5.0 Conclusion

This design uses [x] to . . . This design does not cope well with [x] because . . .[explain why you did not address this issue]

5.0 Conclusion

The *Keith-bus* design provides a lightweight interconnect among intelligent devices. Its principal weaknesses are its reliance on fixed, physical slots, its limited throughput, and its large connectors. The *Keith-bus* is best used in applications where these weaknesses are acceptable, such as within a personal computer.

Step #4d

Write the front and end matter



- Abstract

- Title Page

Title

Your name

ID#

Recitation instructor

Section time

Date

- Acknowledgements

- Anyone who helped you with design

- References
IEEE style

Abstract

The purpose of this project was to design a virtual memory system that met three basic design goals: enforced memory modularity, the capability of dynamic memory reallocation, and minimized power consumption. After analyzing a number of possible designs, a segmented memory system was chosen as the preferred implementation. By performing reads and writes in only one memory access and pushing the burden of computations to the dynamic allocation procedures (which are invoked very rarely), the design presented in this report met all design goals while minimizing energy consumption.

Abstract

The purpose of this project was to design a virtual memory system that met three basic design goals: enforced memory modularity, the capability of dynamic memory reallocation, and minimized power consumption. After analyzing a number of possible designs, a segmented memory system was chosen as the preferred implementation. By performing reads and writes in only one memory access and pushing the burden of computations to the dynamic allocation procedures (which are invoked very rarely), the design presented in this report met all design goals while minimizing energy consumption.

← Purpose statement

← Design Goals

← His Approach

← Advantage of this Approach

Acknowledgements

Thank you to Professor Kaashoek and Chris Lesniewski-Laas for their suggestions on achieving fault isolation.

References

[1] F. Cavalieri, T. Ruscio, R. Tinoco, S. Benedict, C. Davis, and P. K. Vogt, "Isolation of three new avian sarcoma viruses: ASV9, ASV17, and ASV 25," *Virology*, vol. 143, pp.680-683, 1985.

Step #5

A little extra time dedicated for review will improve your grade

- Give your report to a peer for review
- Double-check the design specs
- Consider from the audience perspective. “I’m designing a keyboard and writing its device driver. What do I need to know?”
- DP1 graded on writing & content:
6.033 is CIM course
B or better =no revision; B- or less = revision

Step #6

Proofreading Checklist

- ❑ Did you chunk information into expected sections?

Abstract

Title Page

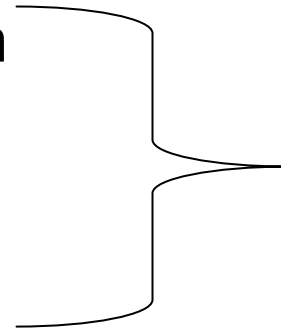
1.0 Design Overview

2.0 Design Description

3.0 Conclusion

Acknowledgements

References



Physical Connection

Identification

Initialization

Application Interface

Step #6

Proofreading Checklist

- Did you # the pages? Your name on every page?
- All figures/tables labeled & referenced in the text?
- All sources cited?
- Did you avoid:
 - naked “this”
 - “the reason is because . . .”
 - “the fact that . . .”
 - over-use of “I”
 - “due to” is an adjective. Try “because”.
 - passive voice
- Did you proofread a printed copy?



Report Format

- 11 or 12 point font
- Single-spaced
- No more than 5,000 words, including executive summary
- Submit 2 copies



Writing Help

- Model DP1 papers on 6.033 website
- Readings in your course packet
- Writing Center <http://web.mit.edu/writing>
- *Mayfield Handbook of Technical and Scientific Writing*

Writing Tutors available: Monday – Thursday am



How do we grade DP1?

Technical staff:

1. Is the design described unambiguously?
2. Is it obvious that the design achieves requirements?
3. Are your design decisions well justified?



How do we grade DP1?

Technical staff:

1. Is the design described unambiguously?
2. Is it obvious that the design achieves requirements?
3. Are your design decisions well justified?

Writing Staff:

1. Is the report well-organized within and across sections?
2. Is the report professionally presented?
3. Are text and figures integrated?
4. Is the writing crafted for readability? Edited prose?
 - 5. Did you use “due to” improperly?