UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

ARQUITECTURA DE COMPUTADORAS

JORGE ERNESTO LOPEZ ARCE DELGADO

ACTIVIDAD 02 - COMPUERTAS LOGICAS

CESAR DAVID AMEZCUA NARANJO

Introducción

Una FPGA (Field-Programmable Gate Array) es un circuito integrado diseñado para ser configurado por el usuario después de su fabricación. A diferencia de los circuitos integrados de propósito específico (ASIC), las FPGAs permiten implementar diseños digitales flexibles y reconfigurables, lo que las hace ideales para prototipado rápido, desarrollo de sistemas embebidos y aplicaciones especializadas. El lenguaje de descripción de hardware Verilog es una herramienta fundamental para programar FPGAs, ya que permite describir el comportamiento y estructura de circuitos digitales a nivel de puertas lógicas, registros y sistemas complejos.

Objetivos

- Comprender el concepto de FPGA y su relación con los lenguajes de descripción de hardware (HDL).
- Verificar la correcta instalación del entorno de desarrollo (compilador/simulador) para trabajar con Verilog.
- Analizar la sintaxis de operadores lógicos combinacionales en Verilog y su aplicación en la implementación de compuertas básicas.

Desarrollo

¿Qué es una FPGA?

Una FPGA está compuesta por bloques lógicos programables (CLB), recursos de memoria, multiplicadores digitales y conexiones reconfigurables. Estos elementos se interconectan mediante una matriz de ruteo controlada por una bitstream (archivo de configuración generado tras sintetizar el código HDL). Su flexibilidad permite implementar desde sistemas simples (como compuertas lógicas) hasta procesadores completos (ej.: softcore como MicroBlaze o RISC-V).

Relación con Verilog

Verilog es un HDL estandarizado (IEEE 1364) que describe el comportamiento de circuitos digitales mediante código textual. Al escribir un módulo en Verilog (ej.: compuertas lógicas), el proceso de síntesis convierte este código en una representación física compatible con la arquitectura de la FPGA. Por ejemplo:

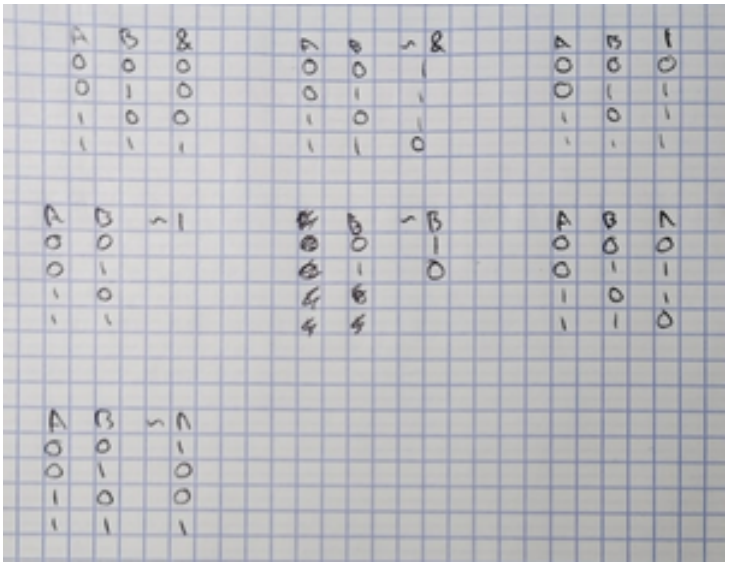
- Las asignaciones `assign AND_out = A & B;` se mapean a recursos lógicos de la FPGA (LUTs o bloques combinatorios).

- El simulador (ej.: ModelSim) verifica el funcionamiento antes de programar la FPGA.

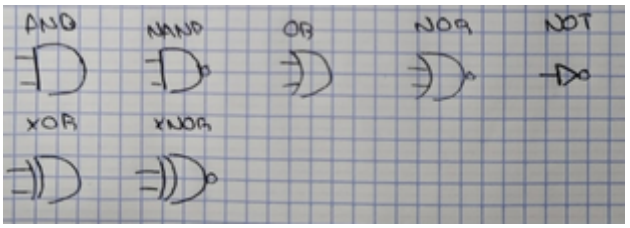
Implementación de compuertas

Tablas de verdad y diagramas de compuertas lógicas

Para comprender y validar el comportamiento del módulo implementado en Verilog, es fundamental referirse a las **tablas de verdad** de cada compuerta lógica. Estas tablas definen todas las combinaciones posibles de las entradas (A y B, excepto en NOT que solo usa A) y sus respectivas salidas. A continuación se resumen:



Adicionalmente, los diagramas de compuertas representan gráficamente cada operación lógica. Por ejemplo:



El archivo `.v` proporcionado implementa las 7 compuertas en un único módulo. Su estructura básica es:

```
module compuertasLogicas(
    input entradaA,
    input entradaB,
```

```
    output and_en,    // Habilitar AND
    output nand_en,   // Habilitar NAND
    output or_en,     // Habilitar OR
    output nor_en,    // Habilitar NOR
    output not_en,    // Habilitar NOT
    output xor_en,    // Habilitar XOR
    output xnor_en,   // Habilitar XNOR
);
```

Este código define operaciones combinacionales (sin retraso de temporización), donde cada salida refleja instantáneamente el resultado de la operación lógica. La síntesis en la FPGA asignará recursos físicos para cada operación, aprovechando la naturaleza paralela de los HDL.

Conclusiones

- Las FPGAs son plataformas versátiles para implementar diseños digitales mediante HDLs como Verilog.

- Verilog actúa como puente entre la descripción lógica del sistema y la configuración física de la FPGA.

- La correcta instalación del entorno de desarrollo es crítica para validar diseños mediante simulación antes de programar la FPGA.

Referencias

61691-4-2004 - IEC 61691-4 Ed.1 (IEEE Std 1364(TM)-2001): Behavioural Languages - Part 4: Verilog(C) Hardware Description Language
https://ieeexplore.ieee.org/document/1406532
Recuperado el 27 de agosto de 2025, de https://ieeexplore.ieee.org

Programación en Verilog - Wikilibros. (s. f.).
https://es.wikibooks.org/w/index.php?title=Programaci%C3%B3n_en_Verilog&oldid=244419
Recuperado el 27 de agosto de 2025, de https://es.wikibooks.org

Implementacion de Codigó

El archivo `.v` ya implementado sigue la estructura mostrada en el Desarrollo. Cada compuerta se define con operadores Verilog nativos (`&`, `|`, `~`, `^`), garantizando que el sintetizador interprete correctamente las operaciones lógicas. Para verificar su funcionamiento:

1. Simular con vectores de prueba (ej.: `A=0, B=1`).
2. Comparar salidas con las tablas de verdad esperadas.
3. Generar la bitstream y programar la FPGA para validación física.

```
module compuertasLogicas(
    input entradaA,
    input entradaB,
    output and_en,    // Habilitar AND
    output nand_en,   // Habilitar NAND
    output or_en,     // Habilitar OR
    output nor_en,    // Habilitar NOR
    output not_en,    // Habilitar NOT
    output xor_en,    // Habilitar XOR
    output xnor_en,   // Habilitar XNOR
);

    // Compuerta AND
    assign and_en = entradaA & entradaB;

    // Compuerta NAND
    assign nand_en = ~(entradaA & entradaB);

    // Compuerta OR
    assign or_en = entradaA | entradaB;

    // Compuerta NOR
    assign nor_en = ~(entradaA | entradaB);

    // Compuerta NOT (solo de entradaA)
    assign not_en = ~entradaA;
```

```
    // Compuerta XOR
    assign xor_en = entradaA ^ entradaB;

    // Compuerta XNOR
    assign xnor_en = ~(entradaA ^ entradaB);

endmodule
```