
Logic and Computer Design Fundamentals

Chapter 7 – Registers and Register Transfers

Part 2 – Counters, Register Cells, Buses, & Serial Operations

Charles Kime & Thomas Kaminski

© 2004 Pearson Education, Inc.

[Terms of Use](#)

(Hyperlinks are active in View Show mode)

Overview

- **Part 1 - Registers, Microoperations and Implementations**
 - Registers and load enable
 - Register transfer operations
 - Microoperations - arithmetic, logic, and shift
 - Microoperations on a single register
 - Multiplexer-based transfers
 - Shift registers
- **Part 2 - Counters, register cells, buses, & serial operations**
 - Microoperations on single register (continued)
 - Counters
 - Register cell design
 - Multiplexer and bus-based transfers for multiple registers
 - Serial transfers and microoperations

Counters

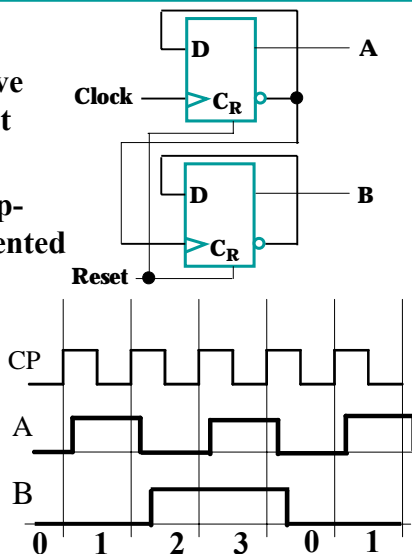
- **Counters** are sequential circuits which "count" through a specific state sequence. They can count up, count down, or count through other fixed sequences. Two distinct types are in common usage:
- **Ripple Counters**
 - Clock is connected to the flip-flop clock input on the LSB bit flip-flop
 - For all other bits, a flip-flop output is connected to the clock input, thus circuit is not truly synchronous
 - Output change is delayed more for each bit toward the MSB.
 - Resurgent because of low power consumption
- **Synchronous Counters**
 - Clock is directly connected to the flip-flop clock inputs
 - Logic is used to implement the desired state sequencing

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 3

Ripple Counter

- **How does it work?**
 - When there is a positive edge on the clock input of A, A complements
 - The clock input for flip-flop B is the complemented output of flip-flop A
 - When flip A changes from 1 to 0, there is a positive edge on the clock input of B causing B to complement



Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 4

Ripple Counter (continued)

- The arrows show the cause-effect relationship from the prior slide =>
- The corresponding sequence of states =>
 $(B,A) = (0,0), (0,1), (1,0), (1,1), (0,0), (0,1), \dots$
- Each additional bit, C, D, ... behaves like bit B, changing half as frequently as the bit before it.
- For 3 bits: $(C,B,A) = (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1), (0,0,0), \dots$

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 5

Ripple Counter (continued)

- These circuits are called *ripple counters* because each edge sensitive transition (positive in the example) causes a change in the next flip-flop's state.
- The changes “ripple” upward through the chain of flip-flops, i. e., each transition occurs after a clock-to-output delay from the stage before.
- To see this effect in detail look at the waveforms on the next slide.

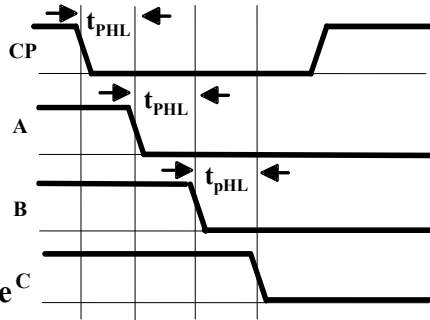
Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 6

Ripple Counter (continued)

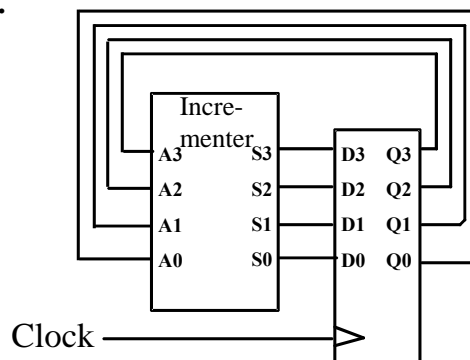
- Starting with $C = B = A = 1$, equivalent to $(C,B,A) = 7$ base 10, the next clock increments the count to $(C,B,A) = 0$ base 10. In fine timing detail:

- The clock to output delay t_{PHL} causes an increasing delay from clock edge for each stage transition.
- Thus, the count “ripples” from least to most significant bit.
- For n bits, total worst case delay is $n t_{PHL}$.



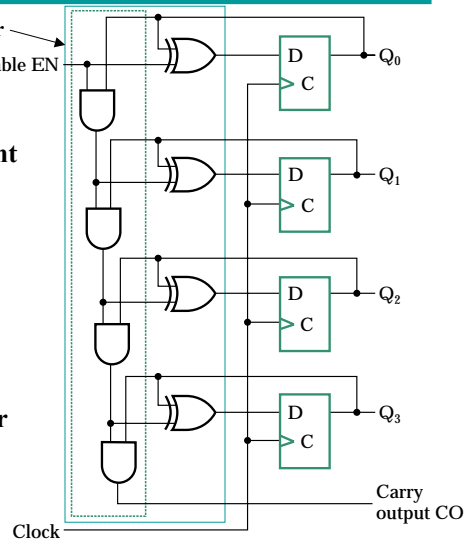
Synchronous Counters

- To eliminate the "ripple" effects, use a common clock for each flip-flop and a combinational circuit to generate the next state.
- For an up-counter, use an incrementer \Rightarrow



Synchronous Counters (continued)

- **Internal details => Incrementer**
- **Internal Logic**
 - XOR complements each bit
 - AND chain causes complement of a bit if all bits toward LSB from it equal 1
- **Count Enable**
 - Forces all outputs of AND chain to 0 to “hold” the state
- **Carry Out**
 - Added as part of incrementer
 - Connect to Count Enable of additional 4-bit counters to form larger counters

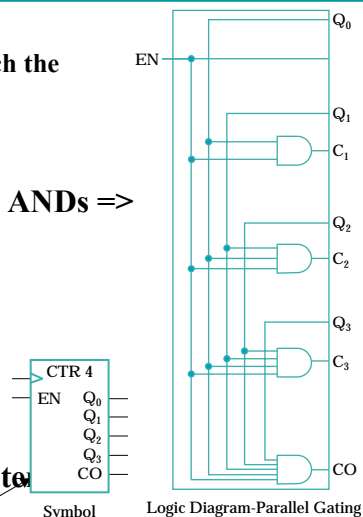


(a) Logic Diagram-Serial Gating

Chapter 7 - Part 2 9

Synchronous Counters (continued)

- **Carry chain**
 - series of AND gates through which the carry “ripples”
 - Yields long path delays
 - Called *serial gating*
- **Replace AND carry chain with ANDs => in parallel**
 - Reduces path delays
 - Called *parallel gating*
 - Like carry lookahead
 - Lookahead can be used on COs and ENs to prevent long paths in large counters
- **Symbol for Synchronous Counter**



Symbol

Logic Diagram-Parallel Gating

Other Counters

- See text for:
 - **Down Counter** - counts downward instead of upward
 - **Up-Down Counter** - counts up or down depending on value a control input such as Up/Down
 - **Parallel Load Counter** - Has parallel load of values available depending on control input such as Load
- **Divide-by- n (Modulo n) Counter**
 - Count is remainder of division by n which n may not be a power of 2 or
 - Count is arbitrary sequence of n states specifically designed state-by-state
 - Includes modulo 10 which is the **BCD counter**

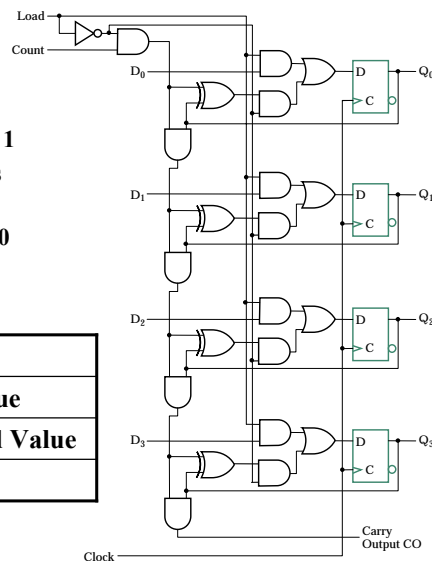
Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 11

Counter with Parallel Load

- Add path for input data
 - enabled for Load = 1
- Add logic to:
 - disable count logic for Load = 1
 - disable feedback from outputs for Load = 1
 - enable count logic for Load = 0 and Count = 1
- The resulting function table:

Load	Count	Action
0	0	Hold Stored Value
0	1	Count Up Stored Value
1	X	Load D



Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 12

Design Example: Synchronous BCD

- Use the sequential logic model to design a synchronous BCD counter with D flip-flops
- State Table =>
- Input combinations 1010 through 1111 are don't cares

Current State				Next State			
Q8	Q4	Q2	Q1	Q8	Q4	Q2	Q1
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0

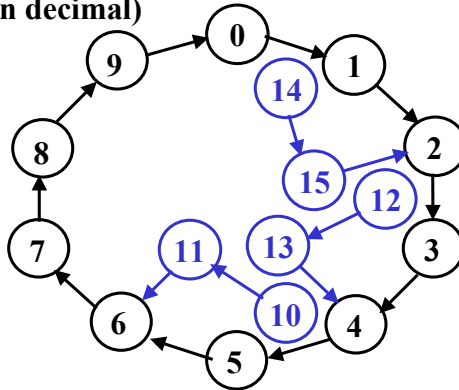
Synchronous BCD (continued)

- Use K-Maps to two-level optimize the next state equations and manipulate into forms containing XOR gates:
$$D1 = \overline{Q1}$$
$$D2 = Q2 \oplus Q1 \overline{Q8}$$
$$D4 = Q4 \oplus Q1 Q2$$
$$D8 = Q8 \oplus (Q1 Q8 + Q1 Q2 Q4)$$
- The logic diagram can be draw from these equations
 - An asynchronous or synchronous reset should be added
- What happens if the counter is perturbed by a power disturbance or other interference and it enters a state other than 0000 through 1001?

Synchronous BCD (continued)

- Find the actual values of the six next states for the don't care combinations from the equations
- Find the overall state diagram to assess behavior for the don't care states (states in decimal)

Present State	Next State
Q8 Q4 Q2 Q1	Q8 Q4 Q2 Q1
1 0 1 0	1 0 1 1
1 0 1 1	0 1 1 0
1 1 0 0	1 1 0 1
1 1 0 1	0 1 0 0
1 1 1 0	1 1 1 1
1 1 1 1	0 0 1 0



Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 15

Synchronous BCD (continued)

- For the BCD counter design, if an invalid state is entered, return to a valid state occurs within two clock cycles
- Is this adequate? If not:
 - Is a signal needed that indicates that an invalid state has been entered? What is the equation for such a signal?
 - Does the design need to be modified to return from an invalid state to a valid state in one clock cycle?
 - Does the design need to be modified to return from a invalid state to a specific state (such as 0)?
- The action to be taken depends on:
 - the application of the circuit
 - design group policy
- See pages 278 - 279 of the text.

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 16

Counting Modulo N

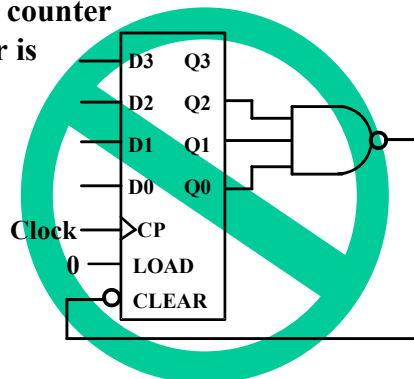
- The following techniques use an n -bit binary counter with asynchronous or synchronous clear and/or parallel load:
 - Detect a *terminal count* of N in a Modulo- N count sequence to asynchronously Clear the count to 0 or asynchronously Load in value 0 (These lead to counts which are present for only a very short time and can fail to work for some timing conditions!)
 - Detect a terminal count of $N - 1$ in a Modulo- N count sequence to Clear the count synchronously to 0
 - Detect a terminal count of $N - 1$ in a Modulo- N count sequence to synchronously Load in value 0
 - Detect a terminal count and use Load to preset a count of the terminal count value minus ($N - 1$)
- Alternatively, custom design a modulo N counter as done for BCD

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 17

Counting Modulo 7: Detect 7 and Asynchronously Clear

- A synchronous 4-bit binary counter with an asynchronous Clear is used to make a Modulo 7 counter.
- Use the Clear feature to detect the count 7 and clear the count to 0. This gives a count of 0, 1, 2, 3, 4, 5, 6, 7(short)0, 1, 2, 3, 4, 5, 6, 7(short)0, etc.
- **DON'T DO THIS!** Referred to as a “suicide” counter! (Count “7” is “killed,” but the designer’s job may be dead as well!)

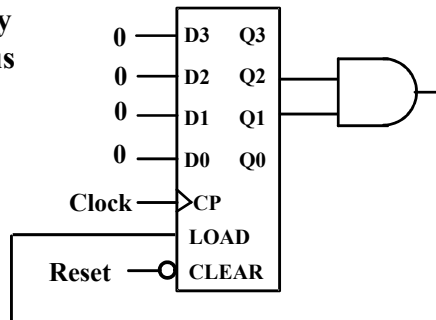


Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 18

Counting Modulo 7: Synchronously Load on Terminal Count of 6

- A synchronous 4-bit binary counter with a synchronous load and an asynchronous clear is used to make a Modulo 7 counter
- Use the Load feature to detect the count "6" and load in "zero". This gives a count of 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, ...
- Using don't cares for states above 0110, detection of 6 can be done with Load = $Q_4 Q_2$

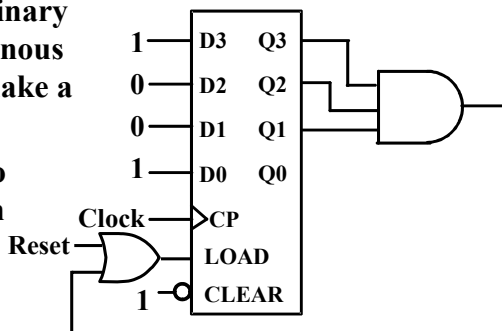


Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 19

Counting Modulo 6: Synchronously Preset 9 on Reset and Load 9 on Terminal Count 14

- A synchronous, 4-bit binary counter with a synchronous Load is to be used to make a Modulo 6 counter.
- Use the Load feature to preset the count to 9 on Reset and detection of count 14.
- This gives a count of 9, 10, 11, 12, 13, 14, 9, 10, 11, 12, 13, 14, 9, ...
- If the terminal count is 15 detection is usually built in as Carry Out (CO)



Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 20

Register Cell Design

- Assume that a register consists of identical cells
- Then register design can be approached as follows:
 - Design representative cell for the register
 - Connect copies of the cell together to form the register
 - Applying appropriate “boundary conditions” to cells that need to be different and contract if appropriate
- Register cell design is the first step of the above process

Register Cell Specifications

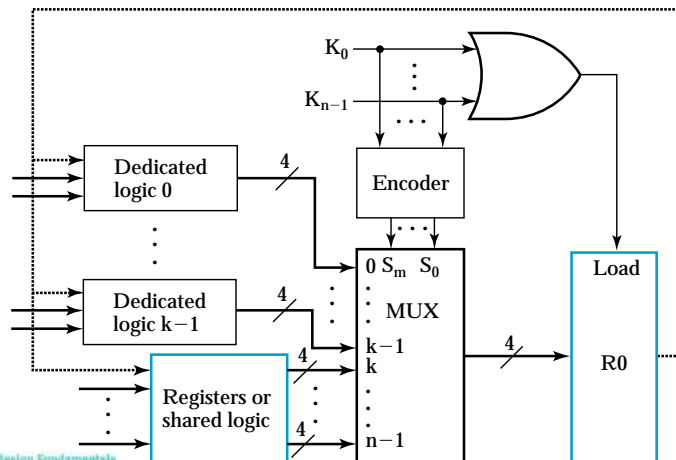
- A register
- Data inputs to the register
- Control input combinations to the register
 - Example 1: Not encoded
 - Control inputs: Load, Shift, Add
 - At most, one of Load, Shift, Add is 1 for any clock cycle
(0,0,0), (1,0,0), (0,1,0), (0,0,1)
 - Example 2: Encoded
 - Control inputs: S1, S0
 - All possible binary combinations on S1, S0
(0,0), (0,1), (1,0), (1,1)

Register Cell Specifications

- A set of register functions (typically specified as register transfers)
 - Example:
Load: $A \leftarrow B$
Shift: $A \leftarrow \text{sr } B$
Add: $A \leftarrow A + B$
- A hold state specification
 - Example:
 - Control inputs: Load, Shift, Add
 - If all control inputs are 0, hold the current register state

Multiplexer Approach

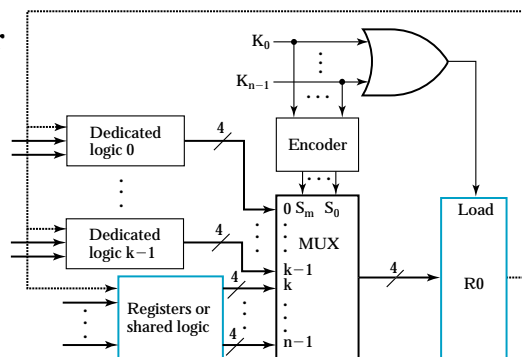
- Uses an n-input multiplexer with a variety of transfer sources and functions



Multiplexer Approach

- Load enable by OR of control signals K_0, K_1, \dots, K_{n-1}
 - assumes no load for 00...0
- Use:
 - Encoder + Multiplexer (shown) or
 - $n \times 2$ AND-OR

to select sources and/or transfer functions



Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 25

Example 1: Register Cell Design

- Register A (m-bits) Specification:
 - Data input: B
 - Control inputs (CX, CY)
 - Control input combinations (0,0), (0,1) (1,0)
 - Register transfers:
 - CX: $A \leftarrow B \vee A$
 - CY: $A \leftarrow B \oplus A$
 - Hold state: (0,0)

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 26

Example 1: Register Cell Design (continued)

- **Load Control**
 $\text{Load} = \text{CX} + \text{CY}$
- **Since all control combinations appear as if encoded (0,0), (0,1), (1,0) can use multiplexer without encoder:**
 $\text{S1} = \text{CX}$
 $\text{S0} = \text{CY}$
 $\text{D0} = \text{A}_i$ **Hold A**
 $\text{D1} = \text{A}_i \leftarrow \text{B}_i \oplus \text{A}_i$ **CY = 1**
 $\text{D2} = \text{A}_i \leftarrow \text{B}_i \vee \text{A}_i$ **CX = 1**
- **Note that the decoder part of the 3-input multiplexer can be shared between bits if desired**

Sequential Circuit Design Approach

- **Find a state diagram or state table**
 - Note that there are only two states with the state assignment equal to the register cell output value
- **Use the design procedure in Chapter 6 to complete the cell design**
- **For optimization:**
 - Use K-maps for up to 4 to 6 variables
 - Otherwise, use computer-aided or manual optimization

Example 1 Again

State Table:

	Hold	$A_i \vee B_i$		$A_i \oplus B_i$	
	$CX = 0$	$CX = 1$	$CX = 1$	$CY = 1$	$CY = 1$
A_i	$CY = 0$	$B_i = 0$	$B_i = 1$	$B_i = 0$	$B_i = 1$
0	0	0	1	0	1
1	1	1	1	1	0

- Four variables give a total of 16 state table entries

- By using:

- Combinations of variable names and values
- Don't care conditions (for $CX = CY = 1$)

only 8 entries are required to represent the 16 entries

Example 1 Again (continued)

- K-map - Use variable ordering CX, CY, A_i, B_i and assume a D flip-flop

D_i		A_i	
0	0	1	1
0	1	0	1
X	X	X	X
0	1	1	1
		B_i	
CX		CY	

Example 1 Again (continued)

- The resulting SOP equation:
$$D_i = CX B_i + CY \bar{A}_i B_i + A_i \bar{B}_i + \bar{C}Y A_i$$
- Using factoring and DeMorgan's law:
$$D_i = CX B_i + \bar{A}_i (CY B_i) + A_i (\bar{C}Y \bar{B}_i)$$
$$D_i = CX B_i + A_i \oplus (CY B_i)$$

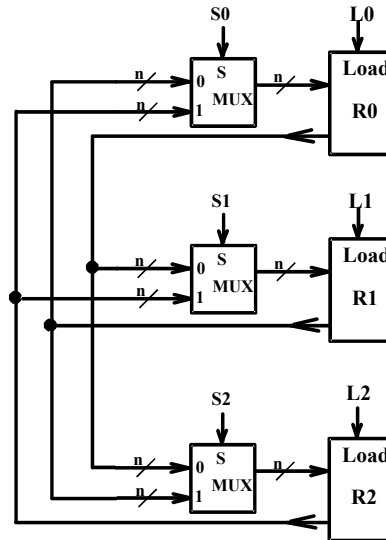
The gate input cost per cell = $2 + 8 + 2 + 2 = 14$
- The gate input cost per cell for the previous version is:
Per cell: 19
Shared decoder logic: 8
- Cost gain by sequential design > 5 per cell
- Also, no Enable on the flip-flop makes it cost less

Multiplexer and Bus-Based Transfers for Multiple Registers

- Multiplexer dedicated to each register
- Shared transfer paths for registers
 - A shared transfer object is called a *bus* (Plural: *buses*)
- Bus implementation using:
 - multiplexers
 - three-state nodes and drivers
- In most cases, the number of bits is the length of the receiving register

Dedicated MUX-Based Transfers

- Multiplexer connected to each register input produces a very flexible transfer structure =>
- Characterize the simultaneous transfers possible with this structure.

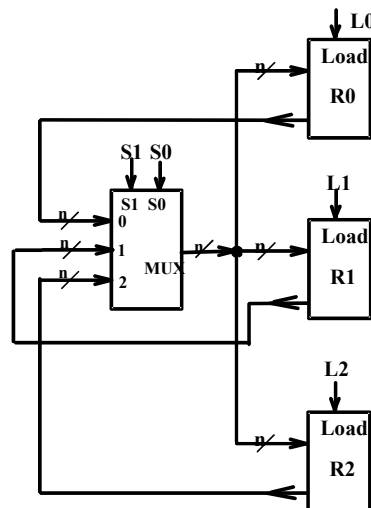


Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 33

Multiplexer Bus

- A single bus driven by a multiplexer lowers cost, but limits the available transfers =>
- Characterize the simultaneous transfers possible with this structure.
- Characterize the cost savings compared to dedicated multiplexers

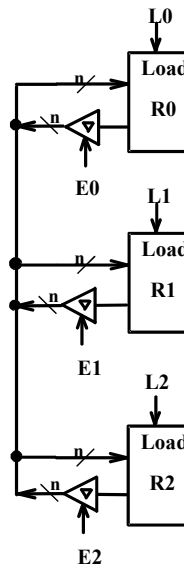


Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 34

Three-State Bus

- The 3-input MUX can be replaced by a 3-state node (bus) and 3-state buffers.
- Cost is further reduced, but transfers are limited
- Characterize the simultaneous transfers possible with this structure.
- Characterize the cost savings and compare
- Other advantages?



Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 35

Serial Transfers and Microoperations

- **Serial Transfers**
 - Used for “narrow” transfer paths
 - **Example 1: Telephone or cable line**
 - Parallel-to-Serial conversion at source
 - Serial-to-Parallel conversion at destination
 - **Example 2: Initialization and Capture of the contents of many flip-flops for test purposes**
 - Add shift function to all flip-flops and form large shift register
 - Use shifting for simultaneous Initialization and Capture operations
- **Serial microoperations**
 - **Example 1: Addition**
 - **Example 2: Error-Correction for CDs**

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 36

Serial Microoperations

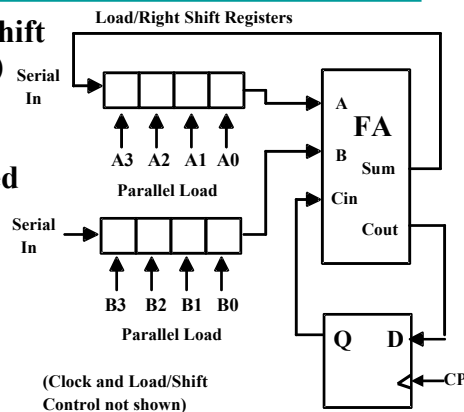
- By using two shift registers for operands, a full adder, and a flip flop (for the carry), we can add two numbers serially, starting at the least significant bit.
- Serial addition is a low cost way to add large numbers of operands, since a “tree” of full adder cells can be made to any depth, and each new level doubles the number of operands.
- Other operations can be performed serially as well, such as parity generation/checking or more complex error-check codes.
- Shifting a binary number left is equivalent to multiplying by 2.
- Shifting a binary number right is equivalent to dividing by 2.

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 37

Serial Adder

- The circuit shown uses two shift registers for operands A(3:0) and B(3:0).
- A full adder, and one more flip flop (for the carry) is used to compute the sum.
- The result is stored in the A register and the final carry in the flip-flop
- With the operands and the result in shift registers, a tree of full adders can be used to add a large number of operands. Used as a common digital signal processing technique.



Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Chapter 7 - Part 2 38

Terms of Use

- © 2004 by Pearson Education, Inc. All rights reserved.
- The following terms of use apply in addition to the standard Pearson Education [Legal Notice](#).
- Permission is given to incorporate these materials into classroom presentations and handouts only to instructors adopting Logic and Computer Design Fundamentals as the course text.
- Permission is granted to the instructors adopting the book to post these materials on a protected website or protected ftp site in original or modified form. All other website or ftp postings, including those offering the materials for a fee, are prohibited.
- You may not remove or in any way alter this Terms of Use notice or any trademark, copyright, or other proprietary notice, including the copyright watermark on each slide.
- [Return to Title Page](#)