Sistemas Digitales

jasson DB

February 2025

1. Objetivo de la materia

El objetivo de Sistemas Digitales es que puedan comprender los principios de diseño y funcionamiento de:

- **La arquitectura de una computadora**
  O sea, como se programa en el lenguaje que la maquina interpreta.
- **Su microarquitectura**
  O sea, como se construye una computadora que pueda interpretar un programa.

2. Hardware - su práctica

Al igual que el software, la práctica del diseño y la implementación del hardware depende del siguiente grupo de actividades:

Diseño → Especificación → Implementación → Validación → Verificación

2.1. Diseño

Descripción esquématica de los componentes de un sistema y como se conectan sus interfaces.

2.2. Especificación

Descripción del comportamiento esperado en términos de presunciones y garantías en términos de sus señales y estado (memoria).

2.3. Implementación

Descripción de comportamiento Una descripción del comportamiento del sistema en un lenguaje apropiado (HDL) de la que puede sintetizarse una implementación funcional.

2.4. validación

Conjunto de pruebas no exhaustivas que prueban el comportamiento de la especificación y/o implementación.

2.5. verificación

Prueba de propiedades formales (con garantías basadas en algún mecanismo matemático) de la especificación y/o implementación

3. Alcance de la práctica

En la práctica de la materia vamos a concentrarnos en las etapas de **diseño** y **implementación**, la primera al estudiar las arquitecturas existentes, sus motivaciones y prácticas comunes, y la segunda al construir un procesador de 32 bits sobre una arquitectura RISC V utilizando un lenguaje de descripción de hardware (HDL) llamado SystemVerilog. La ejecución de tests sobre los componentes con los que vamos a trabajar servirán como una instancia de **validación**.

4. Arquitectura del microprocesador

Recordemos que queremos diseñar e implementar un procesador. Para el alcance de la materia vamos a analizar:

- **La arquitectura**
  Set de instrucciones, registros, memoria (la interfaz a la que accede quien vaya a programar el procesador).
- **La microarquitectura**
  Implementación de los componentes, camino de datos, lógica de control (implementación en un soporte electrónico).

5. Contenidos de la materia

- **Lógica Combinatoria y Secuencial**

- **Diseño de un set de instrucciones**
  → **Arquitectura**
- **Lenguajes de descripción de hardware**
  → **Microarquitectura**

5.1. Lógica Combinatoria

Presenta los principios fundamentales para construir circuitos que implementen en un soporte electrónico la semántica de la lógica proposicional.

5.2. Lógica Secuencial

Introduce los elementos básicos para mantener el valor de un dato a lo largo del tiempo, los mecanismos de sincronización de circuitos y junto con estos la capacidad y técnicas que nos permiten descomponer e implementar un cómputo complejo en una secuencia de pasos atómicos.

5.3. La arquitectura

Va a definir qué instrucciones va a poder ejecutar el procesador, sobre que registros vamos a poder trabajar y cómo se accede a la memoria.

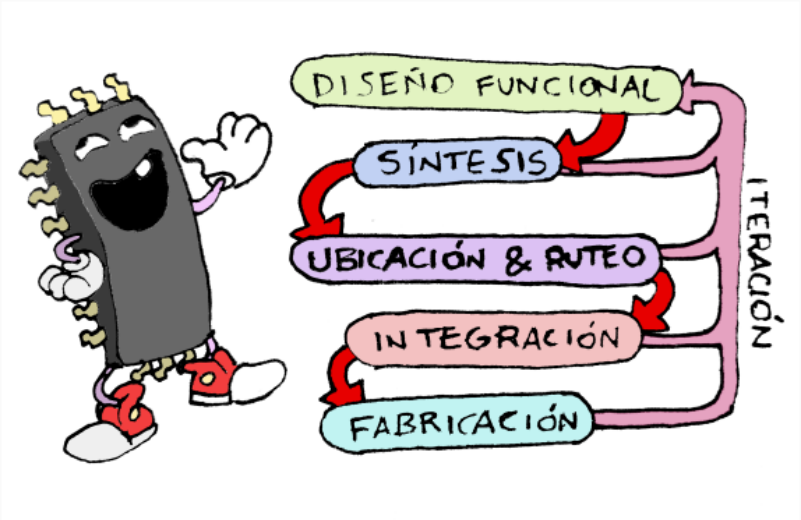
5.4. Los lenguajes de descripción de hardware

Son la forma en la que se describe hardware a nivel industrial y vamos a utilizarlos para construir nuestro procesador.

5.5. La microprogramación

Va describir la forma en que nuestros componentes sincrónicos interactúen para implementar las operaciones descriptas en el set de instrucciones previamente definido.

Fabricación del hardware



Hardware description language (HDL)

Del diseño a la implementación tenemos: Un lenguaje de descripción de hardware, o HDL por sus siglas en inglés (hardware description language) es un lenguaje que describe la estructura y el comportamiento de un circuito digital. Los dos lenguajes más utilizados en la industria son VHDL y Verilog. En Sistemas Digitales vamos a usar **Verilog**.

SystemVerilog

```
module maindec(input  logic [6:0] op,
               output logic [1:0] ResultSrc,
               output logic       MemWrite,
               output logic       Branch, ALUSrc,
               output logic       RegWrite, Jump,
               output logic [1:0] ImmSrc,
               output logic [1:0] ALUOp);
  logic [10:0] controls;

  assign {RegWrite, ImmSrc, ALUSrc, MemWrite,
          ResultSrc, Branch, ALUOp, Jump} = controls;

  always_comb
    case(op)
    // RegWrite_ImmSrc_ALUSrc_MemWrite_ResultSrc_Branch_ALUOp_Jump
      7'b0000011: controls = 11'b1_00_1_0_01_0_00_0; // lw
      7'b0100011: controls = 11'b0_01_1_1_00_0_00_0; // sw
      7'b0110011: controls = 11'b1_xx_0_0_00_0_10_0; // R-type
      7'b1100011: controls = 11'b0_10_0_0_00_1_01_0; // beq
      7'b0010011: controls = 11'b1_00_1_0_00_0_10_0; // I-type ALU
      7'b1101111: controls = 11'b1_11_0_0_10_0_00_1; // jal
      default:    controls = 11'bx_xx_x_x_xx_x_xx_x; // ???
    endcase
endmodule
```