ECE 2031 Project Proposal

Team 101 – Caleb Song, Harsha Tambareni, Maya Rajan, Michael Probst, Sarah Copenhaver

Design Problem

- Create an IO peripheral from SCOMP that will interact with the SRAM chip on the DE2 board.
  - Read/write capabilities
- Balance ease of use and speed to make the most effective peripheral

Design Problem: Inferences

- The memory will likely be used for a storing data from a source like sonar
- Interface must excel in continuous reads and writes

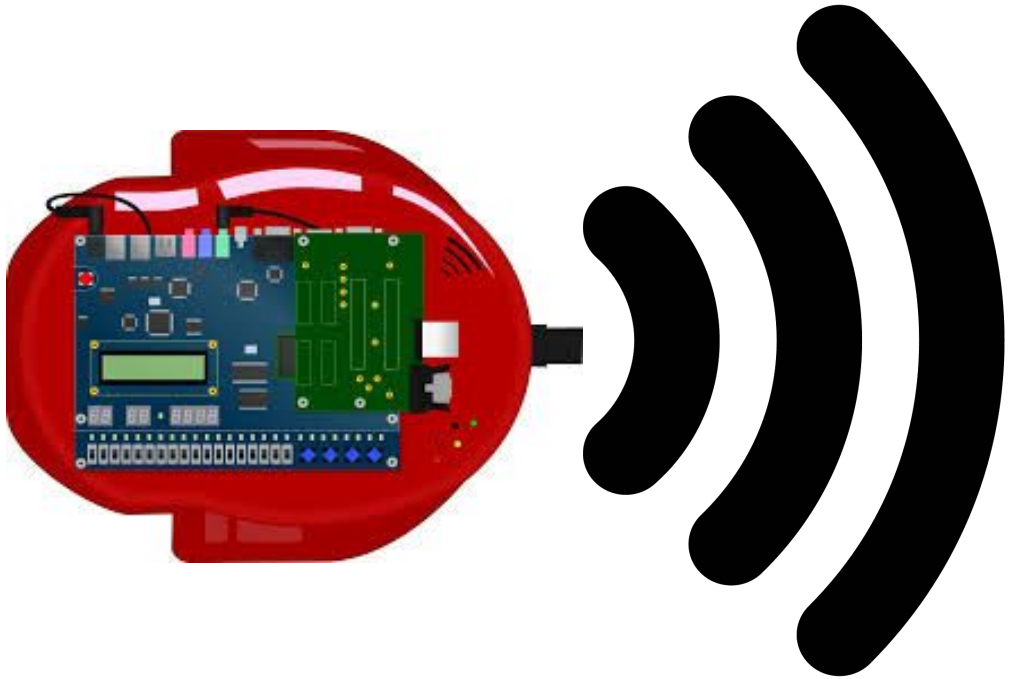


Image Source: http://powersof2.gatech.edu/resources/DE2Bot/DE2BotUsersManual_v5.pdf

Design Solution Overview

- Our solution may contain an option to specify in input data for how many reads or writes will occur
- Streamlined VHDL implementation of SRAM will make timing improvements via state machines and signal concatenations

Image Source: http://powersof2.gatech.edu/resources/DE2Bot/DE2BotUsersManual_v5.pdf

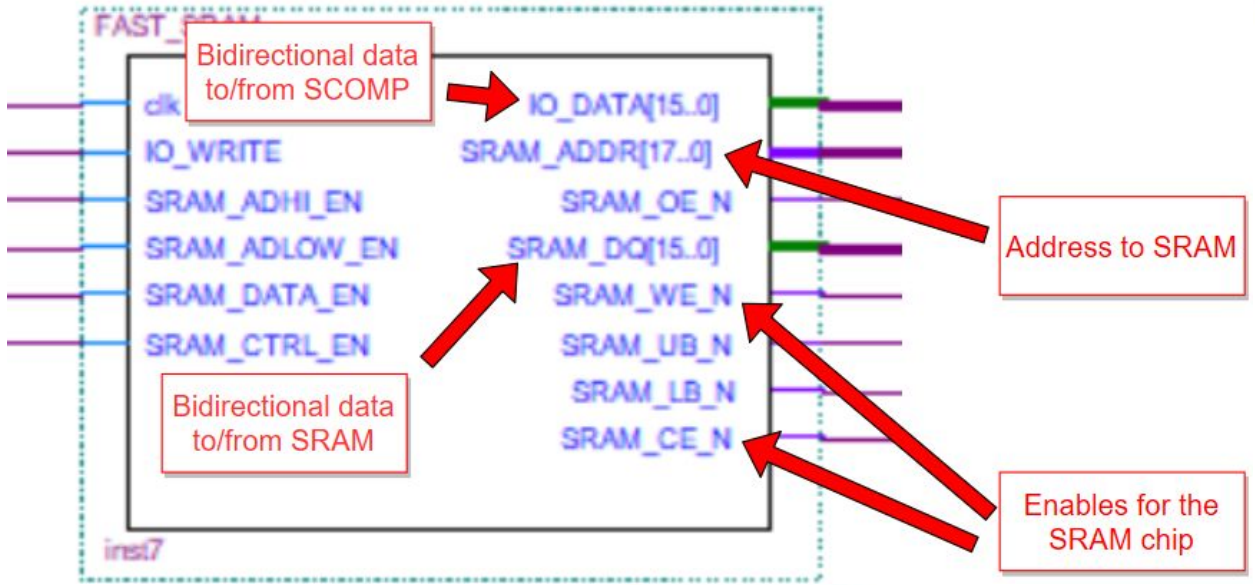
Technical Approach: State Machine

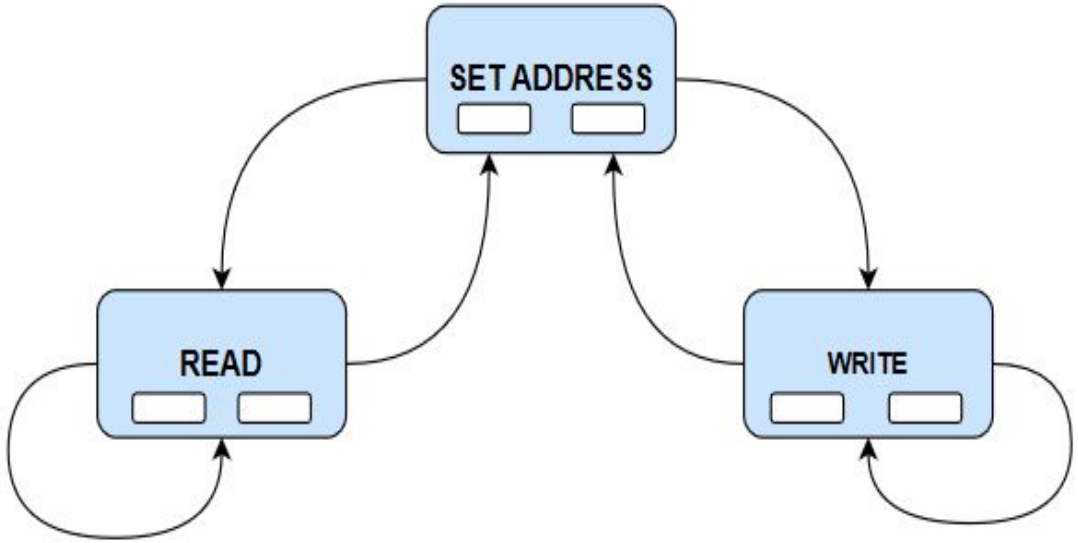
Inputs:

- io_data from decoder
- io_write from scomp
- enables from decoder

Moore outputs:

- sram_dq to sram
- sram_address to sram
- OE, and WE to sram
- io_data to scomp





Pros and Cons of State Machine

Pros:

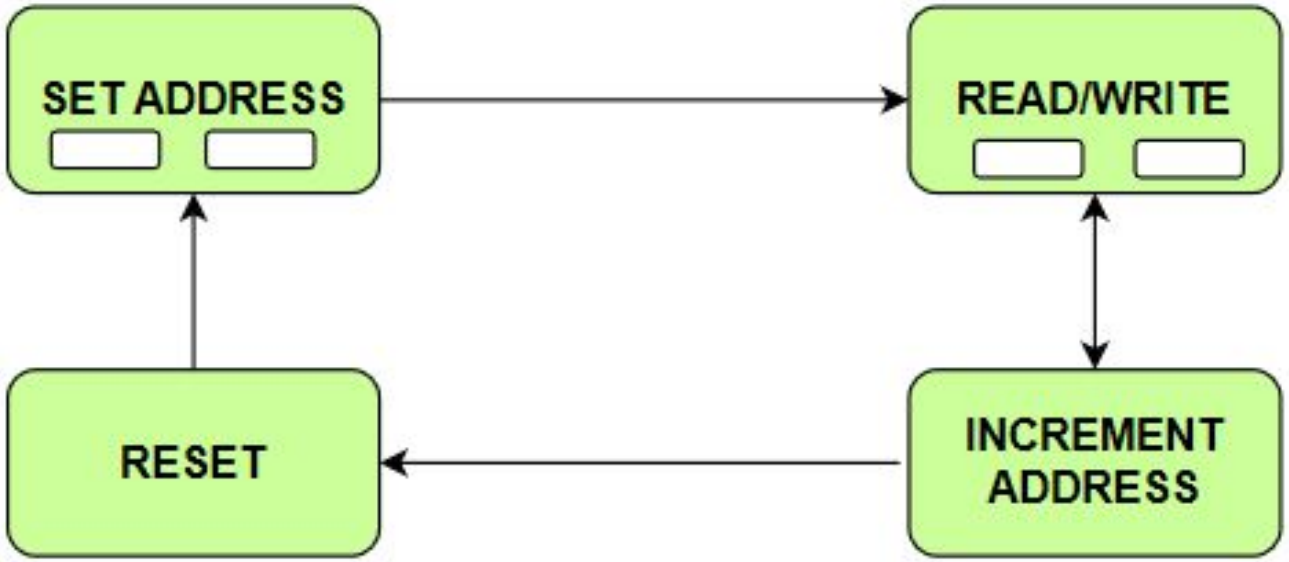
- Provides control over timing to meet requirements
- Utilizes simpler design than combinational logic
- Enables reuse of signals to decrease delays
- Allows user to execute sequential reads or writes

Cons:

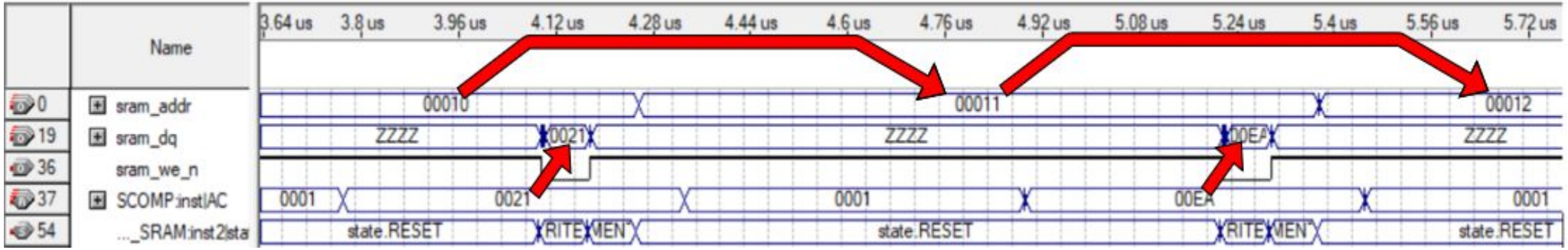
- Takes time to transition between states
- Requires many additional enable and internal signals

Sequential Read/Write

- Prepares the next write address after the current read/write
- Reduces the number of instructions required to read/write
- Requires fewer steps in the assembly code



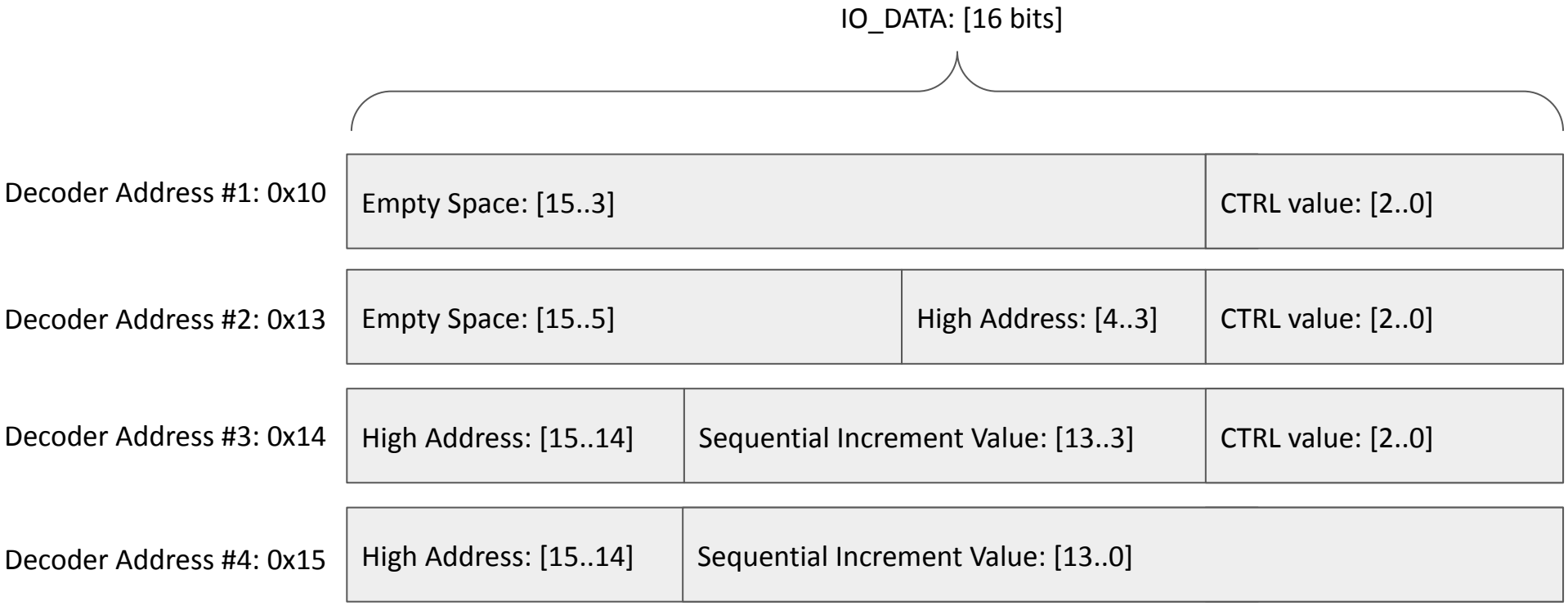
Sequential Write Example



- The data line is driven by the accumulator from the Simple Computer
- Write Enable is asserted (active low)
- Address is automatically incremented during an extra state
- SRAM is prepared to receive the next instruction

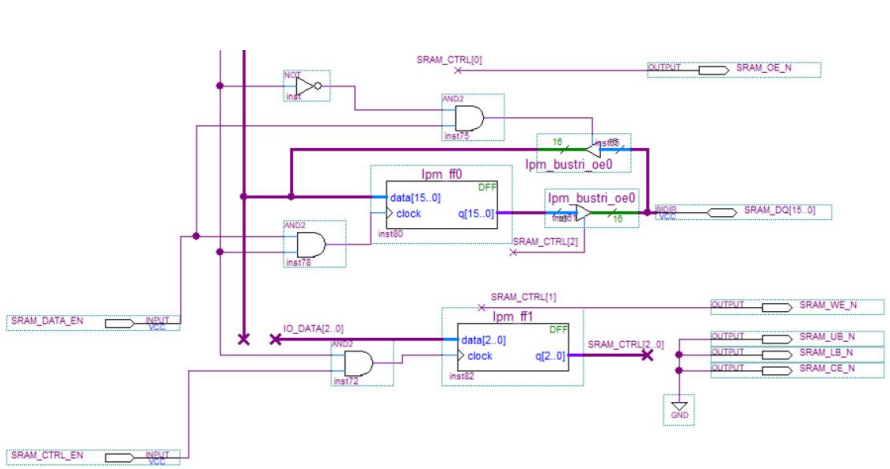
Technical Aspects: Signal Concatenation

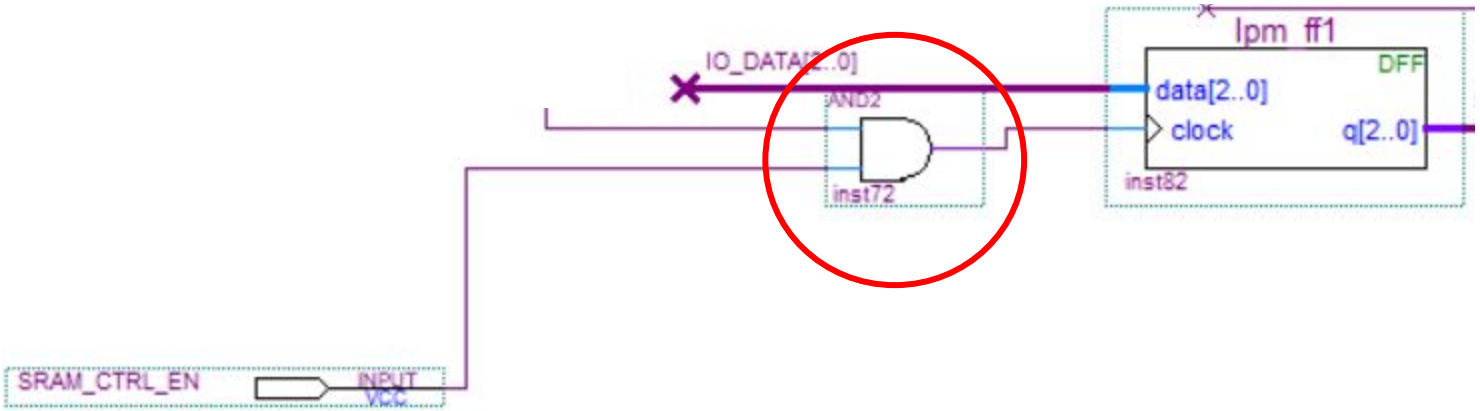
- Reducing the amount of LOADI instructions by combining values in the IO_DATA input



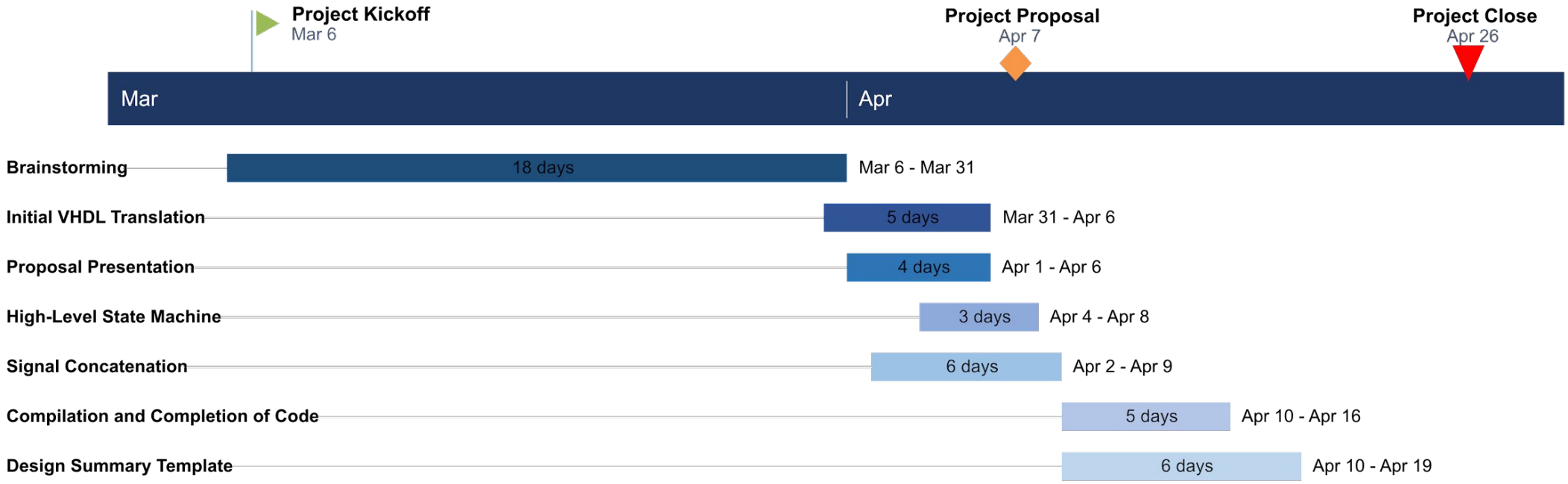
Technical Aspects: Reduced Cycling

- The improved SRAM we are creating has reduced logical dependence over time
  - Utilization of organized states reduces the need for redundant information to be passed through logic gates (each of which can take up to a half of a clock cycle)



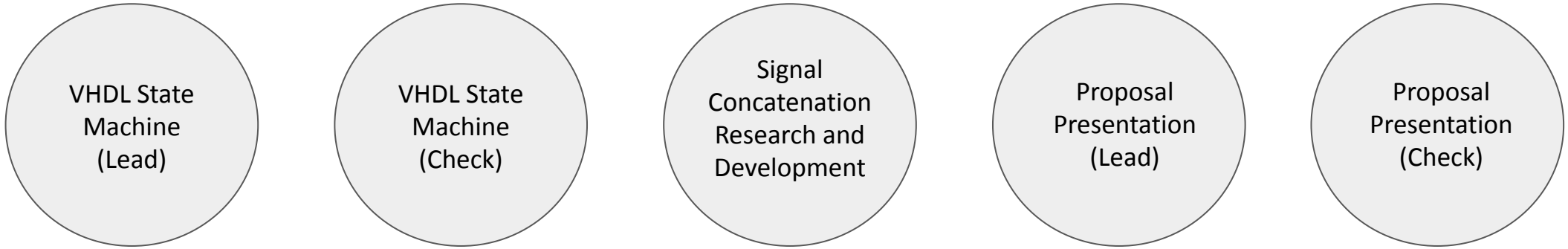


Timeline


Project Kickoff
Mar 6
Project Proposal
Apr 7
Project Close
Apr 26
Mar
Apr
Brainstorming
18 days
Mar 6 - Mar 31
Initial VHDL Translation
5 days
Mar 31 - Apr 6
Proposal Presentation
4 days
Apr 1 - Apr 6
High-Level State Machine
3 days
Apr 4 - Apr 8
Signal Concatenation
6 days
Apr 2 - Apr 9
Compilation and Completion of Code
5 days
Apr 10 - Apr 16
Design Summary Template
6 days
Apr 10 - Apr 19

Division of Labor

- Roles are assigned generally, with specific goals for each role set at every meeting
- Team members work on one of the following:
  - Main VHDL code efficiency
  - Exploration of alternative efficiency improvement methods
  - Project assignments
  - Code organization/readability



Sample Division of Labor for Main Roles

Contingency Plan and Final Thoughts

- In the case our idea does not work, we will implement our existing VHDL
  - Simulation proof of the sequential write capability working
- Basic write and sequential write capability that is ~2x faster than SLOW_SRAM.
- In the case read cannot be implemented, we will have to use the logic diagram to VHDL method for SRAM reads