Lab 3: Integrator Example

Digital Design Lab

Overview

- Lab 3 methodology
- Integrator example
    - Lab procedure walkthrough

Methodology

For the behavioral description of the Integrator:

1. Refresh your memory on FSMD definition and design (see FSMD videos)
2. Create VHDL behavioral description for the Integrator (see prelab FSMD video). (You do not need to account for delay in the behavioral description.)
3. Create assert-based test bench for the Integrator.
4. Simulate your design. Repeat 1-3 until your result is correct.

For the structural description of the Integrator:

5. Describe Integrator FSMD with one state per clock cycle and data operation in each state.
6. Convert operation statement in each state to set of control signals for the datapath.
7. Generate controller FSM with control signal values in each state.
8. Estimate delays of each Boolean equations for each control signal using AND-OR and NOT gates
9. Estimate delays of Boolean equations for next-state signals using AND-OR and NOT gates
10. Create structural description for the Integrator and indicate the Integrator 's clock cycle in your code (there is a commented section for you to fill out).
11. Create assert-based test bench for the structural description.
12. Simulate your design. Repeat 5-11 until your result is correct.

Integrator: Problem Description

- In this assignment, you will design an 8x16 "Integrator" for eight 16-bit two's complement numbers. This Integrator has three 1-bit inputs namely **Start**, **Rst** and **Clk** as well as a 16-bit output **Sum** and a 1-bit output **Done**. It also contains an **8x16 Register File** (**RegFile**) that stores the numbers 19, 17, 13, 11, 7, 5, 3, and 2 in **regArray(0)** to **regArray(7)**.
- When **Rst** = '1', the Integrator's state is set to the initial state in which the Integrator is ready to start the computation.
- When **Start** = '1', the Integrator starts computing the following summation with the numbers stored in **RegFile** and generates the result “7" in 7 clock cycles:

$$\sum_{i=0}^{3} (\mathrm{regArray}(2i) - \mathrm{regArray}(2i+1))$$

- The intermediate results of the formula are stored in **RegFile** during the calculation. When the last arithmetic operation is computed, the signal **OutReg_Ld** is set to '1', and the result is loaded into the **Output Register** (**OutReg**).
- On the 8th clock cycle, the **Oe** signal becomes '1' and the result is available at the output **Sum** port for 1 clock cycle. The **Sum** port is set to high impedance (or to all Zs) at any other time. Similarly, the output signal **Done**(indicating the availability of the result on the **Sum** port) is set to '1' for that clock cycle while it is set to '0' at any other time.

Integrator: Delay Information

Figure 1 shows the structure of the Integrator and its components. The timing diagram of the integrator is displayed in **Figure 2**. These figures are shown on the next slide.

The delays for each unit in the structural model are as follows:

- **Combinational Logic (CombLogic)**: 4 ns for 4-input, 4-wide AND-OR gates, and 1 ns for NOT gates
- **State Register (StateReg)**: 4 ns from Clk to output, 1 ns setup time
- **8x16 Register File (RegFile)**: 6 ns from Clk to output, 1 ns setup time
- **ALU (ALU)**: 8 ns from input to output
- **Output Register (OutReg)**: 4 ns from Clk to output; 1 ns setup time
- **16-bit Three-state Buffer (ThreeStateBuff)**: 1 ns from input to output

*Note about setup time: setup time is the amount of the time that the signal must be stable (i.e. hasn't changed) before the rising edge of the clock (see the 31 textbook).

Integrator: Figures

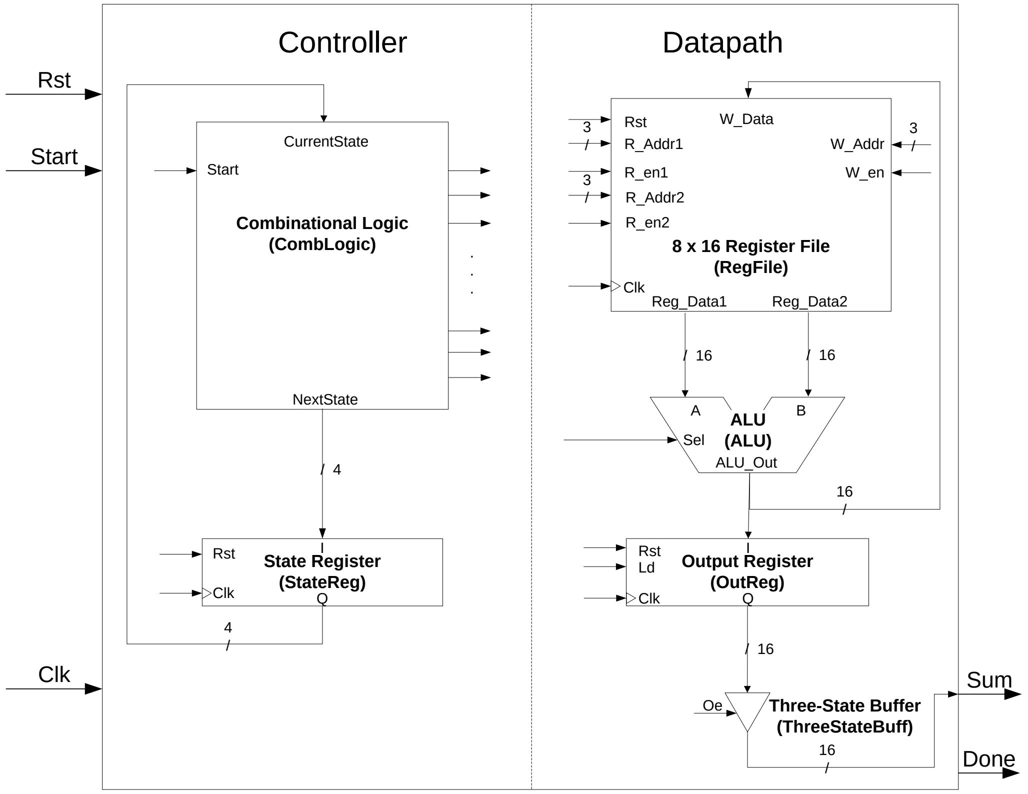


Figure 1: Integrator

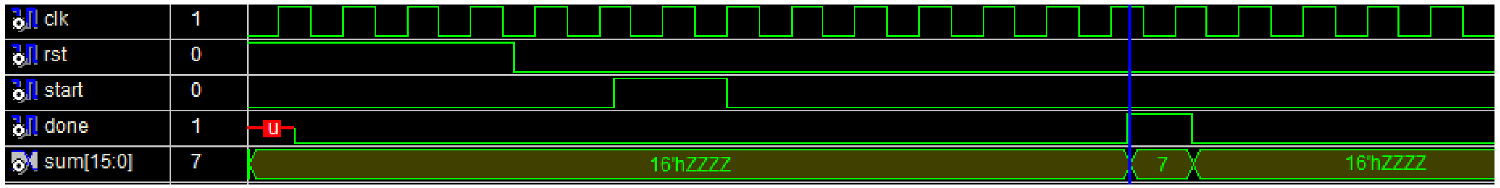


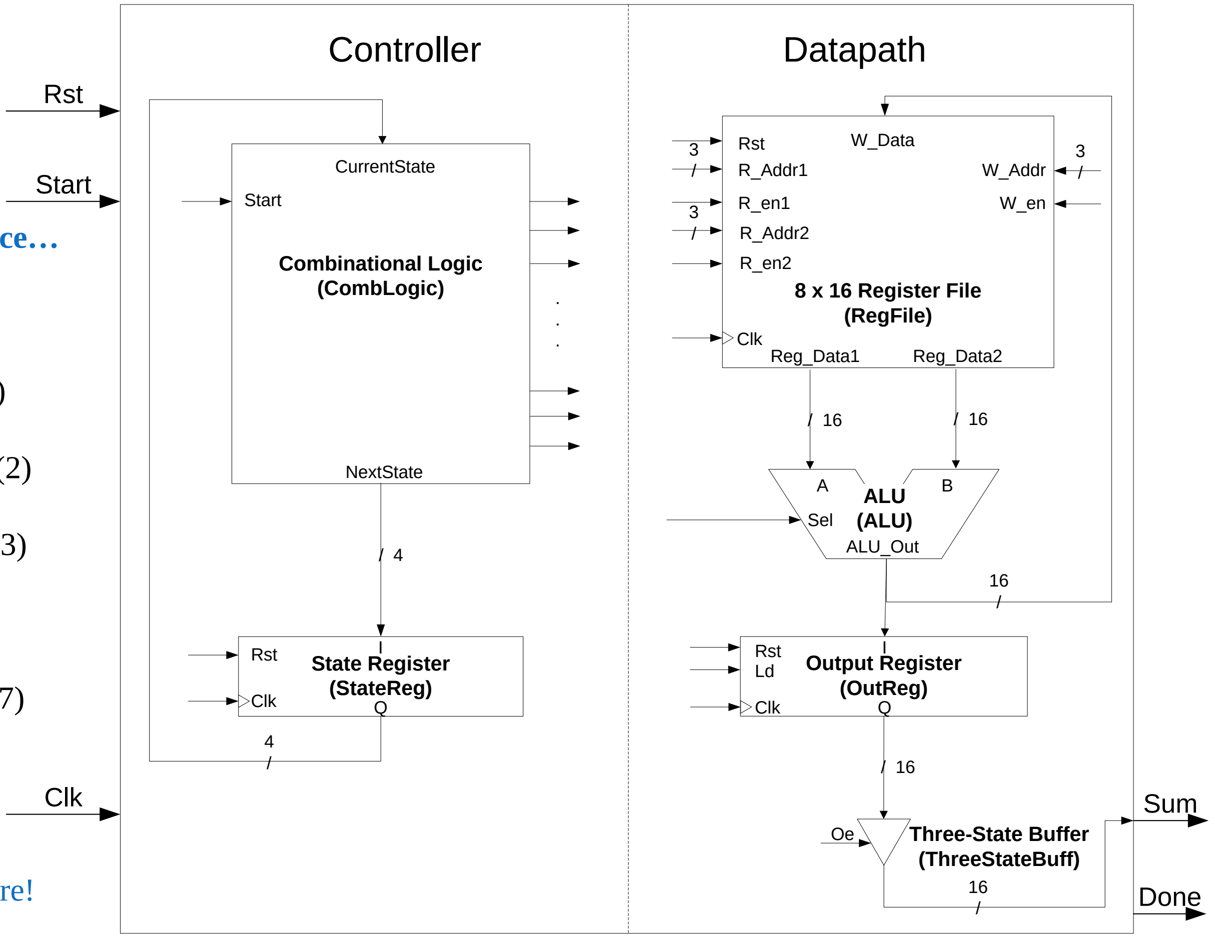
Figure 2: Timing Diagram for Integrator

1-5. Understanding Integrator Behavior

Compute this equation: $\sum_{i=0}^{3} (\text{regArray}(2i) - \text{regArray}(2i+1))$

(regArray(0) – regArray(1)) + (regArray(2) – regArray(3)) + (regArray(4) – regArray(5)) + (regArray(6) – regArray(7))

Using this integrator:



How? ALU only does 2 operations at once…

Initialize: get ready to do computation

1st operation: (regArray(0) – regArray(1))

2nd operation: previous result + regArray(2)

3rd operation: previous result – regArray(3)

…

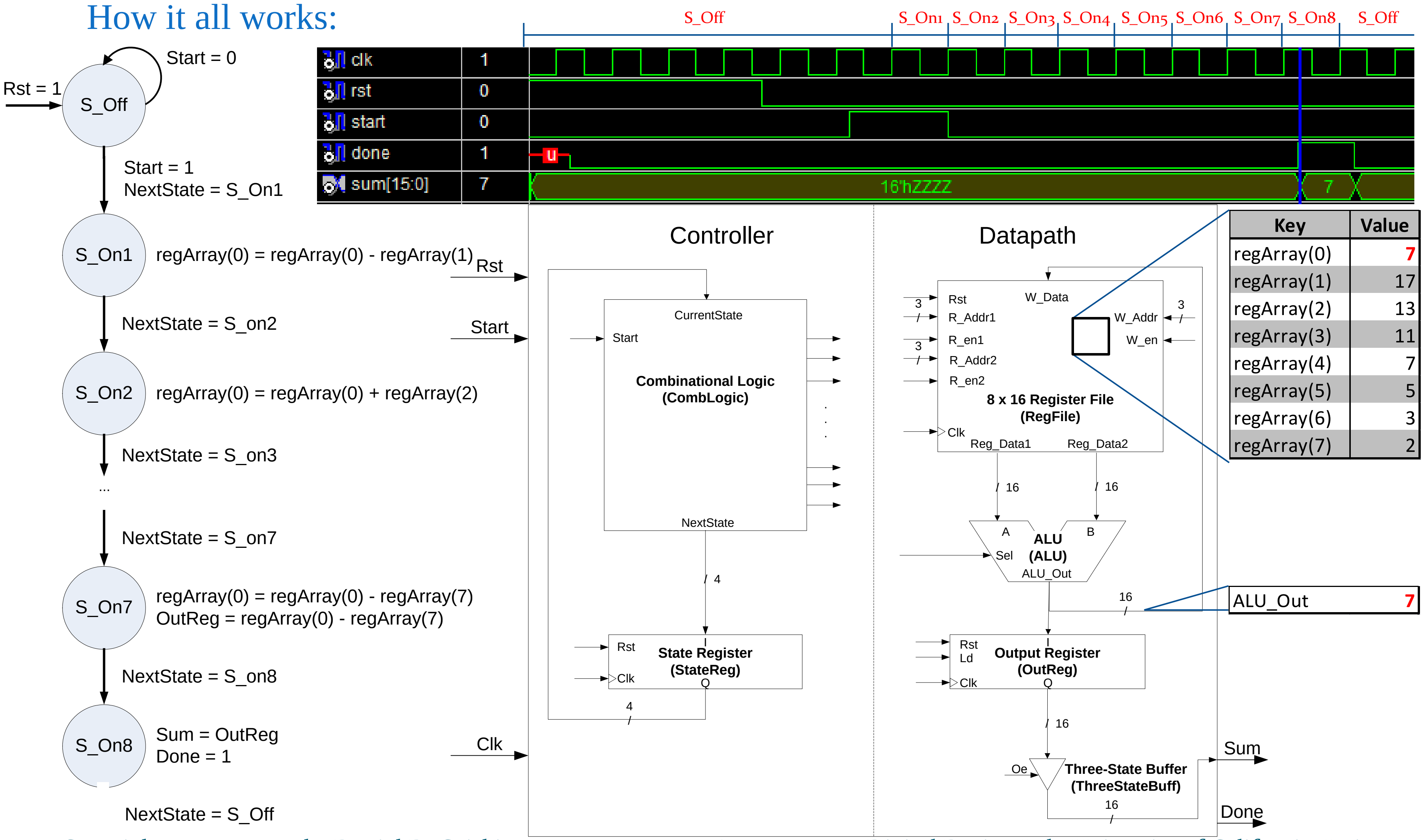
7th operation: previous result – regArray(7)

Finally: display output

This is the behavior.
Can convert this into an FSMD for structure!

1-5. Understanding Integrator Behavior (2)

How it all works:



Key	Value
regArray(0)	7
regArray(1)	17
regArray(2)	13
regArray(3)	11
regArray(4)	7
regArray(5)	5
regArray(6)	3
regArray(7)	2

ALU_Out	7

6-7. Describe Controller and Datapath

Clock-to-clock delay (minimum clock cycle):

Clk → '1',
StateReg (4 ns),
CombLogic → RegFile (5 ns – see next slide),
RegFile (6 ns),
ALU (8 ns),
and 1 ns setup time

Minimum Clock Cycle: 24 ns

Input-to-clock delay:

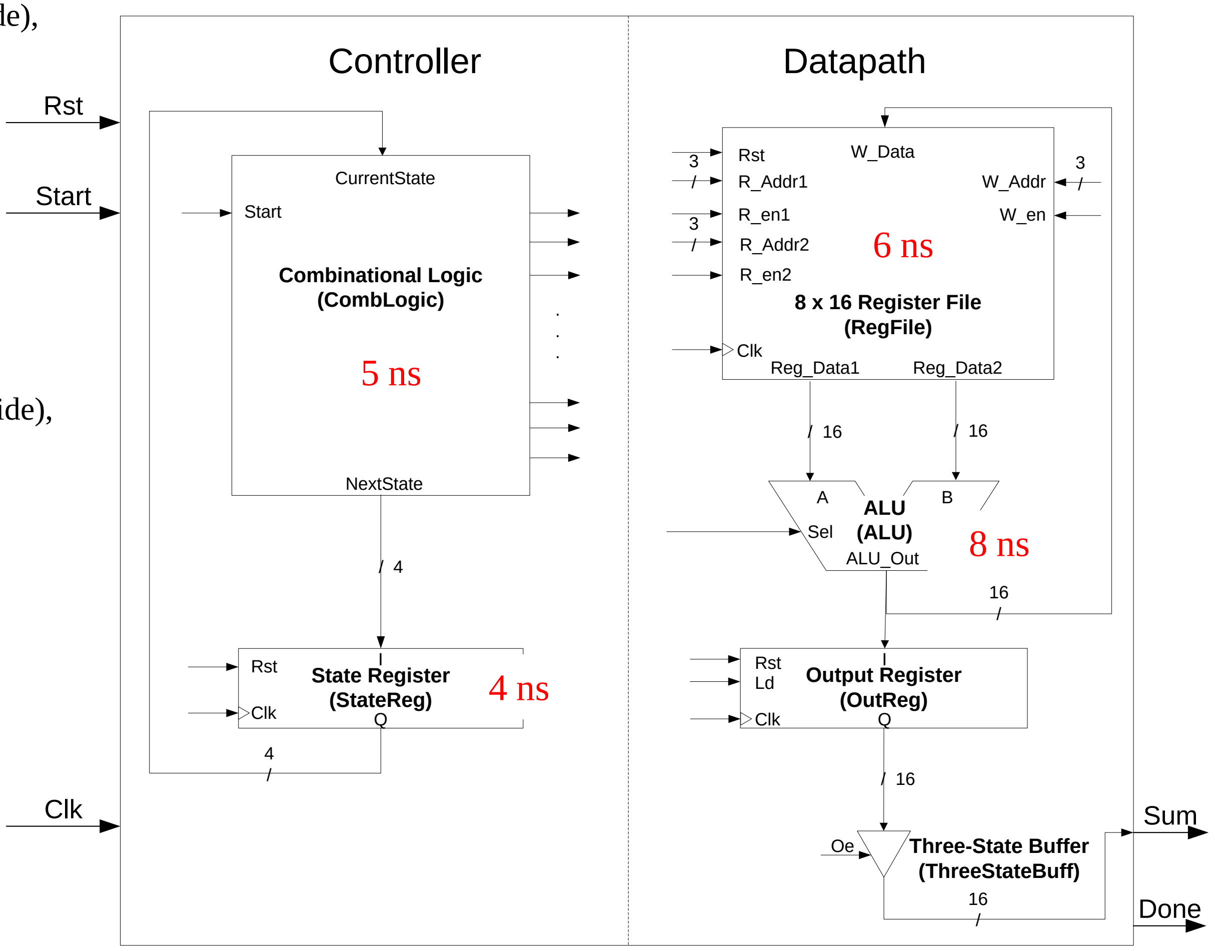
Start → '1',
CombLogic → StateReg (5 ns – see next slide),
and 1 ns setup time

Input-to-clock delay: 6 ns

Clock-to-output delay:

Clk → '1',
OutReg → ThreeStateBuff (4 ns),
ThreeStateBuff → Sum (1 ns)

Clock-to-output delay: 5 ns



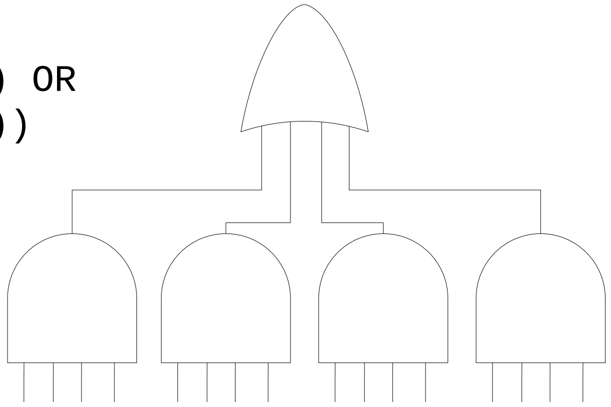
8-9. Control Words and Delays

Truth table for structural model:

Start	CurrState	NextState	R_en1	R_en2	W_en	R_Addr1	R_Addr2	W_Addr	Sel	Oe	Done
1	S_Off/0000	S_On1/0001	X	X	0	X	X	X	X	0	0
X	S_On1/0001	S_On2/0010	1	1	1	000	001	000	1	0	0
X	S_On2/0010	S_On3/0011	1	1	1	000	002	000	0	0	0
X	S_On3/0011	S_On4/0100	1	1	1	000	003	000	1	0	0
X	S_On4/0100	S_On5/0101	1	1	1	000	004	000	0	0	0
X	S_On5/0101	S_On6/0110	1	1	1	000	005	000	1	0	0
X	S_On6/0110	S_On7/0111	1	1	1	000	006	000	0	0	0
X	S_On7/0111	S_On8/1000	1	1	1	000	007	000	1	0	0
X	S_On8/1000	S_Off/0000	1	1	X	X	X	X	0	1	1

Calculating delay:

```
NextState(0) = ((Start AND (NOT CurrState(0))) OR
               (CurrState(2) AND (NOT CurrState(0))) OR
               (CurrState(1) AND (NOT CurrState(0))))
…
```



Using a 4-input, 4-wide AND-OR gate and NOT gates, delay for NextState(0) is 5 ns.

2-4, 10-12. VHDL Coding

Ready for Xilinx!

Summary

- Lab 3 methodology
- Integrator example
    - Lab procedure walkthrough