Design of a 4-Bit Sequence Detector

FSM in VHDL – Nexys 4 DDR Implementation

Tanner Roberson

Project Requirements

- Detect user-defined 4-bit sequence using FSM
- Allow overlap
- VHDL behavioral modeling in AMD Vivado
- Generalized FSM (not hardcoded)
- Test bench for all 16 sequences
- Implement on Nexys 4 DDR:
  SW0: Clock, SW1: Input w
  SW5–SW2: Target sequence
  CPU Reset: Reset
  LD0: Output z

Learning Objectives

- Build sequence detector FSM for any 4-bit sequence
- Design and simulate VHDL test bench
- Practice behavioral modeling
- Learn generalized FSM design
- Implement and test on FPGA

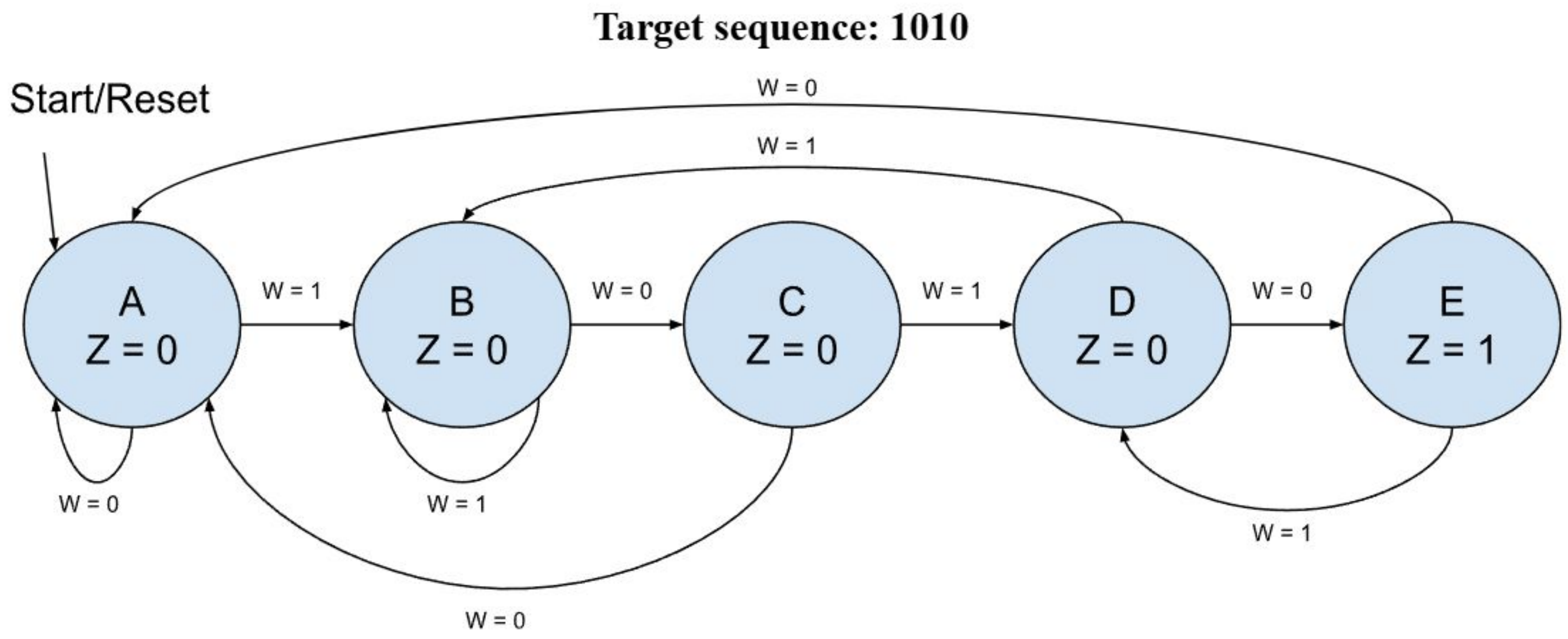
General Approach

1. Define FSM states
2. Write behavioral VHDL FSM logic
3. Implement reset
4. Create test bench for all sequences
5. Verify with waveforms
6. Deploy on FPGA

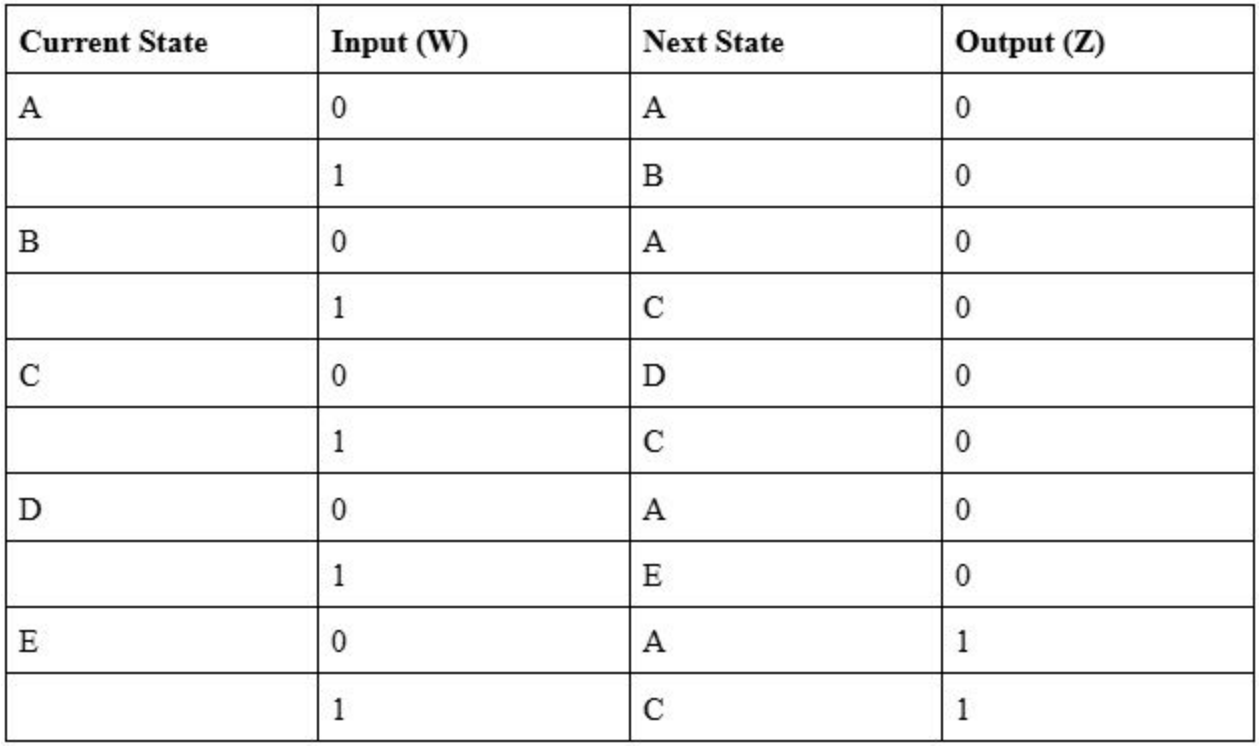
FSM Design Details

- Shift inputs into 4-bit register each cycle
- FSM checks if input == target sequence
- Output z=1 on match
- Overlap handled via transitions
- Reset returns to beginning state

Example State Diagram (1010)



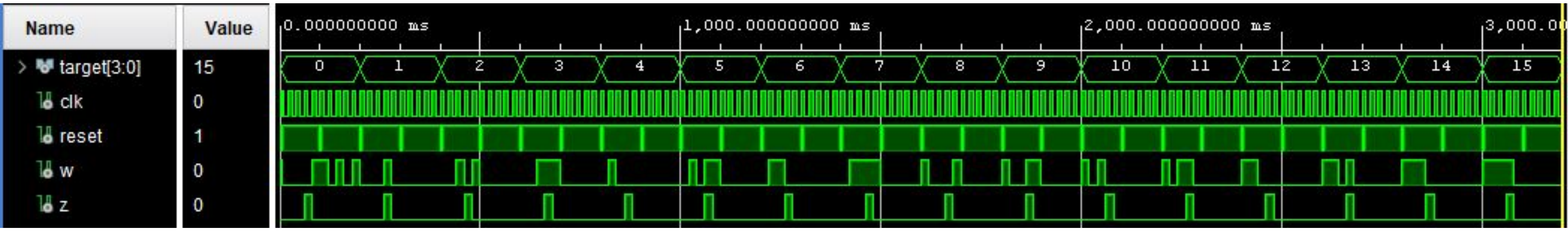
Example State Table (1010)



Current State	Input (W)	Next State	Output (Z)
A	0	A	0
	1	B	0
B	0	A	0
	1	C	0
C	0	D	0
	1	C	0
D	0	A	0
	1	E	0
E	0	A	1
	1	C	1

Simulation Results

- 10 inputs per sequence
- 5 correct → z=1 at 4th input
- 5 incorrect → z=0



FPGA Implementation

- Nexys 4 DDR connections:
- SW0: Clock
- SW1: Input w
- SW5–SW2: Target sequence
- Reset: CPU Reset
- LD0: Output z

Observations & Challenges

- Concurrent VHDL statements caused 1-cycle delay
- Solution: store last 3 inputs in shift register
- FSM generalized for all 16 sequences

Summary & Takeaways

– Designed generalized FSM sequence detector

– Verified with simulation waveforms

– Implemented on Nexys 4 DDR FPGA

– Learned FSM overlap handling, VHDL modeling, test benching, FPGA deployment