

Introduction

Dr. Paul D. Franzon

Outline

1. HDL-based Design Flow
2. Digital system timing

References

1. Smith/Franzon, Chapter 1

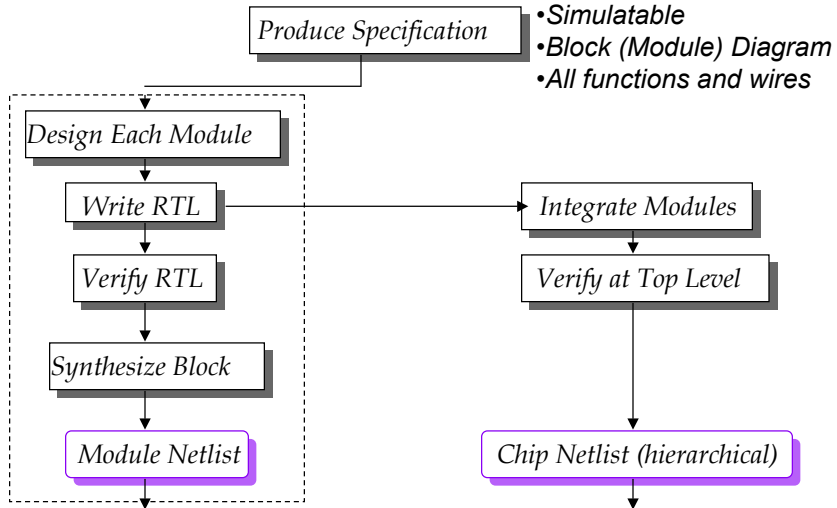
Role of Hardware Description Languages

Modern digital chip and system design centers on the use of Hardware Description Languages (HDL) to capture the design at the Register Transfer Level (RTL)

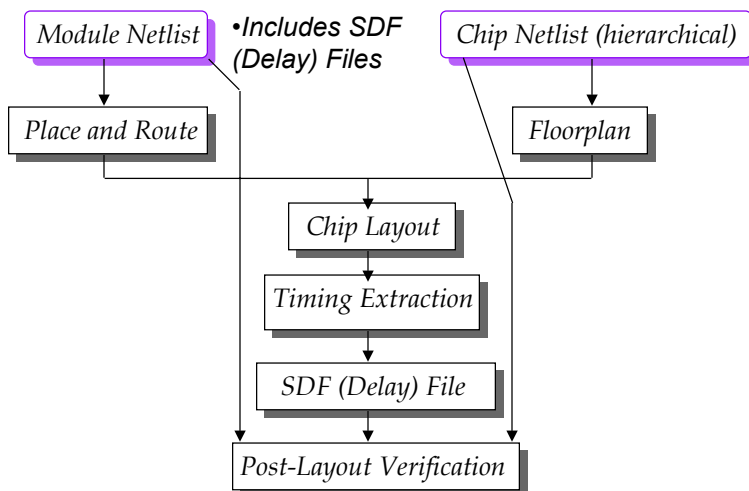
- RTL specifies all registers (flip-flops) and the combinational logic between the flip-flops
- Capturing the design in RTL is much faster than drawing a schematic
- Modern design depends heavily on the use of Computer-Aided Design tools:
 - ◆ To synthesize the RTL design into a schematic
 - ◆ To turn the schematic into a chip layout, FPGA mapping or board layout
 - ◆ To verify the original design, and verify that the more detailed designs are consistent with the original design
- Good designers depend critically on their ability to operate effectively with the CAD tools
 - ◆ Just knowing how to design logic is not enough
 - ◆ Unfortunately, you must learn a lot of tools and learn how to deal with their complexity and bugs
 - ◆ Its important to form a good understanding of the tool's methodology

HDLs simplify Design Capture & Design Automation

HDL-Based Chip Design Flow



... Design Flow

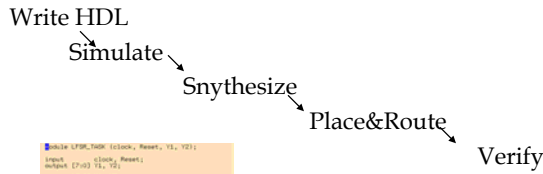


Synthesis-based chip design

The chip is designed using synthesis tools

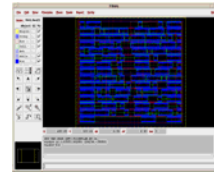
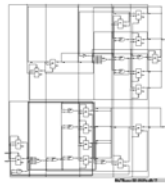
- Used when time-to-market is the most important issue

Basic Steps:



```

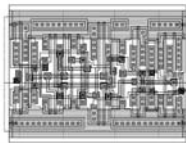
module LFSR_TAP0 (clock, Reset, Y1, Y2);
input clock;
output [7:0] Y1, Y2;
reg [7:0] Y1, Y2;
parameter [7:0] seed0 = 8'h00010001;
parameter [7:0] seed1 = 8'h00110111;
parameter [7:0] TAP0 = 8'h00001101;
parameter [7:0] TAP1 = 8'h00001101;
task LFSR_TAP00, TAP01;
input [7:0] S;
output [7:0] TAP0;
integer N;
reg [6:0] Next_LFSR_Reg;
reg [7:0] Next_LFSR_Reg;
begin
    LFSR_0_Zero = ~ 4'b0101;
    Feedback = S[7] ^ S[6] ^ S[4] ^ S[0];
    for (N=0; N<= 7; N++)
        Next_LFSR_Reg[N] = S[N-1] ^ Feedback;
    S[0] = S[7];
    Next_LFSR_Reg[0] = Feedback;
end
endtask
endmodule
    
```



- Over 40 detailed steps in design process

IC Design Approaches

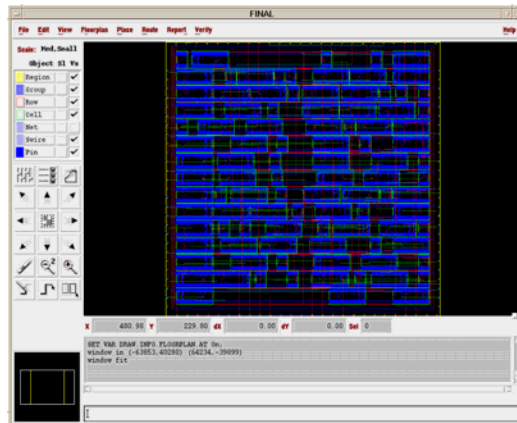
HDL and synthesis based design is used in both Application Specific Integrated Circuits (ASICs):



D-flip-flop



NOR gate

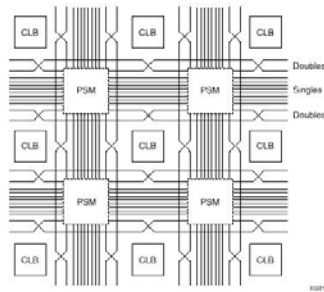


Place and Route Tool

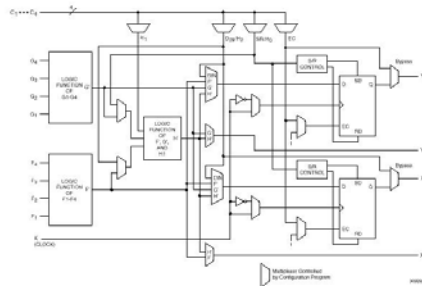
...IC Design Approaches

and Field Programmable Gate Arrays (FPGAs):

Xilinx FPGA architecture



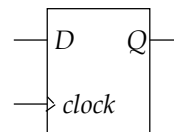
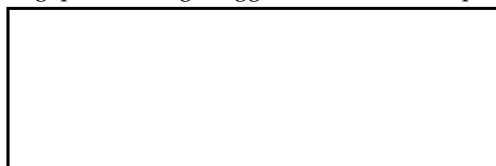
Configurable Logic Block (CLB):



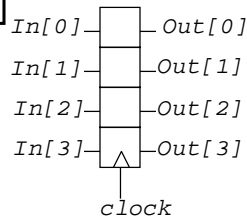
General Principles of Digital System Design

Most Digital Systems are Synchronous

- I.e. All signals are derived off a single Master Clock fed to all registers
- In most logic implementation families, events are synchronized using edge-triggered flip-flops
 - ◆ e.g. positive-edge triggered D- or Data- flip-flop

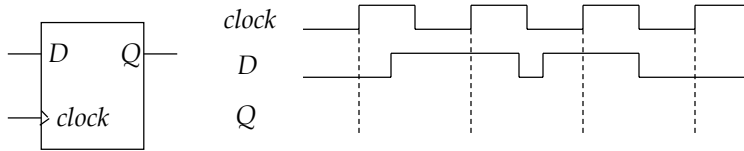


- Groups of flip-flops are referred to as registers



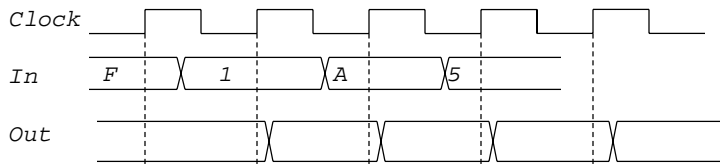
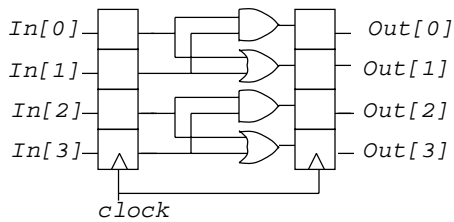
Timing Diagram

The core tool for analyzing synchronous digital designs is a timing diagram.



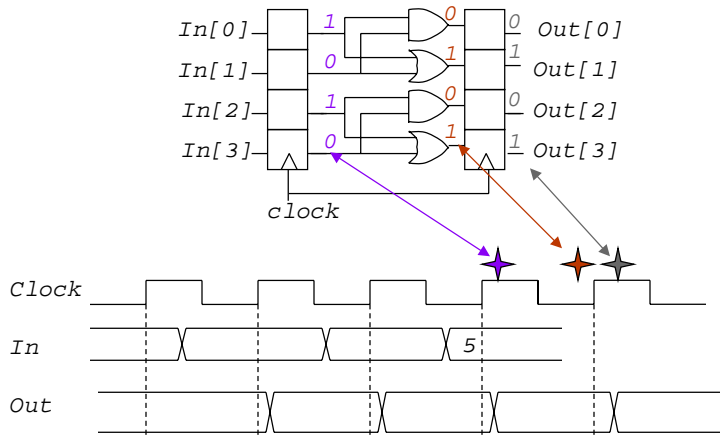
Multiple Logic Stages

Generate Timing Diagram



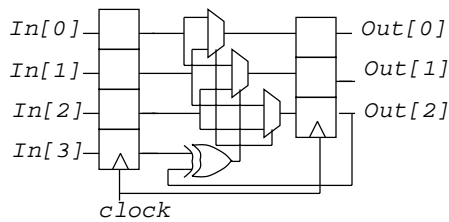
Animation

Track the logic...

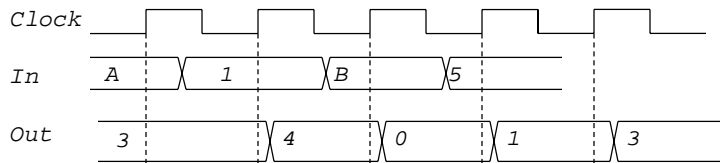


Logic With Feedback

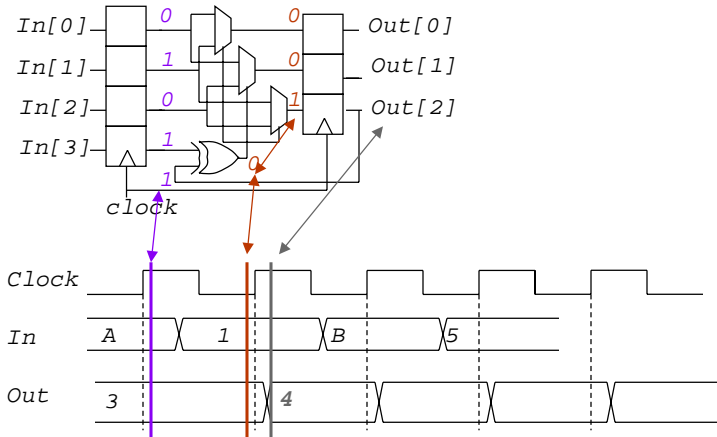
Generate Timing Diagram



Q. If we don't know the initial value of "Out" can we answer this question?



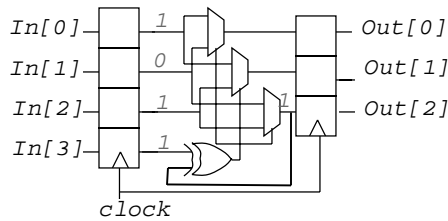
Snapshot



What is Wrong Here?

Combinational Logic Feedback does not work in synchronous design

- Must Feedback through register



- What happens if the logic gets into the state shown?

Summary

What are the advantages of using Hardware Description Languages?

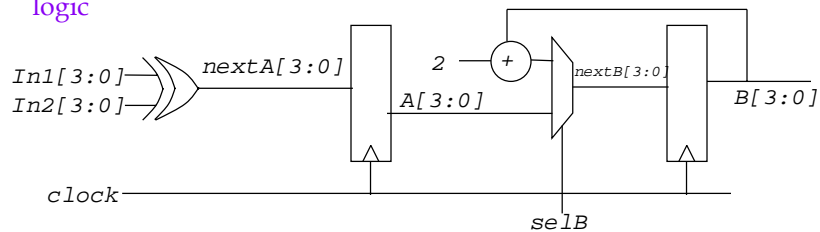
What is the essential element of Synchronous Design?

How does an edge triggered flip-flop work?

Does putting feedback into combinational logic work?

...Digital System Design

Digital sub-systems are built as collections of registers and combinational logic



- Registers store data from the end of one clock period to be available at the start of the next clock period
- Combinational logic can not store data
 - ◆ It only operates on it during each clock period