Digital Logic Design using Verilog and FPGA devices
Part 2

An Introductory Lecture Series

By

Chirag Sangani

A Small Recap

- Verilog allows us to design circuits, FPGAs allow us to test these circuits in real-time.
- The basic unit in a Verilog code is a `module`. A `module` consists of I/O through wires or registers.
- An `always` block allows us to implement sequential circuits as well as complex combinatorial circuits. It enables behavior-based programming.

Conditional Statements

```
module UpDownCounter(
        input wire CLK,
        input wire DIR,
        output reg [7:0] COUNT);

initial
        COUNT <= 8'b0;

always @(posedge CLK)
        if (DIR == 1)
                COUNT <= COUNT + 1;
        else
                COUNT <= COUNT – 1;

endmodule
```

Conditional Statements

- Alternatively:

```
module UpDownCounter2(
      input wire CLK,
      input wire DIR,
      output reg [7:0] COUNT);

initial
      COUNT <= 8'b0;

always @(posedge CLK)
      COUNT <= COUNT + ((DIR==1) ? 1 : -1);

endmodule
```

Parameterized Modules

- A generalized type of module.
- Can be instantiated to any value of parameter.
- Useful in large circuits.

Example: N-bit adder

```
module AdderN #(parameter N = 4) (
    input wire [N-1:0] IN1,
    input wire [N-1:0] IN2,
    output reg [N-1:0] OUT);

always @(*)
    OUT <= IN1 + IN2;

endmodule
```

Modular Circuits

- A modular circuit is one where sub-modules are initialized with interconnects to form even a larger circuit.
- Each sub-module resides in its own Verilog file (extension .v). A sub-module may use another sub-module in its circuit.
- The top-level module has to be indicated to the synthesizer at the time of synthesis.

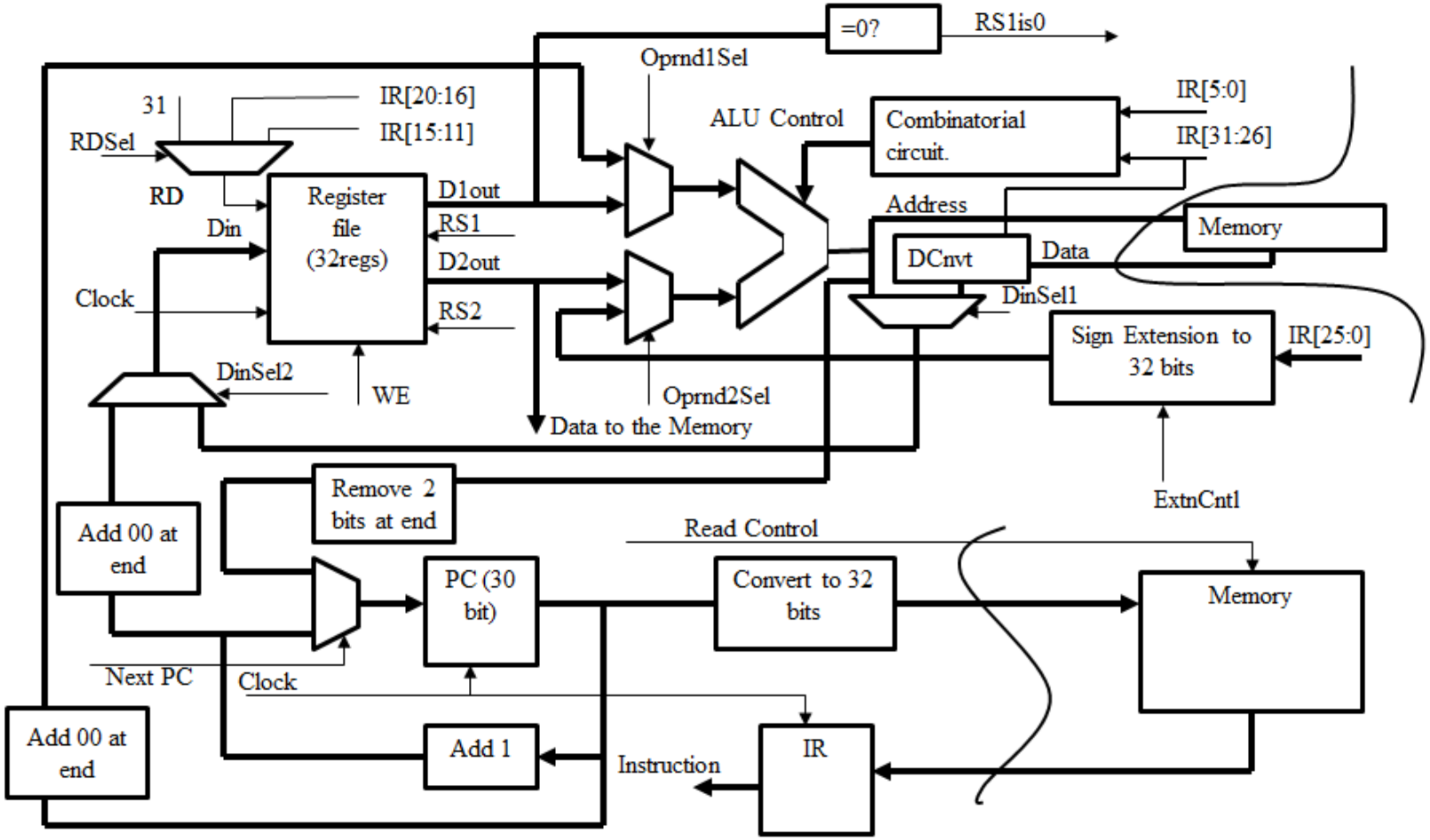
An Example Modular Circuit

```
module MultiSevenSeg (
        input wire [3:0] INP,
        input wire TYPE,
        output reg [6:0] OUT);

wire [6:0] DecToInv;
wire [6:0] INVERTOUT;

SevenSegDec DECODER (
        .inp(INP),
        .out(DecToInv));

BusInverter #(.N(7)) INVERTER (
        .A(DecToInv),
        .B(INVERTOUT));

always @(*)
        OUT <= (TYPE == 1) ?
DecToInv : INVERTOUT;

endmodule
```

```
module SevenSegDec (
        input wire [3:0] inp,
        output reg [6:0] out
);

module BusInverter (
        input wire [N-1:0] A,
        output wire [N-1:0] B
);
```

Why Are Modular Circuits Better?

Why Are Modular Circuits Better?



A two-stage pipelined SDLX Processor
Source: C. Sangani, A. Kasina (2)

Sequential Circuits

- Time-dependent circuits: their state is determined not just by the input but also by current state.
- Their behavior may vary for the same input at different times.
- They may exhibit output without any input.

LCD Driver

- Control of 16*2 character LCD display.
- Control interface consists of a 7 bit bus: 4 bits data and 3 bits instructions.
- To control the LCD, the data bits and the control bits have to be set, and the LCD_E bit has to be strobed at a specified maximum frequency.

LCD Driver

```
module LCDDriver(
        input wire CLK,
        output reg[7:0] LCDCONTROL);

reg [25:0] FREQGEN;

always @(posedge CLK)
begin
        FREQGEN <= FREQGEN + 1;
        case(FREQGEN[25:17])
                0: LCDCONTROL <= XYZ;
                1: LCDCONTROL <= ABC;
        endcase
end

endmodule
```

FPGA Design Challenge Problem Statement

128-BIT AES ENCRYPTION

Terminology

- Plaintext: Input data to the encryption block.
- Ciphertext: Encrypted output by encryption block.
- Key: A secret binary number known by the two communicating parties

Advanced Encryption System

- A symmetric-key encryption standard.
- Key size: 128 bit.
- Plaintext block size: 128 bit.

High-Level Description of the Algorithm

- `KeyExpansion` — round keys are derived from the cipher key using Rijndael's key schedule
- Initial Round
  - `AddRoundKey` — each byte of the state is combined with the round key using bitwise XOR
- Rounds
  - `SubBytes` — a non-linear substitution step where each byte is replaced with another according to a lookup table.
  - `ShiftRows` — a transposition step where each row of the state is shifted cyclically a certain number of steps.
  - `MixColumns` — a mixing operation which operates on the columns of the state, combining the four bytes in each column.
  - `AddRoundKey`
- Final Round (no MixColumns)
  - `SubBytes`
  - `ShiftRows`
  - `AddRoundKey`

Requirements

- Develop a module that performs 128-bit AES encryption.
- The module must be completely self-written and completely self-sufficient.
- Input (key and plaintext) and output (ciphertext) must be through I/O or RAM.

References

1. R. Haskell, D. Hanna: “Introduction to Digital Design Using Digilent FPGA Boards – Block Diagram / Verilog Examples”; available at http://www.digilentinc.com/Data/Textbooks/Intro_to_Digital_Design-Digilent-Verilog_Online.pdf
2. C. Sangani, A. Kasina: “Digital Design Using Verilog and FPGAs: An Experiment Manual”; available at http://www.chiragsangani.com/projects/electronics/FPGADesignManual

References

3. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
   This document outlines every detail of the AES and is considered as the final reference. You are advised to go through this document thoroughly for understanding the problem statement.
4. http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
   A learner-friendly description of the problem statement. Please be advised that this is not the final reference and in case of any conflict, the details mentioned in reference 3 shall be considered as final.
5. http://www.movable-type.co.uk/scripts/aes.js
   A javascript implementation of the AES scheme. This resource will serve useful as a reference pseudo-code. You are advised to ensure that your final implementation stays true to the original standard as described in reference 3.