ENGIN 112

Intro to Electrical and Computer Engineering

Lecture 33

Arithmetic Logic Unit (ALU)

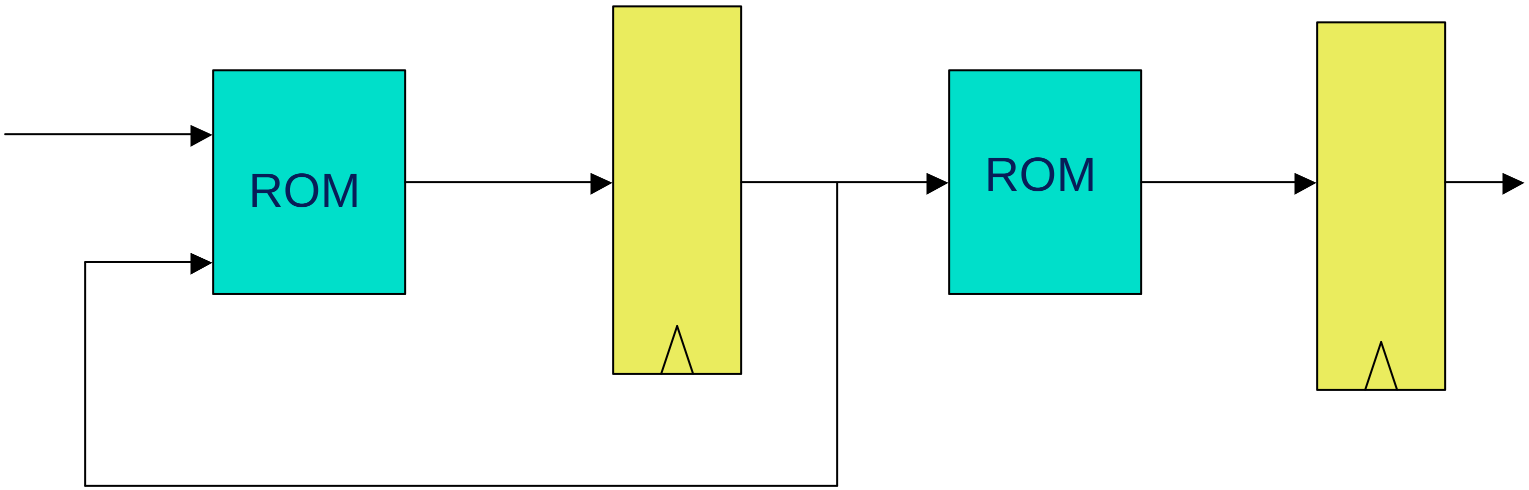
ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF MASSACHUSETTS AMHERST

Overview

- Main computation unit in most computer systems
- ALUs perform a vaiety of different functions
  - Add, subtract, OR, AND…
- Example: ALU chip (74LS382)
  - Has data and control inputs
- Individual chips can be chained together to make larger ALUs
- ALUs are important parts of datapaths
  - ROMs often are usd in the control path
- Build a data and control path

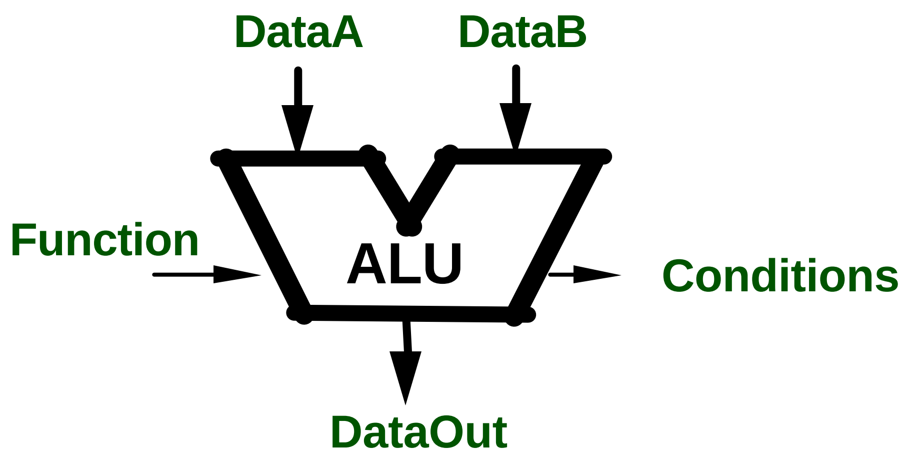
ROM-based Moore Machine Timing

- What is the maximum clock frequency of this circuit?
- Does this circuit satisfy hold time constraints?

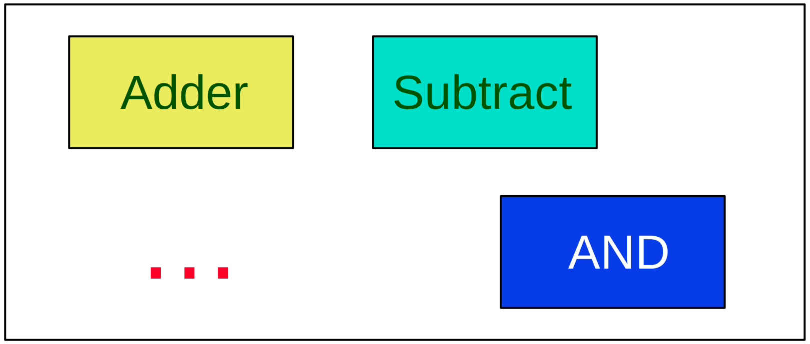


Arithmetic Logic Unit

- Arithmetic logic unit functions
  - Two multi-bit data inputs
  - Function indicates action (e.g. add, subtract, OR…)
- DataOut is same bit width as multi-bit inputs (DataA and DataB)
- ALU is combinational
- Conditions indicate special conditions of arithmetic activity (e.g. overflow).

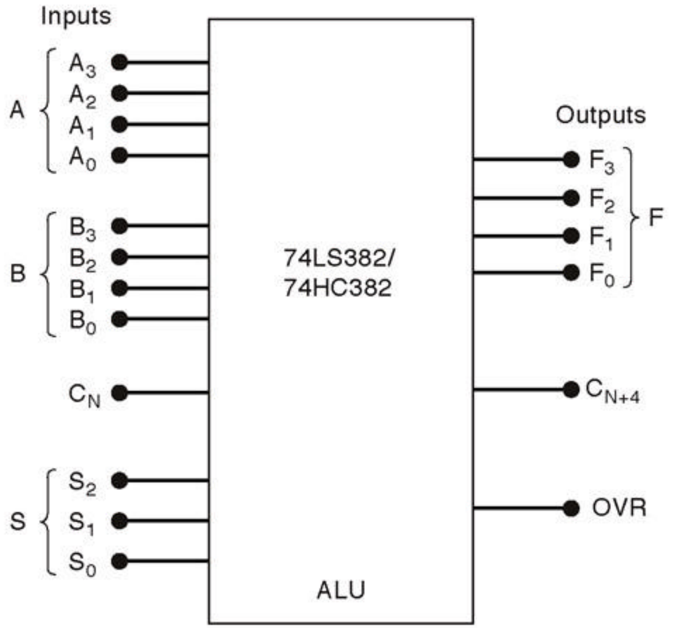


Think of ALU as a number of other arithmetic and logic blocks in a single box! Function selects the block



ALU Integrated Circuit

- Integrated circuit – off-the-shelf components
- Examine the functionality of this ALU chip



A = 4-bit input number
B = 4-bit input number
C_N = carry into LSB position
S = 3-bit operation select inputs
F = 4-bit output number
C_{N+4} = carry out of MSB position
OVR = overflow indicator

(a)

Function Table

S_2	S_1	S_0	Operation	Comments
0	0	0	CLEAR	$F_3F_2F_1F_0 = 0000$
0	0	1	B minus A	Needs $C_N = 1$
0	1	0	A minus B	Needs $C_N = 1$
0	1	1	A plus B	Needs $C_N = 0$
1	0	0	$A \oplus B$	Exclusive-OR
1	0	1	A + B	OR
1	1	0	AB	AND
1	1	1	PRESET	$F_3F_2F_1F_0 = 1111$

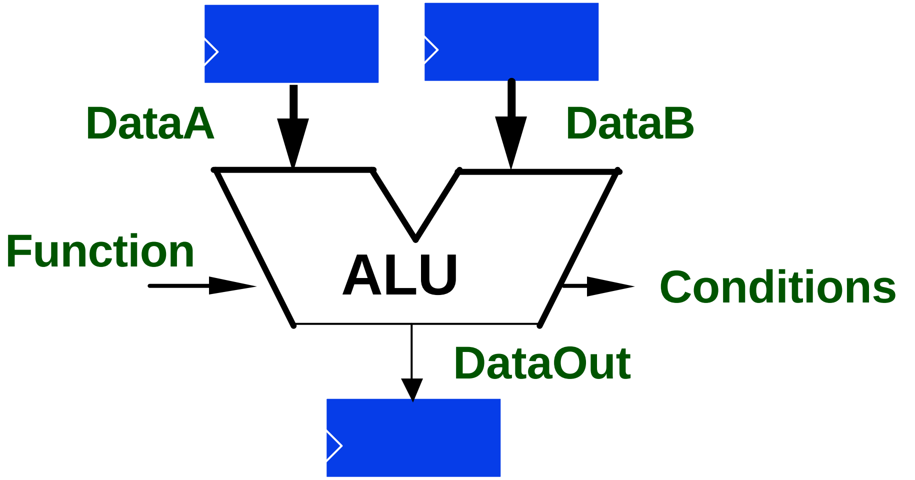
Notes: S inputs select operation.
OVR = 1 for signed-number overflow.

(b)

Performs 8 functions

Example

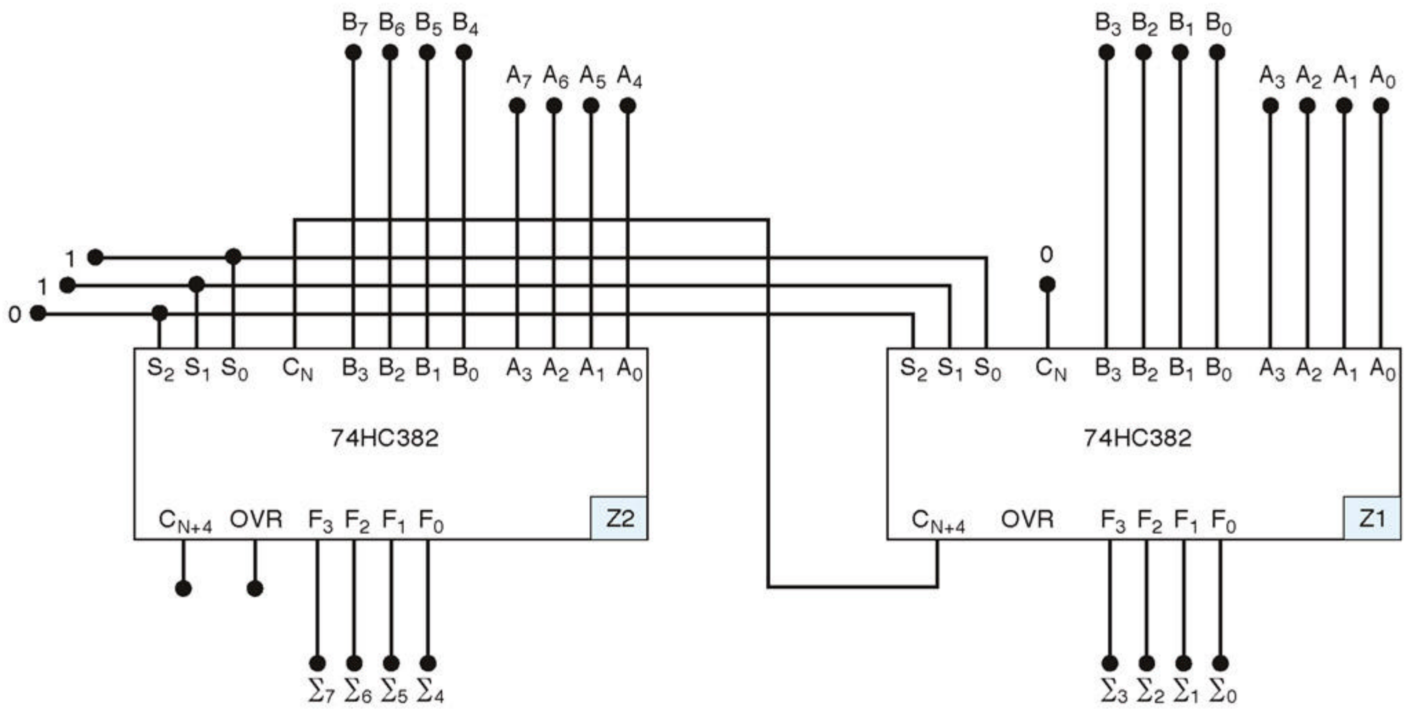
- Determine the 74HC382 ALU outputs for the following inputs: $S_2S_1S_0$=010, $A_3A_2A_1A_0$=0100, $B_3B_2B_1B_0$=0001, and C_N=1.
  - Function code indicates subtract
  - 0100 – 0001 = 0011
- Change the select code to 101 and repeat.
  - Function code indicates OR
  - 0100 OR 0001 = 0101



Synchronize ALU with a clock

Expanding the ALU

- Multi-bit ALU created by connecting carry output of low-order chip to carry in of high order

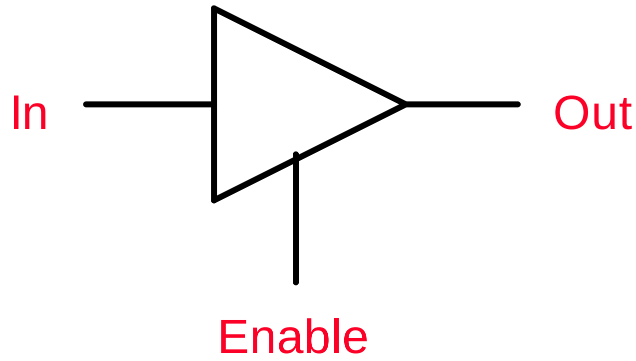


Notes: Z1 adds lower-order bits.
Z2 adds higher-order bits.
Σ_7–Σ_0 = 8-bit sum.
OVR of Z2 is 8-bit overflow indicator.

Eight-bit ALU formed from 2 four-bit ALUs

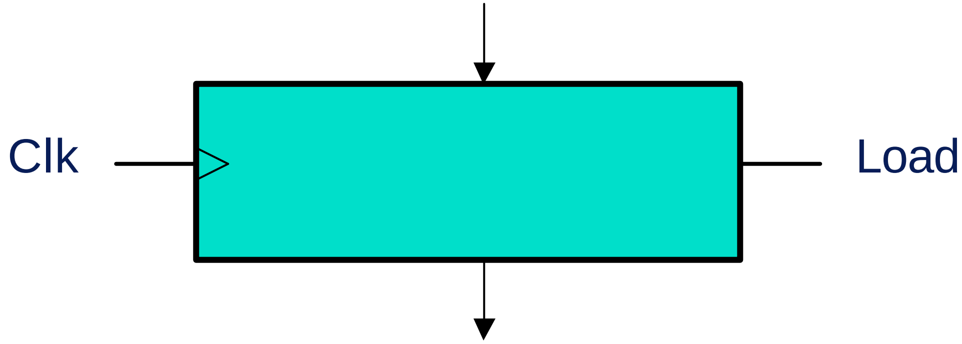
Datapath components

- **Tri-state buffer**



If Enable asserted,
Out = In
Otherwise
Out open-circuit

- **Loadable register**



Data stored on rising edge if Load is asserted (e.g. Load = 1)

Computation in a Typical Computer

- Control logic often implemented as a finite state machine (including ROMs)
- Datapath contains blocks such as ALUs, registers, tri-state buffers, and RAMs
- In a processor chip often a 5 to 1 ratio of datapath to control logic

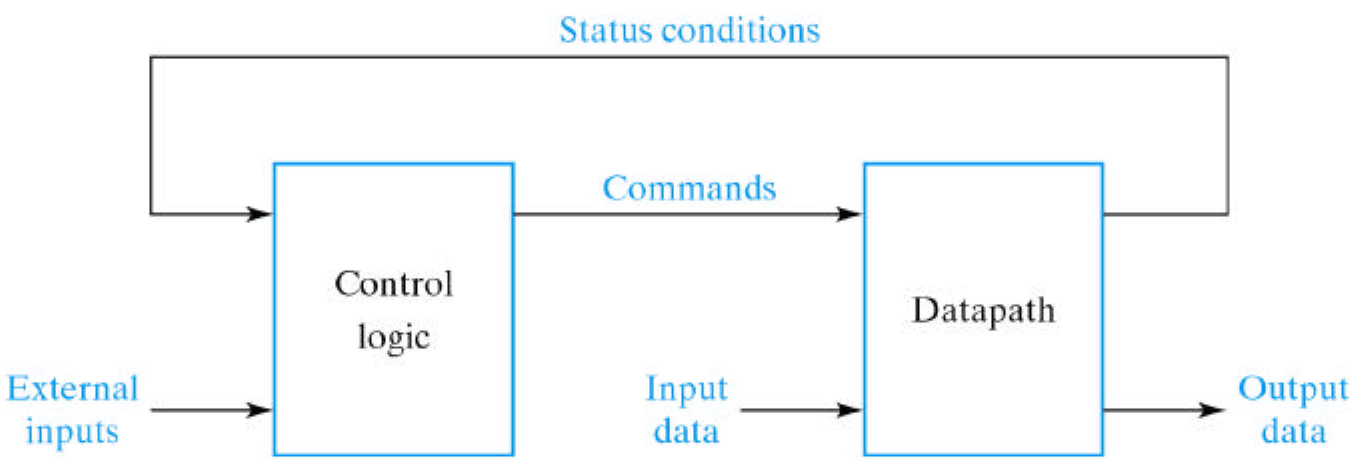
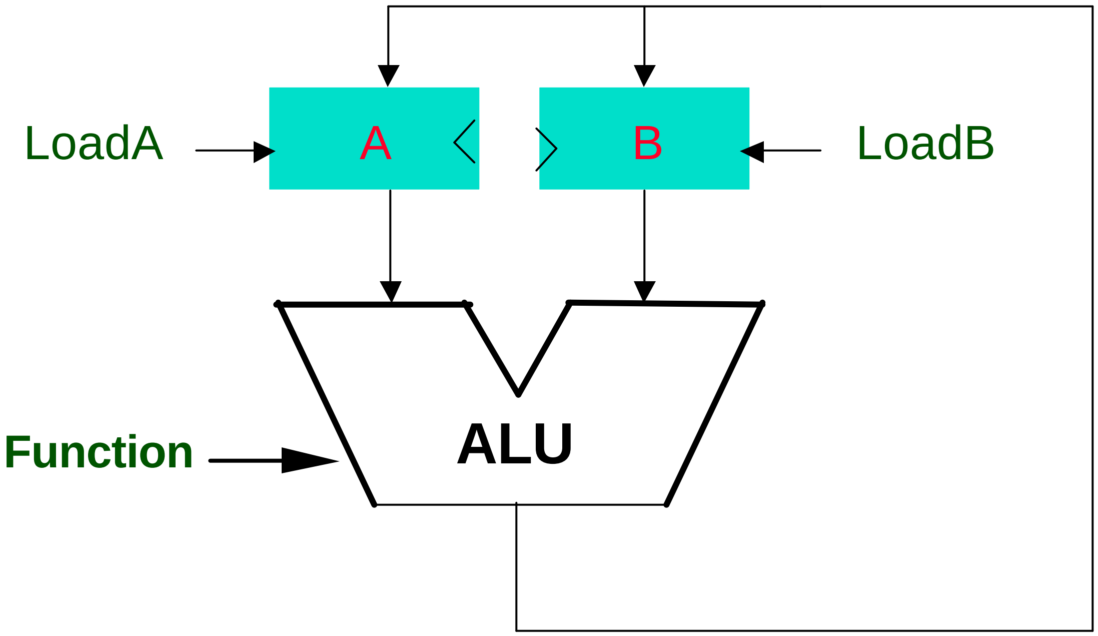


Fig. 8-2 Control and Datapath Interaction

Using a Datapath

- **Consider the following computation steps**
  1. **ADD A, B and put result in A**
  2. **Subtract A, B and put result in B**
  3. **OR A, B put result in A**
  - **Repeat starting from step 1**

Determine values for Function, LoadA, LoadB



Modeling Control as a State Machine

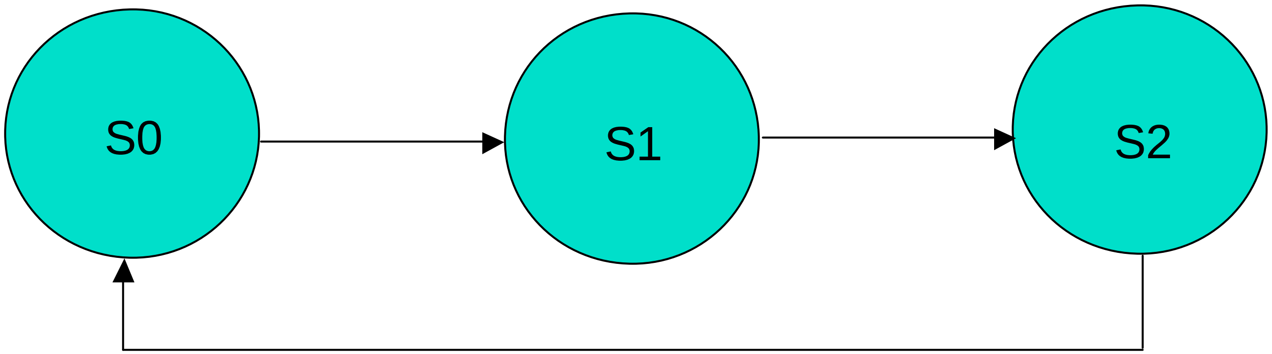
° **Consider the following computation steps**

1. **ADD A, B and put result in A**
2. **Subtract A, B and put result in B**
3. **OR A, B put result in A**

- **Repeat starting from step 1**

Determine values
for Function, LoadA, LoadB

Model control as a state machine.
Determine control outputs for each state



Modeling Control as a State Machine

- **Consider the following computation steps**
  1. **ADD A, B and put result in A**
  2. **Subtract A, B and put result in B**
  3. **OR A, B put result in A**
  - **Repeat starting from step 1**

States
S0 = 00
S1 = 01
S2 = 10

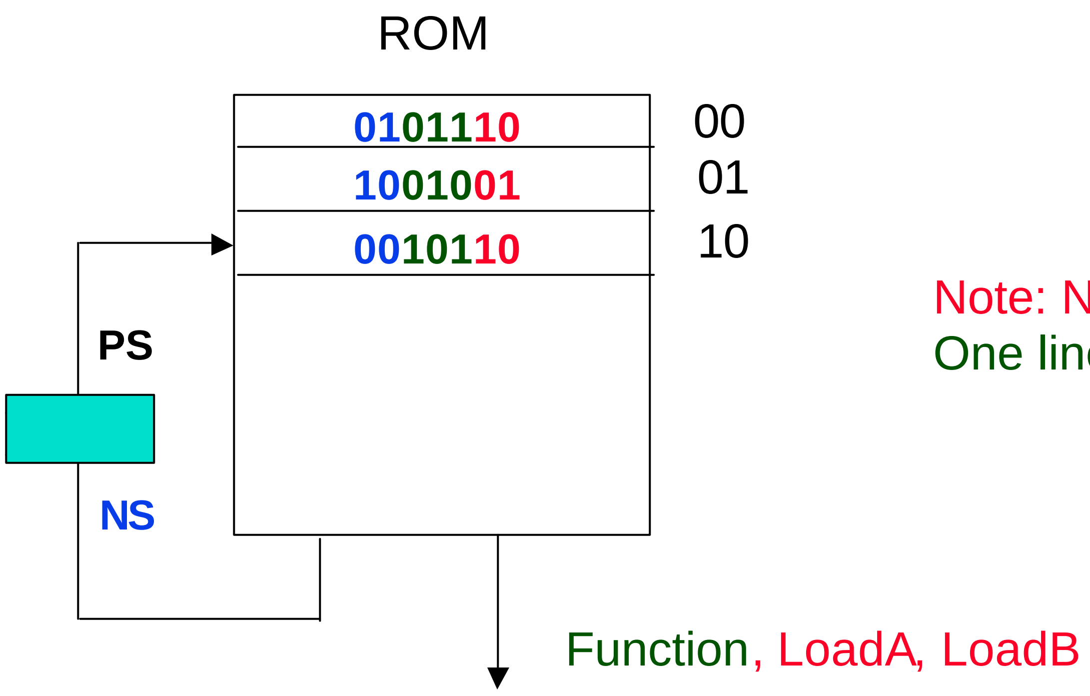
Present State	Next State	Function	LoadA	LoadB
00	01	011	1	0
01	10	010	0	1
10	00	101	1	0

We know how to implement this using an SOP.
Can we use a ROM?

ROM Implementation of State Machine

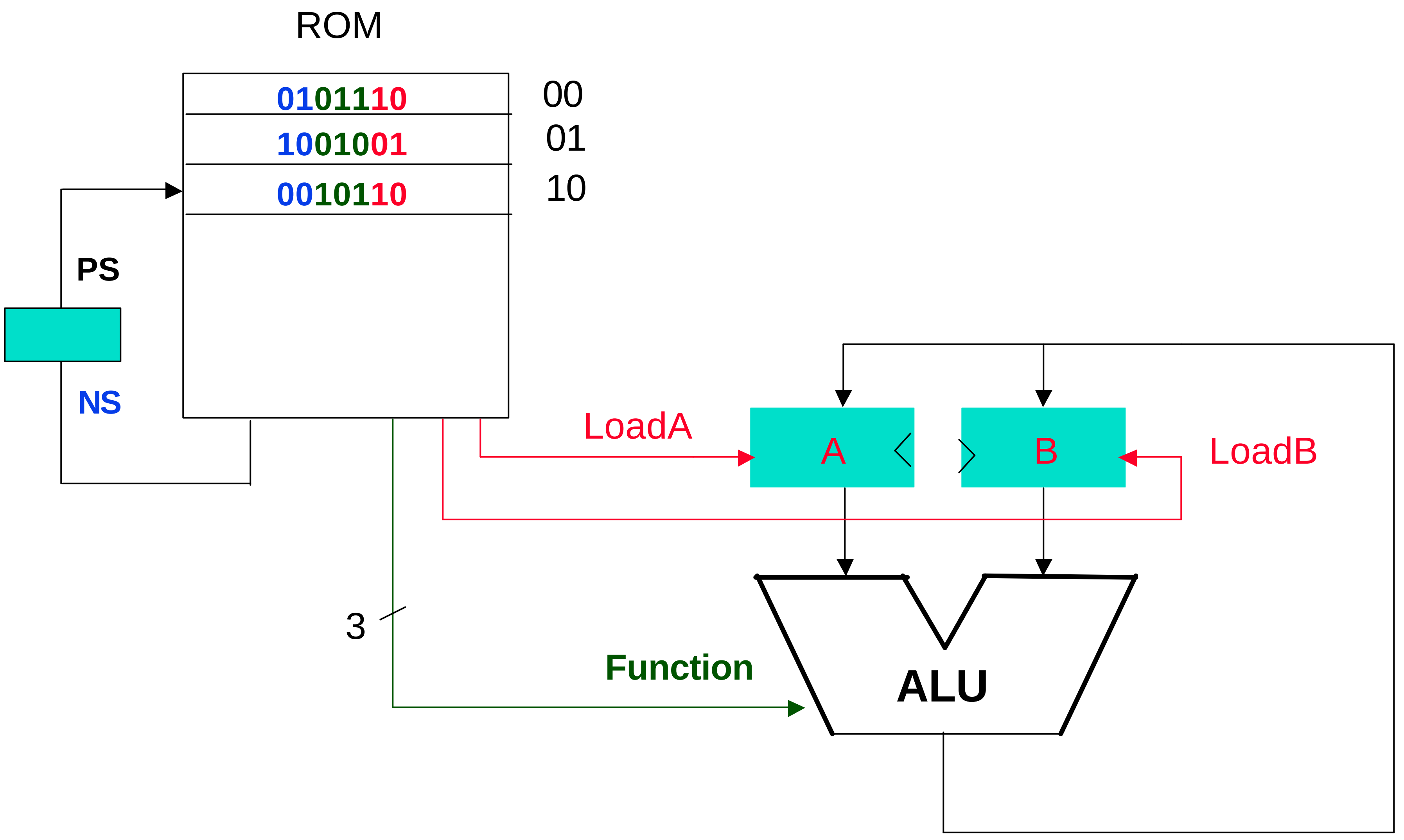
Present State	Next State	Function	LoadA	LoadB
00	01	011	1	0
01	10	010	0	1
10	00	101	1	0

States
S0 = 00
S1 = 01
S2 = 10

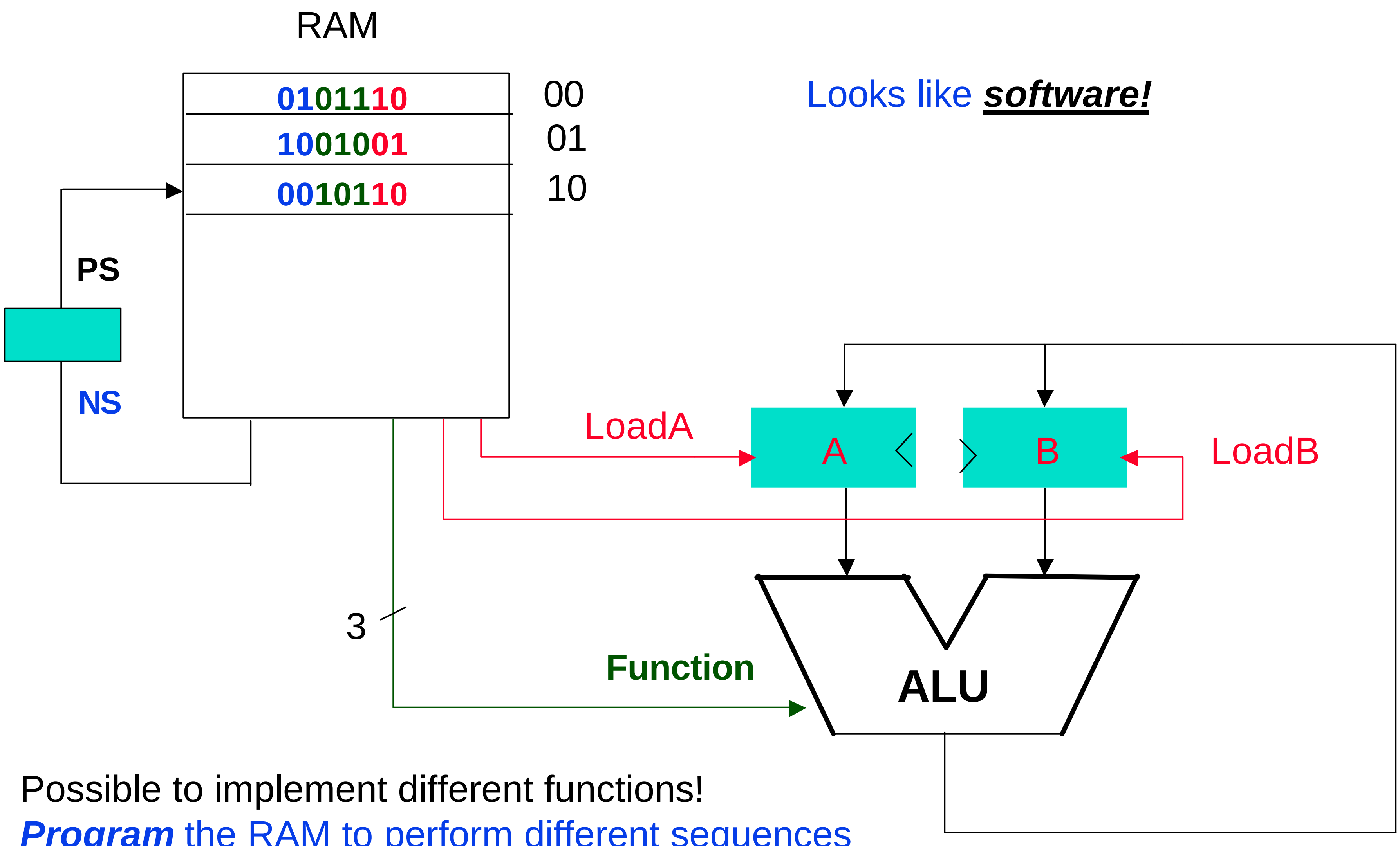


Note: No minimization!
One line in ROM for each state

Putting the Control and Datapath Together



What if we replaced the ROM with RAM?



Looks like ***software!***

Possible to implement different functions!

Program the RAM to perform different sequences

Summary

- ALU circuit can perform many functions
  - Combinational circuit
- ALU chips can be combined together to form larger ALU chips
  - Remember to connect carry out to carry in
- ALUs form the basis of datapaths
- ROMs can form the basis of control paths
- Combine the two together to build a computing circuit
- Next time: more data and control paths