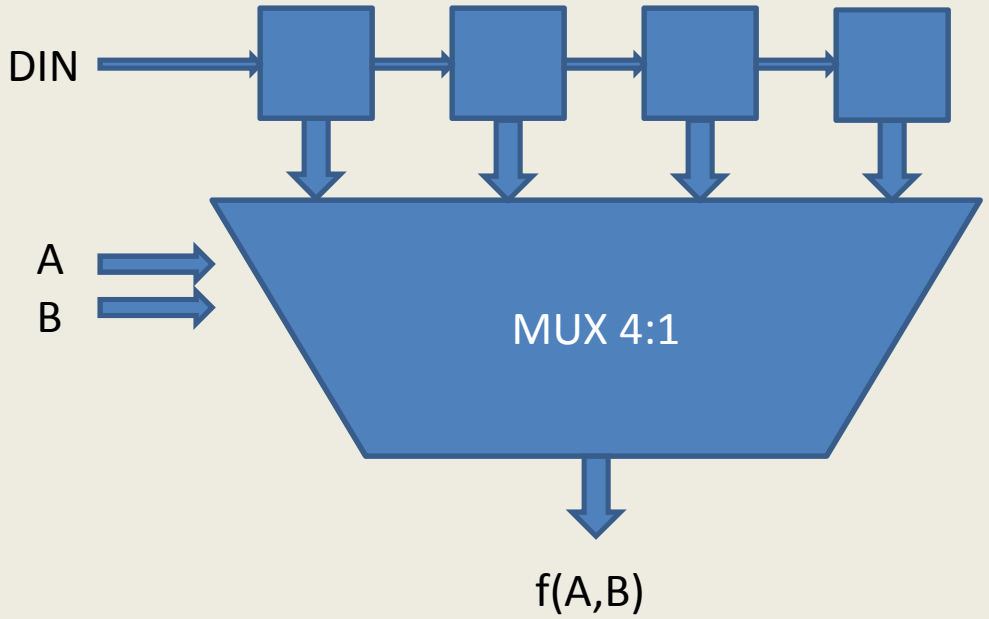
# CLB Shift Registers



# CLB Shift Registers

- Functionality present in SLICEM only
- Shift register length is $2^N$ bits per LUT, where N is the number of LUT inputs
- Output only through LUT
- Output Tap given by LUT inputs

# HDL Guide to CLB Shift Registers

- No Reset
- Can have clock enable
- Synthesis tool will automatically use CLB shift registers whenever possible (this behavior can be specified in synthesis properties)
- Synthesis tool will most often insert a Slice Flip-Flop as the last shifter stage, for better timing

# HDL Guide to CLB Shift Registers

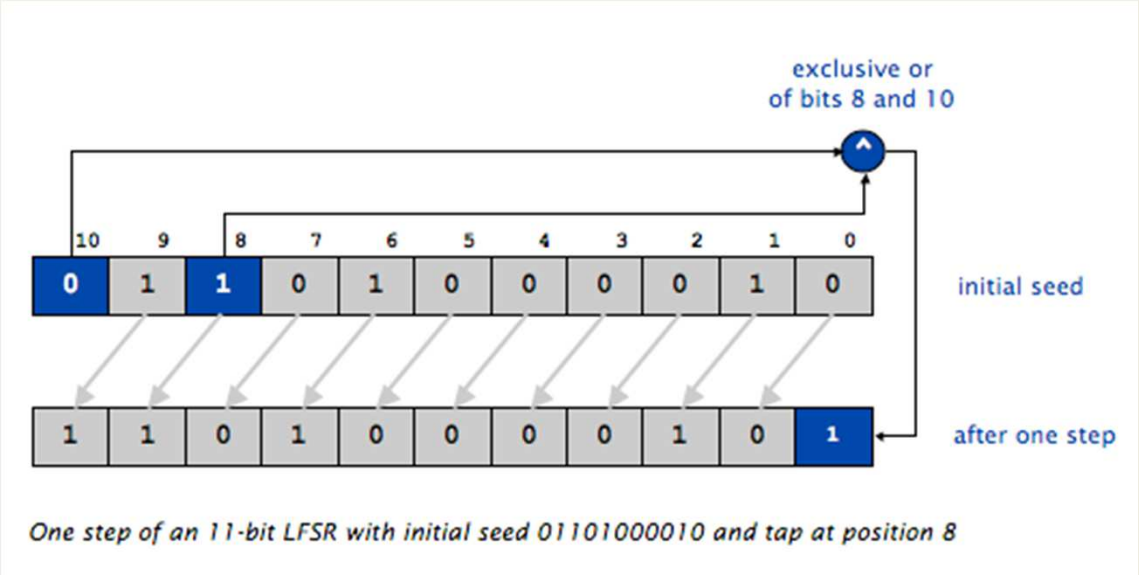
- Complex functionality: Variable-Tap Shift Register (Dynamic Shift Register)
- Output is selected based on an input address
- Synthesis tool will most often NOT insert an output Slice Flip-Flop -> watch your timing results
- Very useful for delay-line and synchronous FIFO applications

# FFs vs CLB Shift Registers

- We want to use CLB shift registers (SRLs=shift-register-LUTs) when:
  - Local delay line or shift register application (e.g. LFSR)
  - Synchronous Fifo
- We want to use FFs when:
  - Global data transport (to segment routing delays)
  - Fanout control – build a FF tree to reduce fanout

# FFs vs CLB Shift Registers

- LFSR example: bits 0-8 implemented as SRL, bits 9-10 as SRL
- Reduced shift register from 10FFs to 2 LUTs and 2 FFs




*One step of an 11-bit LFSR with initial seed 01101000010 and tap at position 8*

# FFs vs CLB Shift Registers

- Global data transport example
  - Transport data from one side of the FPGA to the other, 4ns delay
  - Implement 8-stage shift register to segment delay into 0.5ns
  - Synthesis tool will insert an 8-stage SRL and long routes (~2ns) between signal source, SRL and destination

# FFs vs CLB Shift Registers

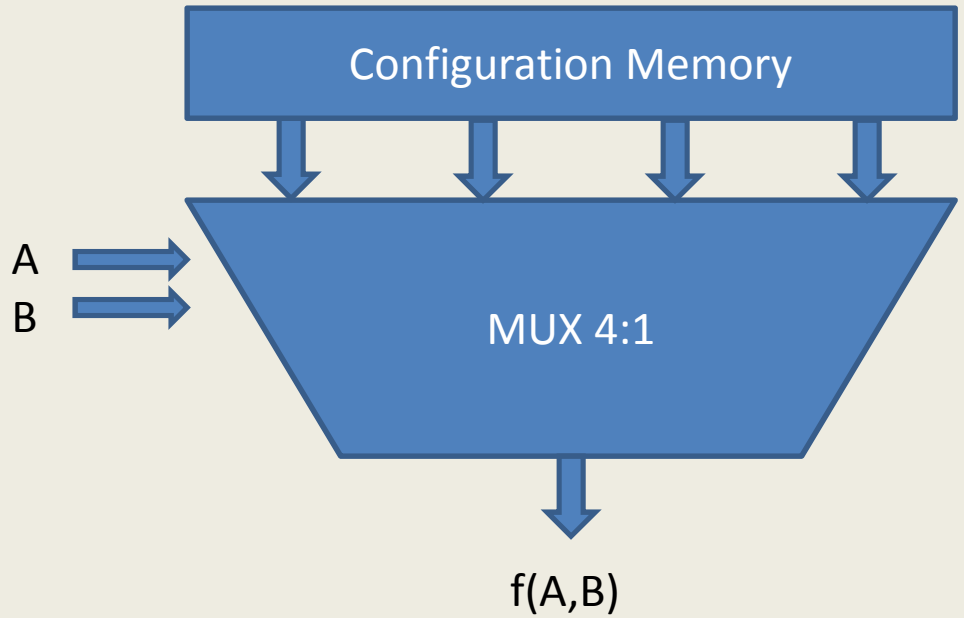
- Fanout control example
  - Signal has fanout of 128, destinations spread across the FPGA
  - Implement 7-deep logarithmic tree to reduce fanout
  - Synthesis tool will insert an 7-stage SRL and will fanout the SRL output to 128 destinations

# FFs vs CLB Shift Registers

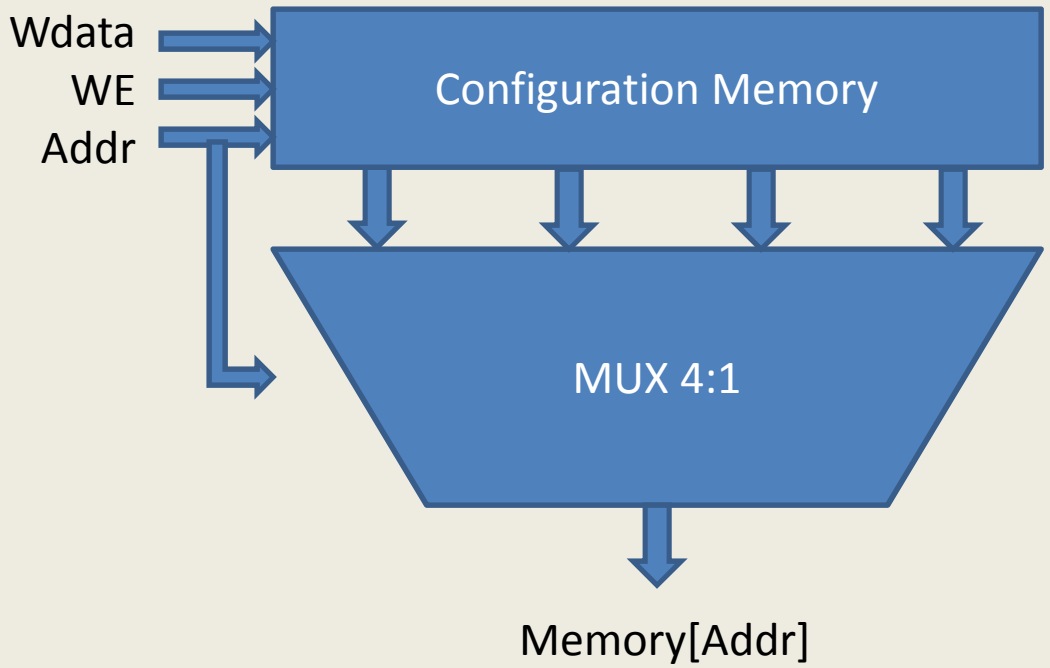
- To prevent SRL insertion, use the KEEP attribute:

```
(* KEEP = "TRUE" *) reg [WIDTH-1:0] shift[LENGTH-1:0];
integer i;

always @(posedge clock) begin
        shift[0] <= in;
        for(i=1;i<LENGTH;i=i+1)
                shift[i] <= shift[i-1];
end

assign out = shift[LENGTH-1];
```

# CLB Distributed Memory



# CLB Distributed Memory



# CLB Distributed Memory

- Some FPGA LUTs have user write access to their configuration memory -> the LUT becomes a memory read port
- A fraction of FPGA Slices (25%-50%) are SLICEM, which contain LUTs with this capability
- Memory capacity is $2^N$ x1bit per LUT, where N is the number of LUT inputs
- Using multiple LUTs and Slice multiplexers, various port configurations can be generated

# CLB Distributed Memory

- Example: Virtex-6 Architecture CLB Ram Primitives
  - Single-Port 32 x 1-bit RAM = 1LUT
  - Dual-Port 32 x 1-bit RAM = 1LUT
  - Quad-Port 32 x 2-bit RAM = 4LUTs
  - Simple Dual-Port 32 x 6-bit RAM = 4LUTs
  - Single-Port 64 x 1-bit RAM = 1LUT
  - Dual-Port 64 x 1-bit RAM = 2LUTs
  - Quad-Port 64 x 1-bit RAM = 4LUTs
  - Simple Dual-Port 64 x 3-bit RAM = 4LUTs
  - Single-Port 128 x 1-bit RAM = 2LUTs
  - Dual-Port 128 x 1-bit RAM = 4LUTs
  - Single-Port 256 x 1-bit RAM = 4LUTs

# HDL Guide to CLB Memory

- The following attributes describe CLB Memories, and must be present in your HDL code:
  - Synchronous write
  - Asynchronous read
  - 1 port is read-write
  - Remaining ports are read-only
  - Synchronous read can be implemented through the use of Slice Flip-Flops

# HDL Guide to CLB Memory

- Example: Dual-port 128x1 RAM in Virtex-4
  - Implemented using 8 16x1D primitives
  - Write enables for each 16x1D primitive is computed using the global write enable and the global address (1 extra LUT per 16x1D primitive)
  - 8:1 output multiplexer per-port, implemented in extra Slices

# Further Reading

- Virtex-4 FPGA User Guide, Chapter 5
- Virtex-6 FPGA Configurable Logic Block User Guide
- Spartan-3 FPGA User Guide, Chapter 5
- Multiplexer Selection, Xilinx White Paper
- Get Smart About Reset: Think Local, not Global
- Get your Priorities Right: Make your Design Up to 50% Smaller
- HDL Coding Practices to Accelerate Design Performance
- Using Look-Up Tables as Shift Registers in Spartan-3 Generation FPGAs