Diseño digital en Verilog: Comparador de palabras de N bits

Grupo 3
Lidia Magaly García González, B83169
Roger Daniel Piovet García, C15990

IE0323 Circuitos Digitales I GR003
Escuela de Ingeniería Eléctrica
Universidad de Costa Rica

29 noviembre 2023

Diseño

Se siguió la metodología de diseño mediante redes iterativas para ambos recorridos [1]

1. Definir los estados que resuelven el problema.
2. Construir la Tabla de Transición de Estados.
3. Hacer la asignación de estados, la declaración de variables de estado y variables de salida.
4. Hacer la Tabla de Transición de Estados codificada.
5. Dibujar el diagrama de bloque de la celda típica.
6. Diseñar la celda típica.
7. Diseñar la celda inicial.
8. Diseñar la celda final.

Diseño (Izquierda a derecha)

Definición de estados

- *a*: Hasta el momento, la primera palabra es igual a la segunda palabra porqué los pares de bits en la misma posición en ambas palabras recibidos han sido los mismos, o el estado se encuentra presente en la celda inicial **(estado inicial)**.
- *b*: La primera palabra es mayor a la segunda palabra porqué se ha recibido un bit igual a uno en la primera palabra y un bit igual a cero en la segunda palabra en la misma posición en ambas palabras.
- *c*: La segunda palabra es mayor a la primera palabra porque se ha recibido un bit igual a cero en la primera palabra y un uno en la segunda palabra en la misma posición en ambas palabras.

Asignación de estados: *a*:01 *b*:10 *c*:11

Diseño (Derecha a izquierda)

Definición de estados

- *a*: Hasta el momento, la primera palabra es menor o igual a la segunda palabra porqué los pares de bits en la misma posición en ambas palabras recibidos han sido los mismos, o se ha recibido un bit igual a cero en la primera palabra y un bit igual a uno en la segunda palabra, o el estado se encuentra presente en la celda inicial **(estado inicial)**.
- *b*: Hasta el momento, la primera palabra es mayor a la segunda palabra porqué se ha recibido un bit igual a uno en la primera palabra y un bit igual a cero en la segunda palabra en la misma posición en ambas palabras.

Asignación de estados: *a*:1 *b*:0

Diseño

Por medio de mapas de Karnaugh, se obtuvieron las funciones de próximo estado y de la salida final.

- Izquierda a derecha

$$Z = P = p(\overline{A_i} + B_i) + \overline{A_i} B_i$$
$$P = p + A_i \oplus B_i$$

- Derecha a izquierda

$$Z = Q = q(p + \overline{A_i} + B_i)$$

Implementación

Se partió del siguiente criterio técnico

- Diseño con menos compuertas $\Rightarrow$ Estructural
- Diseño con más compuertas $\Rightarrow$ Conductual

Para cada recorrido, se crearon módulos para cada celda y un módulo para la red iterativa.

- `celdaInicial`
- `celdaTipica`
- `celdaFinal`
- `redIterativa`

Puertos I/O en base a sus diagramas de bloques.

Implementacion

- Red inicial

```
module redIterativaDerIzq_test
(
    input [2:0] A, B,
    output wire Zout
);
    wire [1:0] P;

    celdaInicialDerIzq celdaInit (.A0(A[0]), .B0(B[0]),
↪   .Pinit(P[0]));
    celdaFinalDerIzq celdaFin (.An_1(A[2]), .Bn_1(B[2]),
↪   .p(P[2]), .Z(Zout));
    celdaTipicaDerIzq celdaTyp (.p(P[0]), .Ai(A[1]),
↪   .Bi(B[1]), .P(P[1]) );
endmodule
```

Implementación

- Próximo estado `P[i]`, estado presente `P[i - 1]`. A partir de este patrón, se utiliza `generate`.

```
generate
    for (i = 0; i <= N - 1; i = i + 1)
    begin
        case (i)
            0:       //instanciar celda inicial
            N - 1:   //instanciar celda final
            default: //instanciar celda típica
        endcase
    end
endgenerate
```

Pruebas

Testbenchs para cada módulo de ambos recorridos. Se hicieron las siguientes pruebas

- Combinaciones binarias de A_i y B_i para la celda inicial
- Combinaciones binarias de A_i y B_i con cada estado presente posible para la celda típica y final
- Palabras A y B predefinidas para la red iterativa
- Casos de esquina y combinaciones binarias entre A y B con N bits para la red iterativa

Pruebas

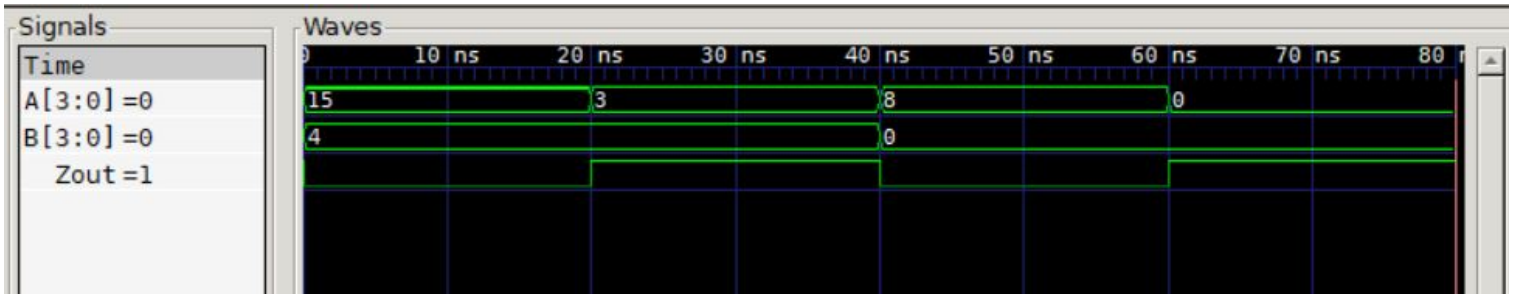

Figure: Prueba realizada por medio de un testbench en Verilog a la implementación conductual de la red con casos explícitos

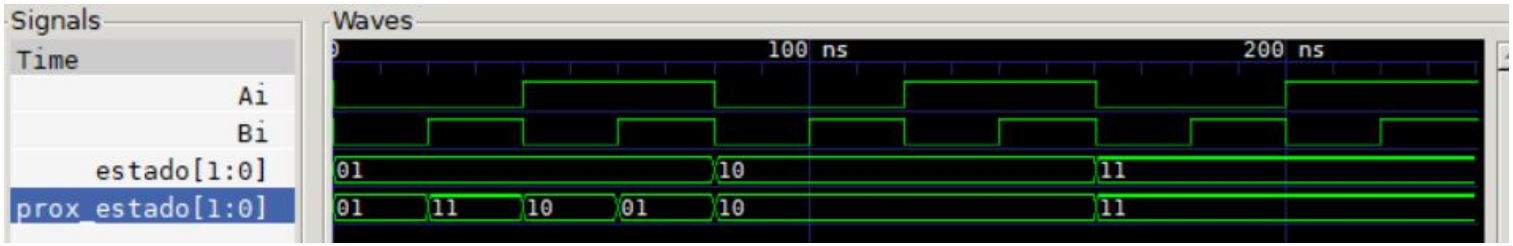

Figure: Prueba realizada por medio de un testbench en Verilog a la implementación conductual de la celda típica

Pruebas

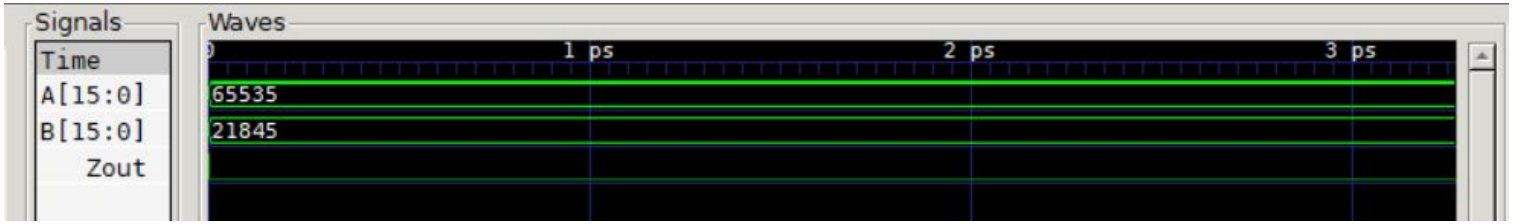

Figure: Prueba realizada por medio de un testbench en Verilog a la implementación conductual de la red con una cantidad de bits 'grande'

Referencias


Geovanny Delgado Cascante. Redes Iterativas I Parte. Universidad de Costa Rica, Escuela de Ingeniería Eléctrica. 12 edition, 2011


Geovanny Delgado Cascante. Lenguaje de descripción de Hardware: VERILOG. Universidad de Costa Rica, Escuela de Ingeniería Eléctrica.