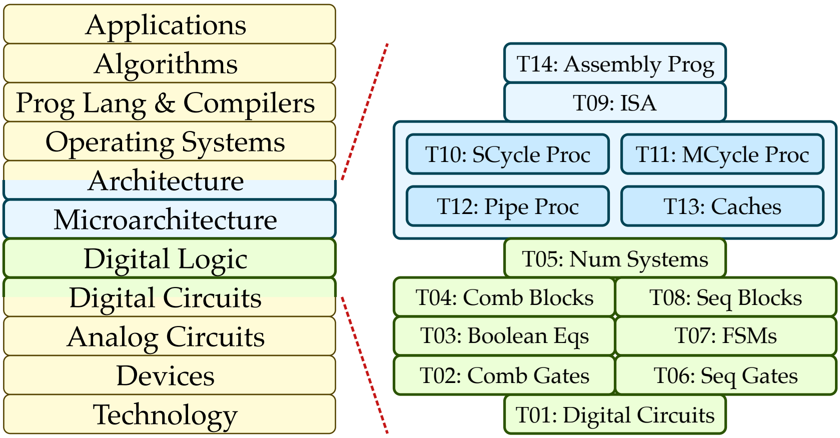
ECE 2300 Digital Logic and Computer Organization Fall 2024

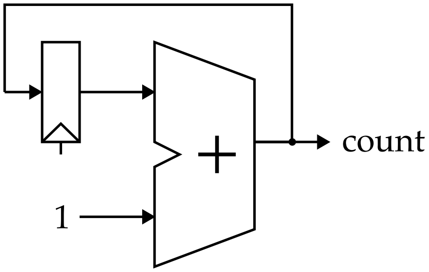
Topic 8: Sequential Building Blocks

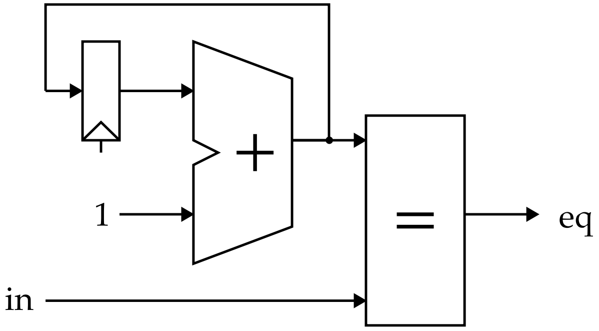
School of Electrical and Computer Engineering
Cornell University

revision: 2024-10-25-15-05


Applications
Algorithms
Prog Lang & Compilers
Operating Systems
Architecture
Microarchitecture
Digital Logic
Digital Circuits
Analog Circuits
Devices
Technology
T14: Assembly Prog
T09: ISA
T10: SCycle Proc
T11: MCycle Proc
T12: Pipe Proc
T13: Caches
T05: Num Systems
T04: Comb Blocks
T08: Seq Blocks
T03: Boolean Eqs
T07: FSMs
T02: Comb Gates
T06: Seq Gates
T01: Digital Circuits

1. Counters





Summay of Kinds of Counters

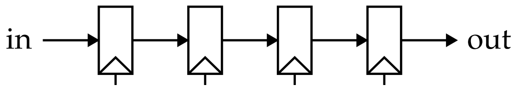
- Counters with enable and/or clear
- Up, down, up/down counters
- Variable increment counters (count by more than one)
- Variable start counters (control starting value)
- Decade counters (counts to ten then wraps around)
- Saturating counters (does not wrap around)

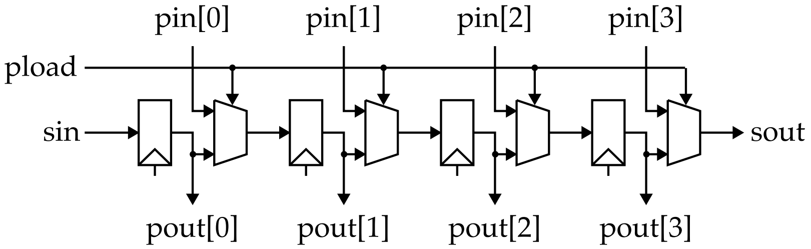
Implement an increment-by-one clearable up/down counter.

- clear: if 1 then clear counter, counter outputs zero following cycle
- updown: if 1 then count up by one, if 0 then count down by 1
- done: should be 1 if counter is at zero on current cycle
- counter should roll-over

Derive setup time constraint (max-delay constraint) as a function of the flip-flop timing parameters and the combinational block delays.

2. Shift Registers





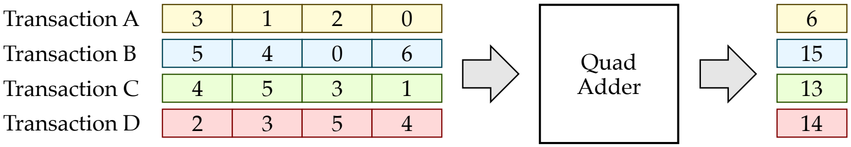
Summay of Kinds of Shift Registers

- Serial-in, serial-out shift registers
- Parallel-in, serial-out shift registers
- Serial-out, parallel-out shift registers
- Bi-directional shifters (can shift left and shift right)
- Variable shift registers (can vary shift amount)

Derive hold time constraint (min-delay constraint) as a function of the flip-flop timing parameters and the combinational block delays for the serial-in/out, parallel-in/out shift register.

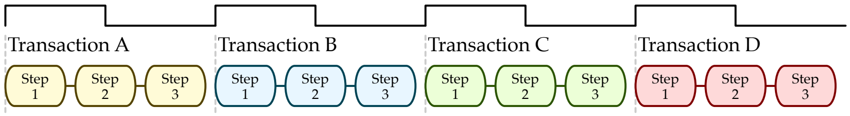
3. Sequential Arithmetic

- A quad adder adds four 4-bit input values to produce one 4-bit output value (ignore overflow)

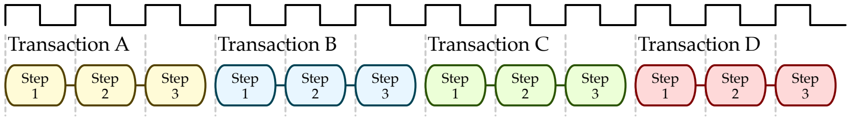


- A *transaction* involves processing a block of four 4-bit values
- The quad adder executes a *sequence* of transactions
- Each *transaction* can be broken down into three steps
  - Step 1. Add element 0 and element 1
  - Step 2. Add element 2 to the result of step 1
  - Step 3. Add element 3 to the result of step 2

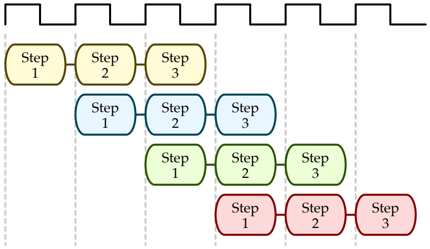
Single-Cycle Quad Adder



Multi-Cycle Quad Adder



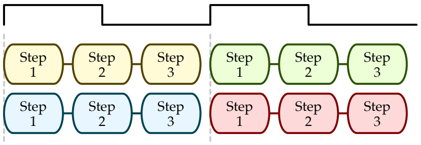
Pipelined Quad Adder



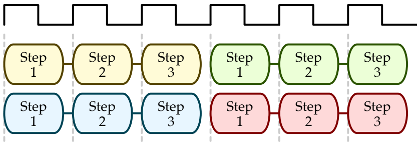
$$\frac{\text{Time}}{\text{Sequence}} = \frac{\text{Transactions}}{\text{Sequence}} \times \frac{\text{Avg Cycles}}{\text{Transaction}} \times \frac{\text{Time}}{\text{Cycle}}$$

	Area	$\frac{\text{Avg Cycles}}{\text{Trans}}$	$\frac{\text{Time}}{\text{Cycle}}$
Single-Cycle	medium	1	long
Multi-Cycle	small	3	short
Pipelined	large	≈ 1	short

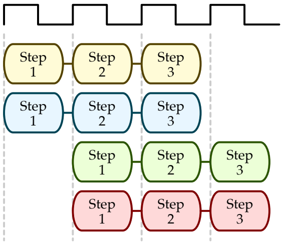
Parallel Single-Cycle Quad Adder



Parallel Multi-Cycle Quad Adder



Parallel Pipelined Quad Adder



$$\frac{\text{Time}}{\text{Sequence}} = \frac{\text{Transactions}}{\text{Sequence}} \times \frac{\text{Avg Cycles}}{\text{Transaction}} \times \frac{\text{Time}}{\text{Cycle}}$$

	Area	$\frac{\text{Avg Cycles}}{\text{Trans}}$	$\frac{\text{Time}}{\text{Cycle}}$
Parallel Single-Cycle	2× Single-Cycle	0.5	long
Parallel Multi-Cycle	2× Multi-Cycle	1.5	short
Parallel Pipelined	2× Pipelined	≈0.5	short

- Our goal is to do a quantitative comparative analysis of area and performance across six different implementations

$$\frac{\text{Time}}{\text{Sequence}} = \frac{\text{Transactions}}{\text{Sequence}} \times \frac{\text{Avg Cycles}}{\text{Transaction}} \times \frac{\text{Time}}{\text{Cycle}}$$

	Area	$\frac{\text{Trans}}{\text{Sequence}}$	$\frac{\text{Avg Cycles}}{\text{Trans}}$	$\frac{\text{Time}}{\text{Cycle}}$	$\frac{\text{Time}}{\text{Sequence}}$
Single-Cycle		100			
Multi-Cycle		100			
Pipelined		100			
Parallel SCycle		100			
Parallel MCycle		100			
Parallel Pipe		100			

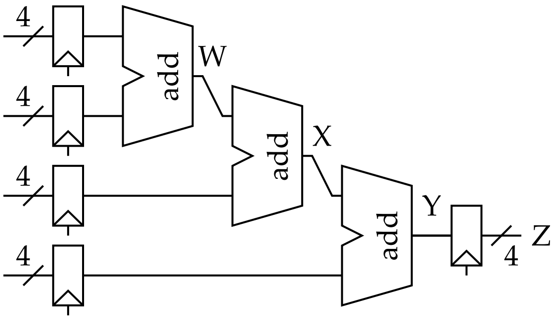
Constant Delay Model

	t_{pd}
4-bit 2-to-1 Mux	8τ
4-bit 3-to-1 Mux	10τ
4-bit Adder	40τ
4-bit Reg Clk-to-Q	9τ
4-bit Reg Setup	10τ

Area Model

	Area
4-bit 2-to-1 Mux	4α
4-bit 3-to-1 Mux	8α
4-bit Adder	12α
4-bit Reg	11α

3.1. Single-Cycle Quad Adder



Area

Time/Cycle

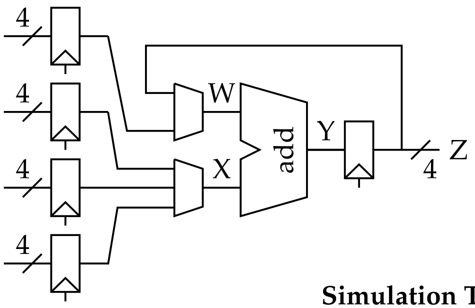
Simulation Table

Cycle	Input	After Reg	W	X	Y	Z
0	3, 1, 2, 0					
1	5, 4, 0, 6					
2	4, 5, 3, 1					
3	2, 3, 5, 4					
4						
5						

Transaction Diagram

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Transaction A														
Transaction B														
Transaction C														
Transaction D														

3.2. Multi-Cycle Quad Adder



Area

Time/Cycle

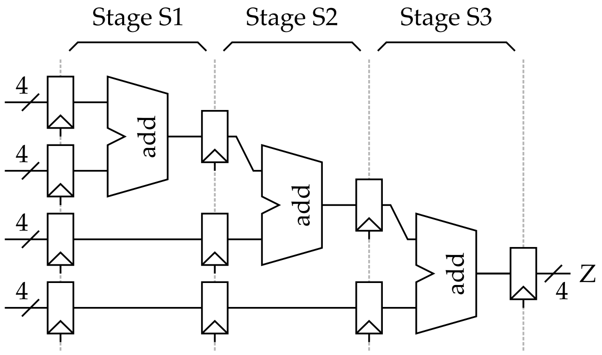
Simulation Table

Cycle	Input	After Reg	W	X	Y	Z
0	3, 1, 2, 0					
1						
2						
3	5, 4, 0, 6					
4						
5						
6						

Transaction Diagram

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Transaction A														
Transaction B														
Transaction C														
Transaction D														

3.3. Pipelined Quad Adder



Area

Time/Cycle

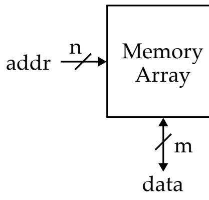
Simulation Table

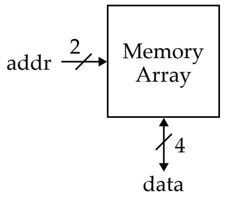
Cycle	Input	Stage S1	Stage S2	Stage S3	Z
0	3, 1, 2, 0				
1	5, 4, 0, 6				
2	4, 5, 3, 1				
3	2, 3, 5, 4				
4					

Transaction Diagram

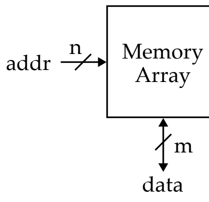
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Transaction A														
Transaction B														
Transaction C														
Transaction D														

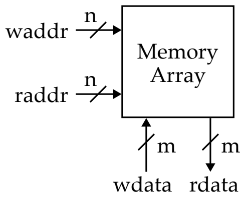
4. Memory Arrays

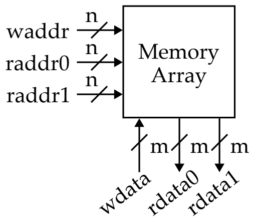


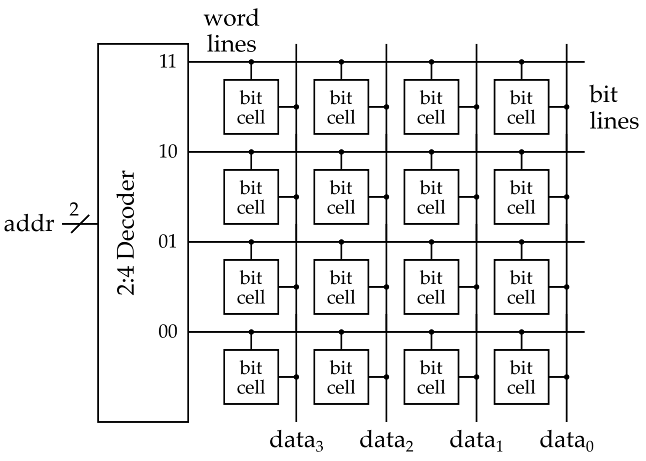


addr	data			
11	0	1	1	0
10	0	0	1	1
01	1	1	0	0
00	1	0	0	0

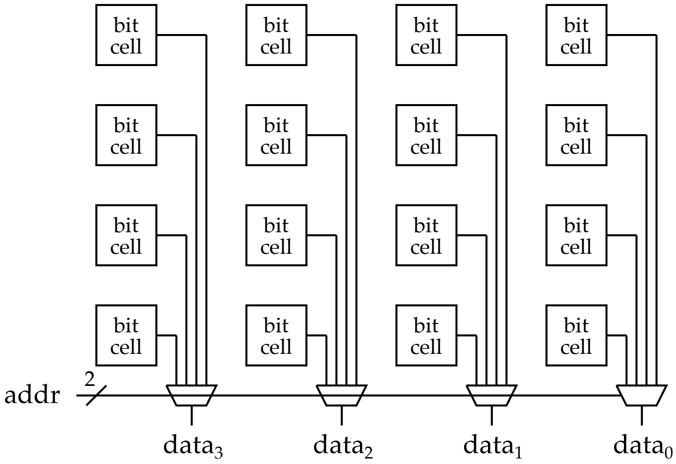




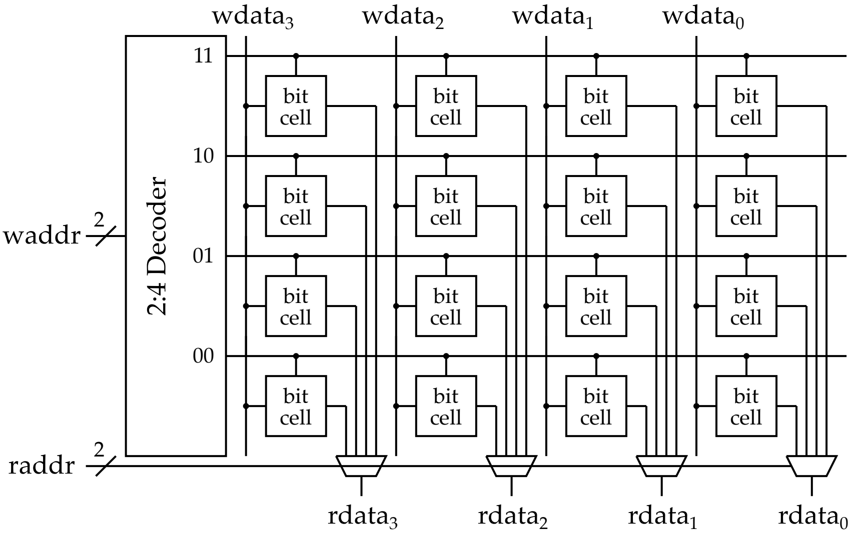




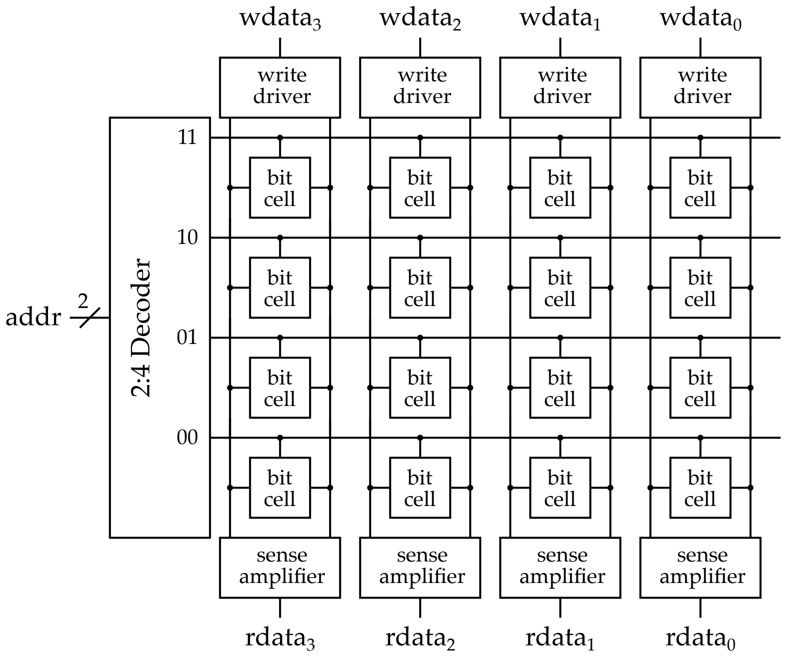
4.1. Read-Only Memories



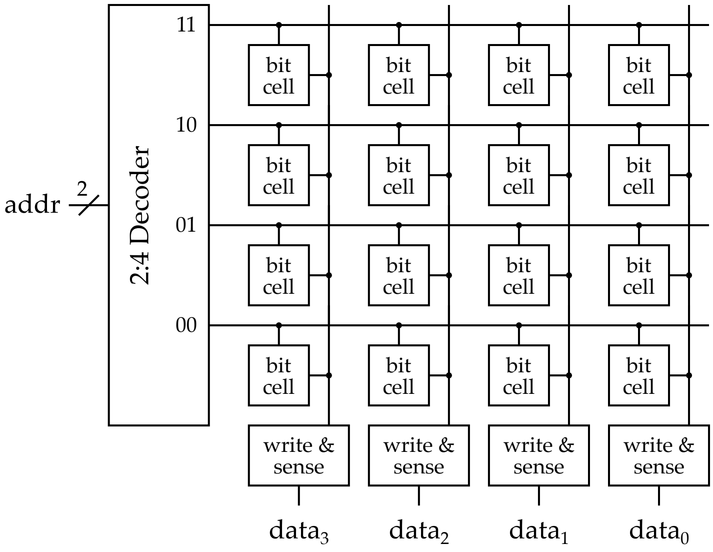
4.2. Register Files



4.3. Static Random Access Memories (SRAM)



4.4. Dynamic Random Access Memories (DRAM)



4.5. Comparing Memory Arrays

Memory Type	Transistors per Bit Cell	Density	Latency
Register File			
SRAM			
DRAM			

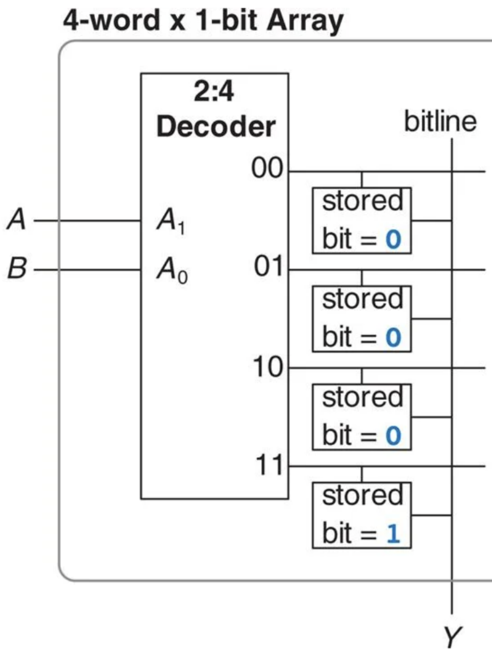
5. Programmable Logic

- Programmable logic uses sequential logic-gates to enable reconfiguring ("programming") the behavior of hardware after a chip has been fabricated

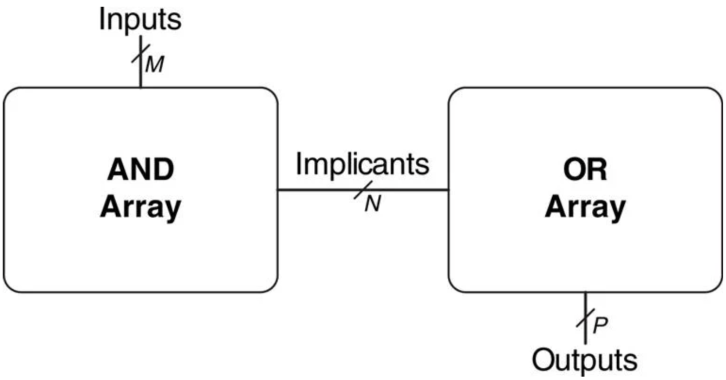
5.1. Programmable Truth Tables

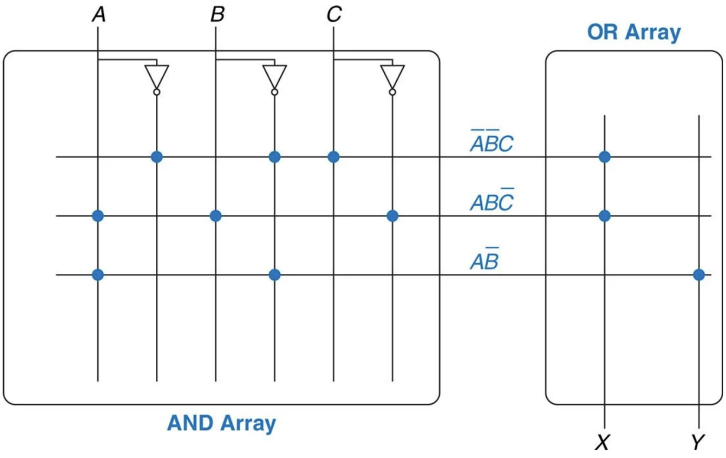
Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

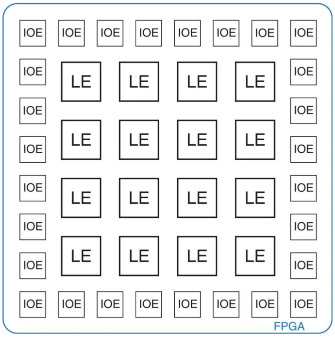


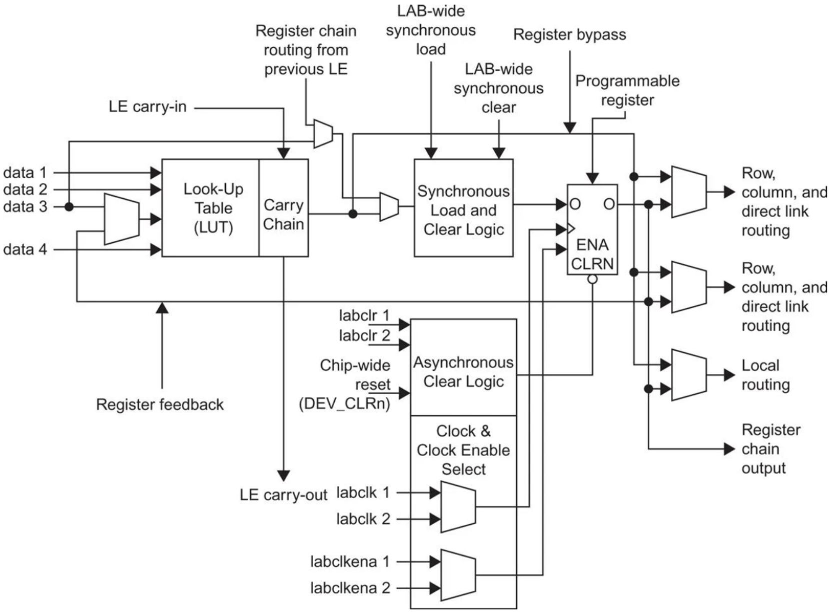
5.2. Programmable Logic Arrays





5.3. Field Programmable Gate Arrays




Register chain routing from previous LE
LAB-wide synchronous load
LAB-wide synchronous clear
Register bypass
Programmable register
LE carry-in
data 1
data 2
data 3
data 4
Look-Up Table (LUT)
Carry Chain
Synchronous Load and Clear Logic
O
O
ENA
CLRN
Row, column, and direct link routing
Row, column, and direct link routing
Local routing
Register chain output
labclr 1
labclr 2
Chip-wide reset (DEV_CLRn)
Asynchronous Clear Logic
Clock & Clock Enable Select
Register feedback
LE carry-out
labclk 1
labclk 2
labclkena 1
labclkena 2