ECE 385

Spring 2020

Experiment # 3

Logic Processor

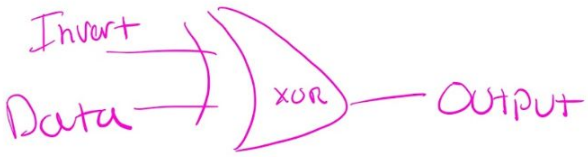
Megh Shah, Ishaan Patel
Lab Section: ABC, Tuesday 11:30 AM
TA: Gene Shiue, David Zhang

Introduction:

Computers have many functions; One of them being able to perform logic operations on bits to control low-level hardware. In this lab, we explored that concept. We built a circuit that takes in 4-bit data from the user, performs one of eight logic operations, such as AND, OR, XOR, "1111", NAND, NOR, XNOR, "0000", and then stores that data back into the 4-bit registers of the circuit. The purpose of building a circuit such as this is for us to become familiar with how a computer and CPU takes in data and manipulates it depending on what the user wants. Overall, a lab such as this provided us with new knowledge and a newfound curiosity.

Answers to Pre-Lab Questions:

A. One could use a XOR circuit to be used as a MUX that inverts the output if the invert bit is high.



B. Modular designs improve testability and cut down development time as they make the overall process of creating circuits more efficient. If we utilize modular designs, then essentially we are splitting the circuit up into "blocks" which we can remove and add, and are mostly independent of each other with the exception of maybe some inputs and outputs being dependent on each other. This modularity allows us to test each part of the circuit. If there is an error in the circuit, then we can easily determine where it occurs as each modular part is separated from one another. This improves testing as we do not have to check every part of the circuit since we can determine which block the issue might be coming from. It also cuts down development time as each block can be split within a group. One person can work on one block while someone else works on another, and in the end, they can combine the blocks together and create the final product. Finally, if an error occurs in a block, you can "Hot Swap" them out for a quick turnaround time.

Operation of Logic Processor:

A. To load data into either register A or register B, the user must first flip the LOAD A switch if they want to load into register A or flip the LOAD B switch if they want to load into register B. Afterwards, the user must decide on a 4-bit data that they want to load into either register, and flip the four data switches, D[3:0], that represent their selected data. Finally, after enough cycles have passed, the data

will be loaded into the selected register and the user can flip the LOAD A or LOAD B switches back down.

B. To initiate a computation and routing operation, the user must first load their desired data into registers A and B using the steps outlined above. Afterwards, the user must set the logic operation they desire by setting the three function switches, F[2:0], to the correct bit. Once those switches are set, the user must set the two routing switches, R[1:0], to represent where they want to output, if they choose to store it, to be stored. Finally, the user must flip the execute switch and wait four clock cycles for the circuit to compute the logic and route the output into their desired register.

Written description, High-Level Block Diagram, and State Machine Diagram

A. Written Description:

a. Control Unit: The circuit begins its execution of the logic operation and routing at the control unit. When the execute switch is flipped, the three D flip-flops, which determine the next state value for the current state, the counterbit C1, and the counterbit C0, begin determining what current state the state machine is in and what the next state will be. Depending on what state the machine is currently in and its inputs, the control unit will control the S0 pin on the shift registers which allow the data to either be loaded or shifted and operated on. Finally, if execute is flipped down during an execute cycle, the processor will still continue the operation.

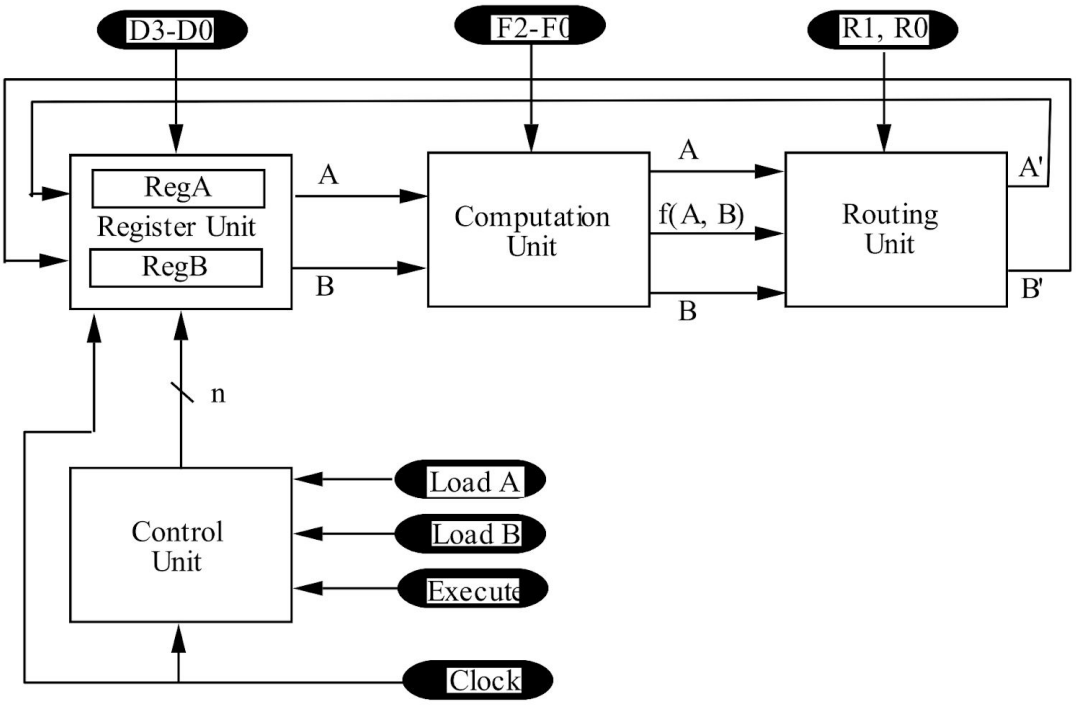
b. Register Unit: The register unit consists of two shift registers, one for register A and another for register B. The registers hold in the data that is set by the data switches. This data is parallel loaded into the registers when the LOAD A or LOAD B is flipped on. Afterwards, when the execute switch is flipped on, the data in the registers shift right for four cycles. The data leaves the registers and goes into the logic unit and routing unit, and then shifts back into the registers to hold data until the next operation.

c. Computation Unit: The computation unit consists of many NAND gates, NOR gates, and a 4-to-1 MUX. The unit takes two separate bits from the register units and determines which logic operation to perform on the two bits depending on the inputs of the function switches. A selection is made in the 4-to-1 MUX and the correct logic is performed which may be inverted depending on the inputs. After the logic is successfully computed, the output and the two original bits are then forwarded to the routing unit.

d. Routing Unit: The routing unit has the sole purpose of taking in five inputs: bit A, bit B, the output of the computation unit, and the routing switches R[1:0]. The routing switches serve as the selection for the 2-to-1 MUX. Depending on the output of the MUX, input bits A and B and the

output of the computation unit are then routed back into registers A and B. All this is done in 4 clock cycles.

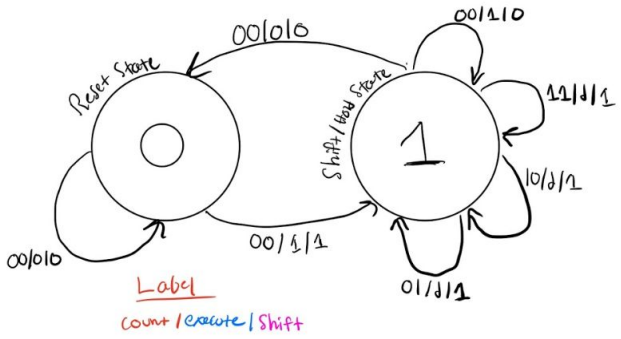
B. High-Level Block Diagram:



C. State Machine Diagram:

a. For this lab, we utilized a Mealy machine. We made this decision because when we derived our next-state expressions from the truth table, we saw that some of the expressions depended on the execute switch. Since the execute switch is an input in our circuit, and the next-state does not simply depend on the current state, our circuit was implemented using a Mealy machine.

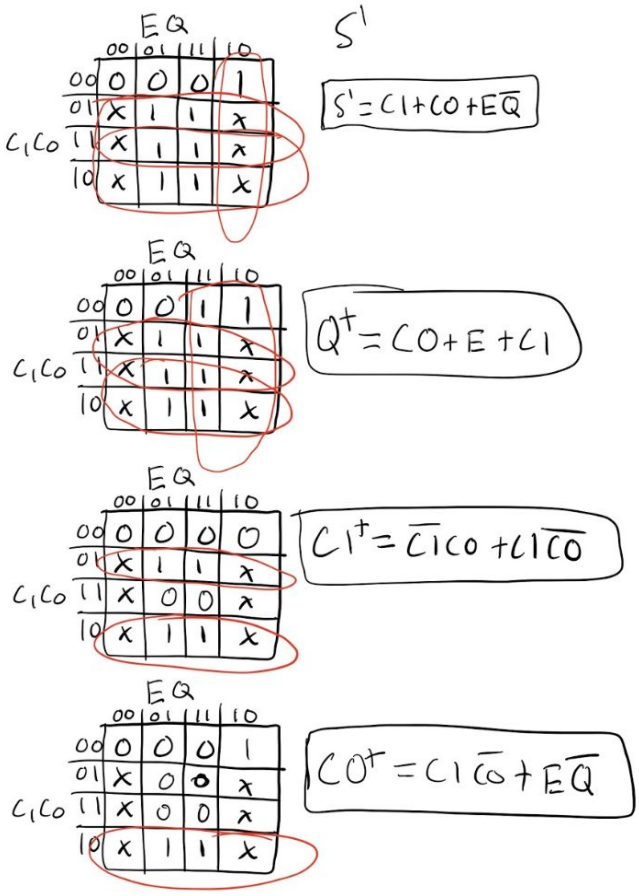
b.



Design Steps Taken and Detailed Circuit Schematic Diagram:

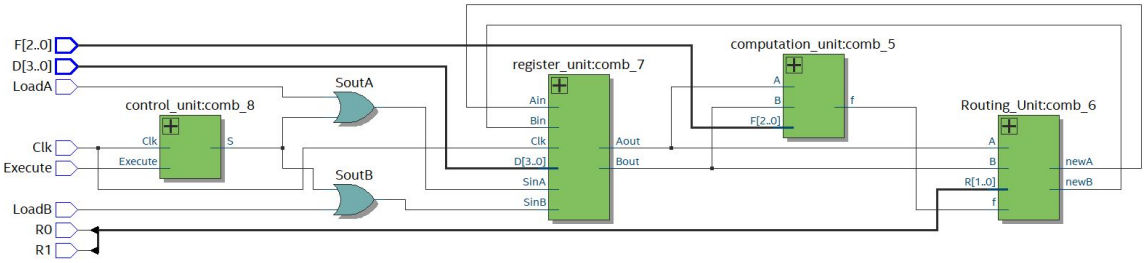
A. Design Steps Taken:

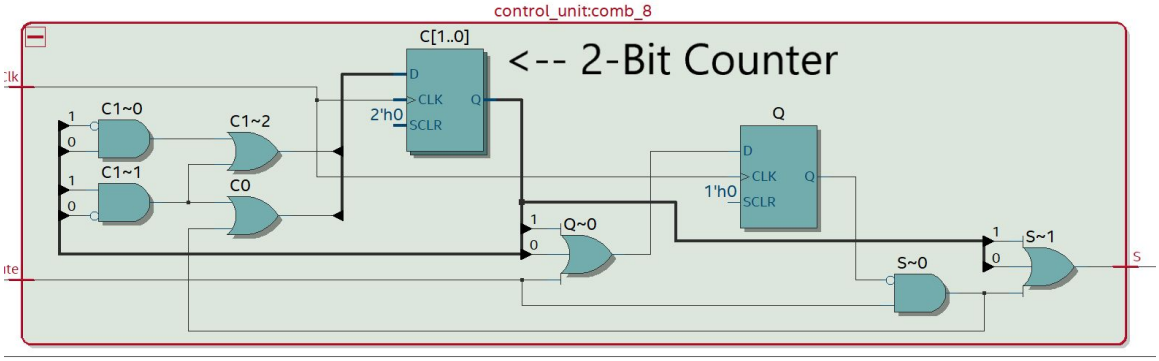
a.

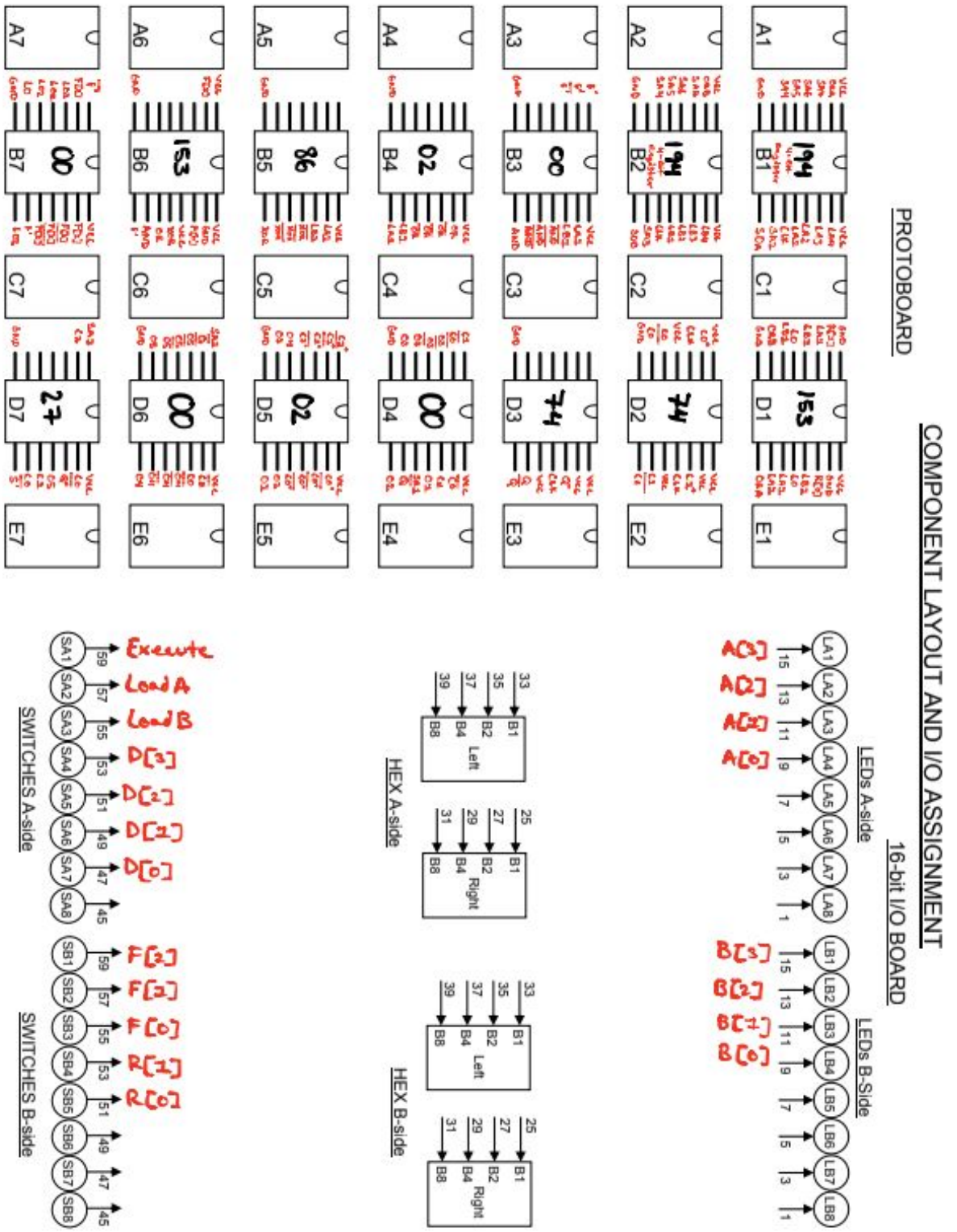


b. Regarding design steps that we took, we had a very linear process for most of our design considerations. The shift register, logic unit, and routing unit all have a straightforward implementation. However, for the logic unit, instead of using a 8-to-1 MUX to incorporate the 8 logic functions, we used a 4-to-1 MUX to select four of the logic functions, and the most significant of the function switches was combined with combinational logic which determine whether the output of the MUX should be inverted or not since half of the logic operations were inverses of one another. Also, for the control unit, we decided to create our own 2-bit counter instead of using the 4-bit counter chip and having it work as a 2-bit counter. This was because we used K-maps to determine the next state for the counter and found that it was dependent on the execute signal, so we found it easier to use two D Flip-Flops for it

B. Detail Circuit Schematic Diagram

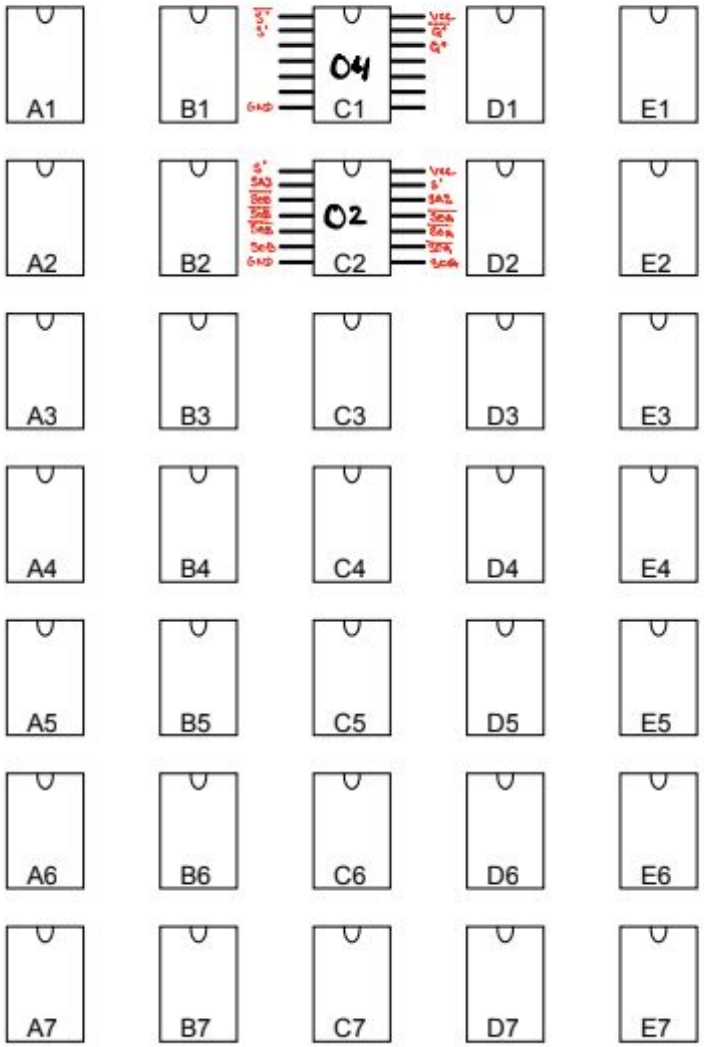




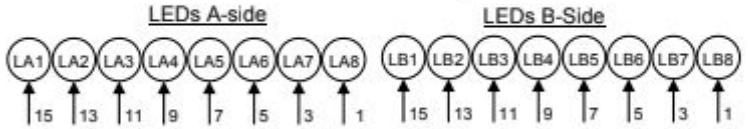

COMPONENT LAYOUT AND I/O ASSIGNMENT
PROTOBOARD
16-bit I/O BOARD
LEDs A-side
LEDs B-Side
A[3]
A[2]
A[1]
A[0]
B[3]
B[2]
B[1]
B[0]
HEX A-side
HEX B-side
Left
Right
SWITCHES A-side
SWITCHES B-side
Execute
Load A
Load B
D[3]
D[2]
D[1]
D[0]
F[2]
F[1]
F[0]
R[1]
R[0]

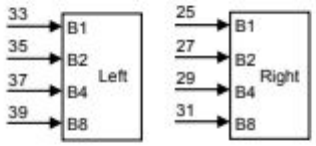
COMPONENT LAYOUT AND I/O ASSIGNMENT

PROTOBOARD

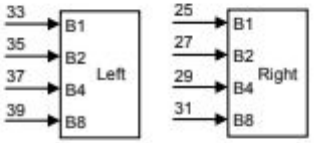


16-bit I/O BOARD

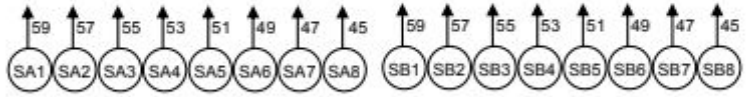




HEX A-side



HEX B-side



SWITCHES A-side

SWITCHES B-side

Description of All Bugs Encountered and Corrective Measures Taken:
Regarding this lab, we built the shift registers, logic unit, and routing unit first. Once this was built, we tested this part individually, and we discovered that everything worked as intended. There were no bugs present in this part of the circuit. However, we did run into errors when designing the control unit. For the control unit, we initially designed it to include the 74LS169 chip which is a 4-bit counter. We wired it up in a way where it would be used as a 2-bit counter. However, we had issues with this chip as it did not work as intended. The chip would constantly have an output of "1". We determined that this was a logic and design error on our part. Due to this, we removed this chip and utilized two D Flip-Flops to create our own counter. We had one for the first bit of the counter, C0, and one for the second bit of the counter, C1. After we created our own counter, our control unit worked as it was supposed to.

Answers to Post-Lab Questions:

A. Regarding our Pre-Lab writeup, we believe that we were mostly right in our thinking of modular designs. Since we had a modular approach regarding this lab, we did not have many difficulties debugging our circuit. We were able to build each component, such as the shift registers, logic unit, routing unit, and control unit, separately and test them individually using test inputs. This let us identify where the bugs were in the circuit. For example, when creating the logic unit and routing unit, we found out that they did not work initially. However, since we had already created the logic unit first and tested it and knew that it was functional, we knew the error must be in the routing unit. We tested the routing unit and it turned out one of the pins on the chips was broken. All in all, building a modular design circuit is helpful as we can build each part individually and test it, so we can easily identify bugs and speed up development time.
B. In our machine, we decided to implement a Mealy machine as our states depend on the input as well as the current state. Using a Mealy machine allows us to have less states than compared to a Moore machine. However, one drawback to Mealy machines is that since they are dependent on the input as well, they are asynchronous. Moore machines are synced with the clock so they are updated with each clock cycle while Mealy machines are not.

Conclusion:
Overall, we felt this lab encompassed all the fundamental ideas of TTL chips and building physical circuits on the breadboard. Not only did we design and implement a complex circuit using nothing but low-level chips, but we used new chips or we found new ways to implement existing concepts, such as us creating our own 2-bit counter. We also learned how to incorporate theoretical concepts that we have not before, such as translating a Mealy machine from a state diagram to a physical circuit using D Flip-Flops. All in all, this lab taught us a lot regarding logic processing, TTL chips, and state machines.