

Lab4: Multiplexors VS. Shift Registers



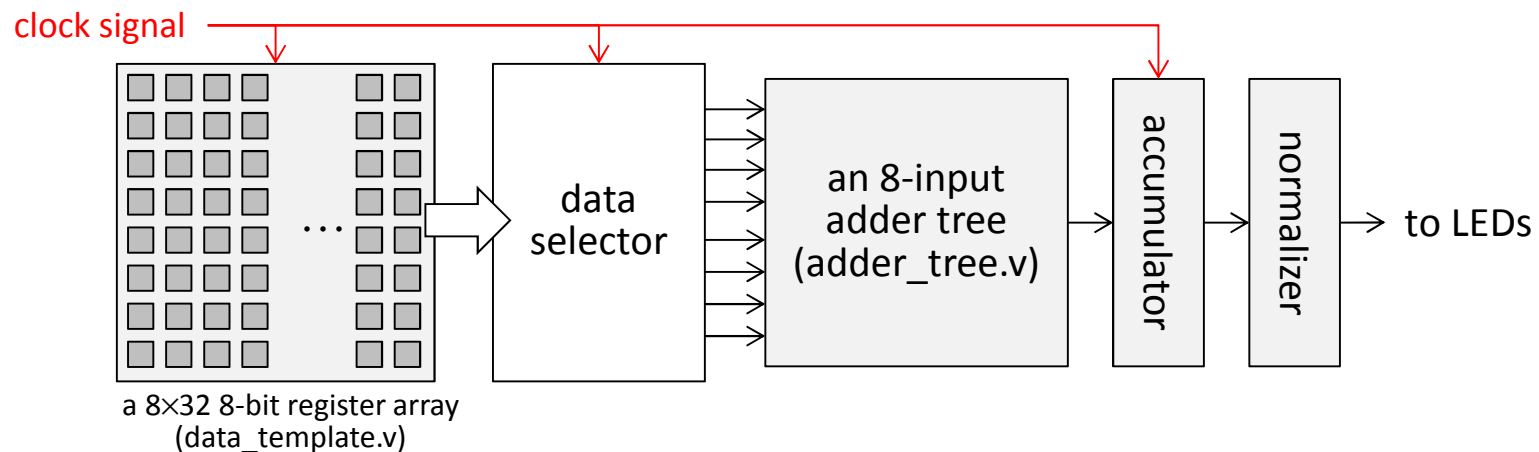
National Chiao Tung University
Chun-Jen Tsai
10/16/2015

Lab 4: Multiplexors v.s Shift Registers

- ❑ In this lab, you will design a circuit to reuse an adder-tree circuit to compute the average value of a 2D 8×32 8-bit integer array
 - The final average value, rounded to an 8-bit integer, will be displayed using the LEDs
 - You are **required** to use two different coding styles (to be specified later) to implement the circuit, and you must compare the two designs from cost and performance perspectives
- ❑ You will demo the design to your TA during the lab hours on 10/26

The System Behavior of Lab3

- ❑ The system architecture you will construct is as follows:



- At every clock cycle, the data selector selects one column of data registers and connect them to the adder tree
- The accumulator accumulates the output from the adders, and after 32 clock cycles, it has the total sum of 256 data registers
- The normalizer then compute the average for LED output

Provided Code Modules in Lab4

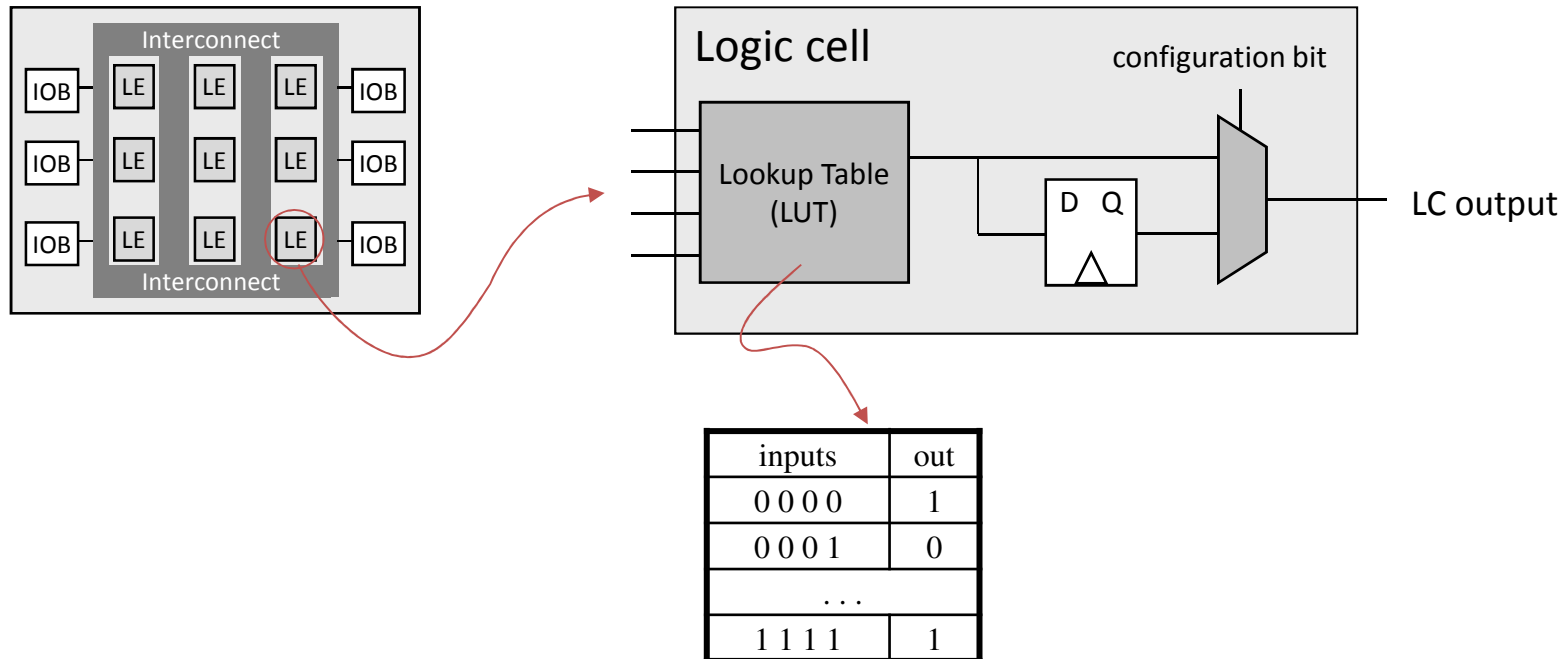
- ❑ In lab 4, you can download three Verilog files
 - `adder_tree.v` – the adder tree module that you can use to compute the sum of eight 8-bit signals
 - `data_template.v` – this is **not** a complete module! It merely shows you how the 2D data array is declared. Note that, although Verilog allows us to declare a 2D 8-bit register array, we still use eight 1D 8-bit register arrays to ensure portability
 - `data.dat` – Verilog code to initialize the 2D data array
- ❑ You must write Verilog code for the rest of the system using two different types of data selectors
 - Write one Verilog program for each type, not integrating two different types in one program
 - This is called “design exploration” in digital circuit design

FPGA Circuit Resources

- ❑ How does an FPGA implements a digital circuits
 - It does not simply use 2-input gates due to wiring and area efficiency cost issue
- ❑ An FPGA has different types of **basic** circuit units
 - Gates
 - Multiplexors
 - Look-up Tables (LUTs)
 - Flip-flops
- ❑ LUTs and flip-flops are the key logic elements for circuit implementation in an FPGA
 - An LUT is a small RAM with 4-bit or 6-bit (address) input and a single bit output

Flip-Flops and LUTs in FPGA

- ❑ Traditionally, a “logic cell” in an FPGA contains a 4-bit input, 1-bit output LUT and a flip-flop

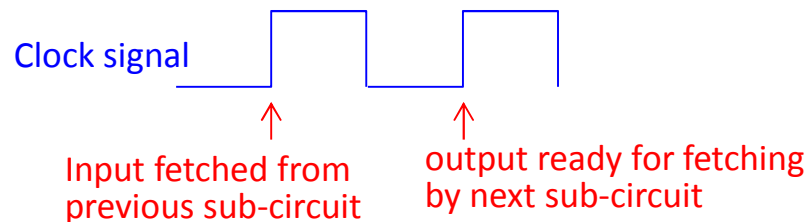


Logic Resource Usage

- ❑ Any digital circuit designs must find a good tradeoff between performance and logic usage
- ❑ It is important for you to understand the impact of different coding styles to
 - The amount of LUTs, flip-flops, BRAMs used
 - The length of critical path

Critical Path Length

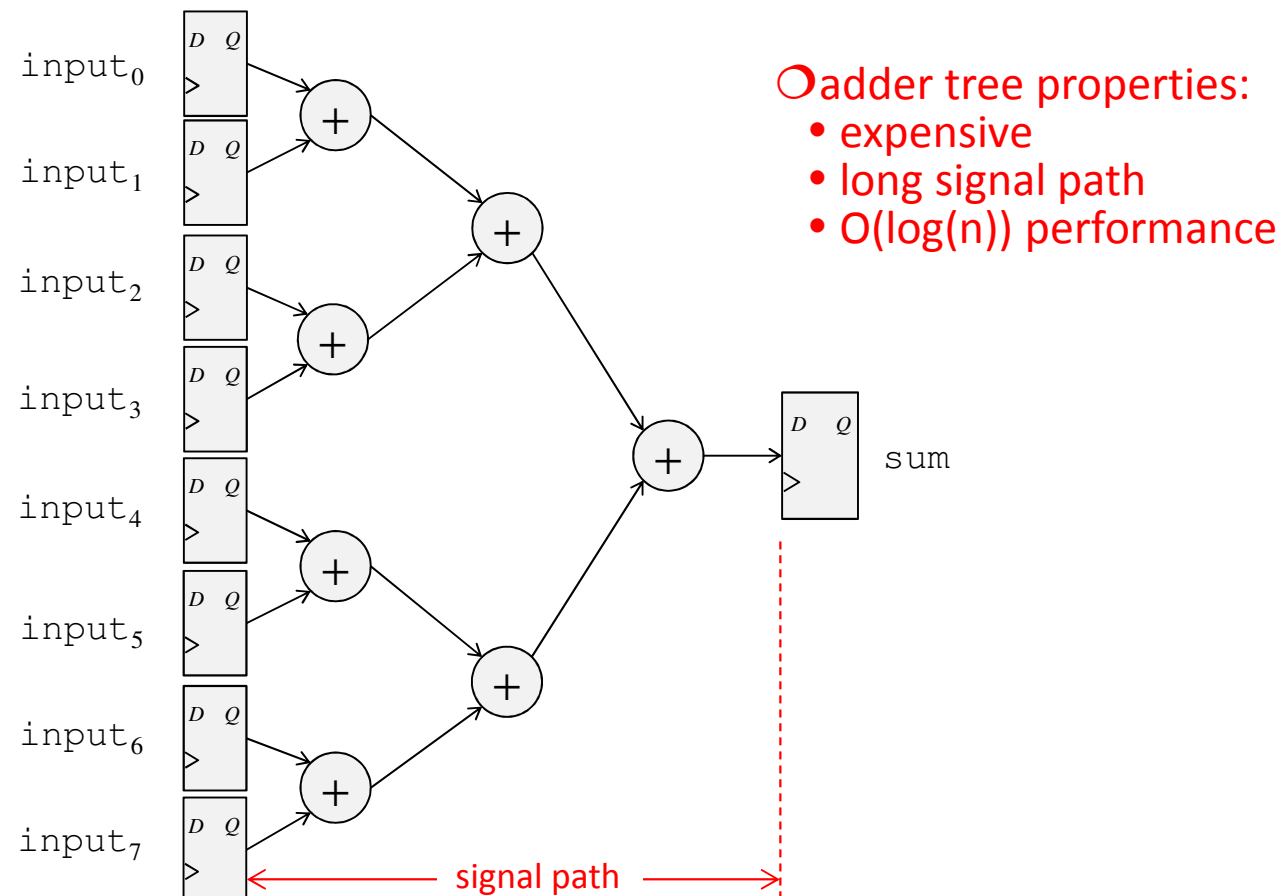
- ❑ In synchronous circuits, all combinational circuits take new input at a rising (or falling) edge of the clock signal and **must** produce a stable output before the next rising (or falling) edge of clock



- ❑ A design contains many sub-circuits, the one with the longest time to produce stable output is the critical path
- ❑ The length of the critical path determines the maximal working frequency of your logic

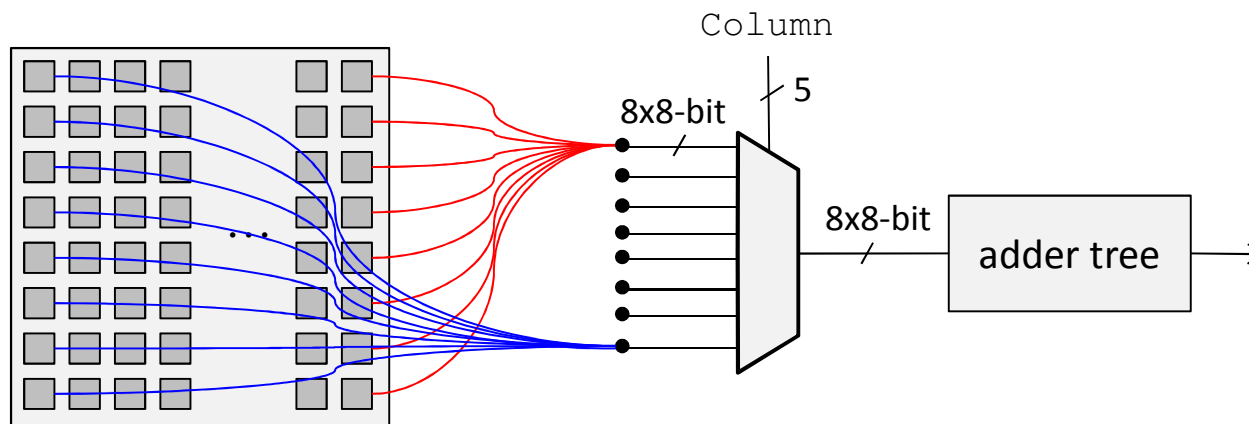
Adder Tree

- ❑ An adder tree is a combinational circuit to compute the sum of an array of numbers in a single clock cycle



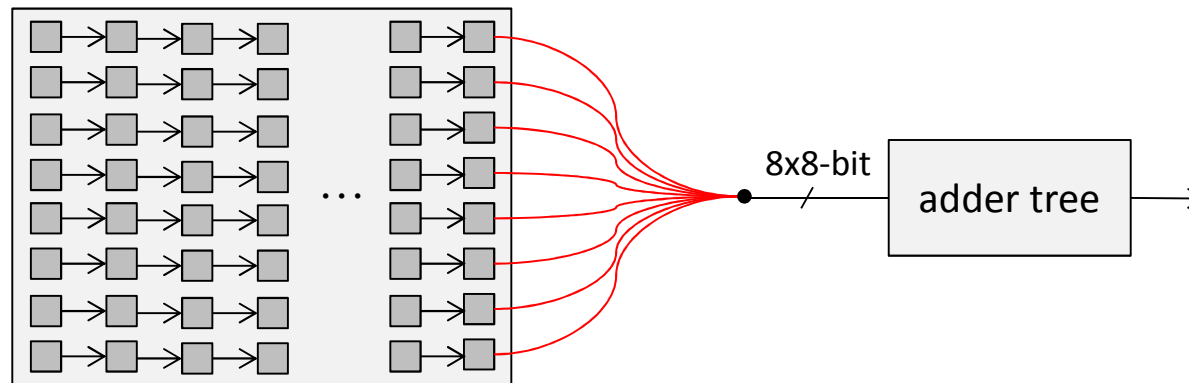
Mux-based Data Selector

- ❑ An intuitive idea in Verilog to select different columns of data is to use multiplexors (i.e., `if-then-else` or `case` statements):
 - Blue wires connects the first column of registers to the adders
 - Red wires connects the last column of registers to the adders
 - The “Column” signal selects different columns at different clock cycles



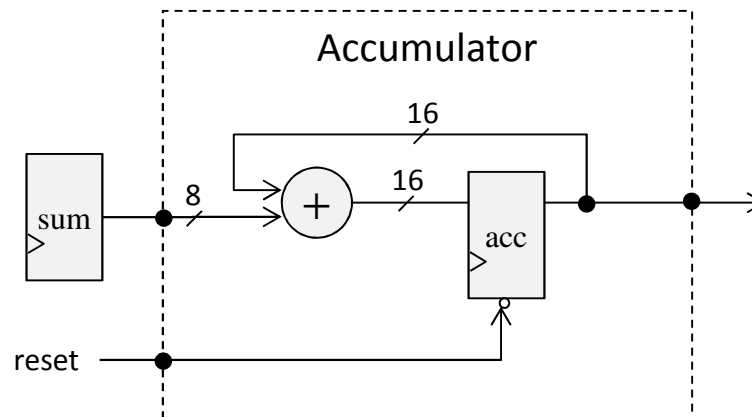
Shift Register-based Data Selector

- ❑ Another idea is to use shift-registers
 - The adder tree always connects to the last column of registers
 - The remaining columns will be shifted to the last column one-by-one at each clock cycles



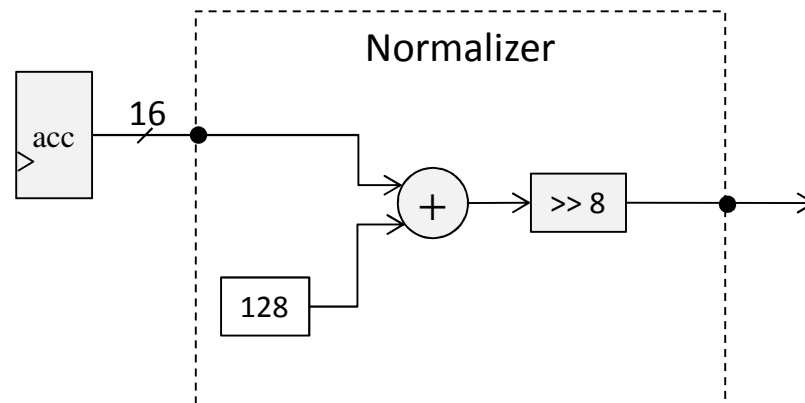
Accumulator

- ❑ The accumulator computes the total sum of a sequence of number in a sequential manner
 - One clock cycle per summation
 - The temporary result is stored in a 16-bit register because we have 256 8-bit values to add, so the maximum sum is 2^{16}



Normalizer

- ❑ We have 256 input data, to compute their average, we must divide the total sum by 256
 - A simple 8-bit shifter can do the job
 - In hardware, you don't actually “shift” the data bits, you just drop the LSBs!
- ❑ However, we want to round the average to the nearest integer, so the we must add 128 to the total sum first



How to Check Logic Usages

- ❑ Design summary in ISE shows logic usage:

ISE Project Navigator (P.20131013) - C:\Users\cjtsai\lab4\lab4.xise - [Design Summary (Synthesized)]

File Edit View Project Source Process Tools Window Layout Help

Design View: Implementation Simulation

Hierarchy

- lab4
 - xc3s700an-4fgg484
 - data_template (data_template.v)
 - add - adder_tree (adder_tree.v)

No Processes Running

Processes: data_template

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

Start Design Files Libraries

Design Summary (Synthesized) data_template.v

Console

Maximum output required time after clock: 5.558ns
Maximum combinational path delay: No path found

Process "Synthesize - XST" completed successfully

Errors Warnings Find in Files Results

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	416	5888	7%
Number of Slice Flip Flops	69	11776	0%
Number of 4 input LUTs	766	11776	6%
Number of bonded IOBs	34	372	9%
Number of GCLKs	1	24	4%

How to Check Maximum Frequency

- ❑ The console window shows the working frequency:

The screenshot shows the ISE Project Navigator interface. The Design Summary (Synthesized) window is open, displaying various design metrics. The Console window at the bottom is circled in red, showing the following timing information:

```
Speed Grade: -4  
  
Minimum period: 19.065ns (Maximum Frequency: 52.453MHz)  
Minimum input arrival time before clock: 2.992ns  
Maximum output required time after clock: 5.558ns  
Maximum combinational path delay: No path found
```

The Design Summary (Synthesized) window also displays the following information:

Target Device:	xc3s700an-4fgg484	Errors:	No Errors
Product Version:	ISE 14.7	Warnings:	2020 Warnings (2020 new)
Design Goal:	Balanced	Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	Timing Constraints:	
Environment:	System Settings	Final Timing Score:	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	416	5888	7%
Number of Slice Flip Flops	69	11776	0%
Number of 4 input LUTs	766	11776	6%
Number of bonded IOBs	34	372	9%
Number of GCLKs	1	24	4%

The Console window also displays the following information:

Detailed Reports	
Design Summary (Synthesized)	data_template.v

Notes on Demo

- ❑ For the demo of this lab, you must implement the averaging system in two different styles: mux-based and shift register-based data selector
- ❑ You must investigate the logic usage vs. performance tradeoff between your two coding styles and discuss them with your TA