Lab01 - ARCH - 2021-II

{TAs: rodrigo.salazar, Prof: ladanaque} @utec.edu.pe

September 10, 2021

1 Indicaciones

1. Realizar los ejercicios en Verilog (usar Icarus y GTKWave).
2. Mantener buenas prácticas de diseño usando Verilog (nombres, orden, etc.).
3. Generar ***waveforms*** en archivos VCD creando un *testbench* para cada ejercicio.
4. Escribir un informe explicando de forma breve y directa el diseño de cada ejercicio. Según sea necesario: mostrar ecuaciones booleanas, K-map, esquemático, figuras, tablas de verdad, indicar valores usados u obtenidos en testbench.
5. Separar en carpetas para cada ejercicio **con los nombres:** Ejercicio 1, Ejercicio 2, Ejercicio 3, Ejercicio 4. Subir en un archivo comprimido (.zip) en Gradescope **solamente**: 1) carpetas con archivos en Verilog (los módulos y testbench), VCD files; e 1) informe en .pdf (fuera de las carpetas).
6. Fecha de **entrega**: `Verificar Sección de Tareas L01 de Canvas`.
7. Las entregas son realizadas **solamente por Gradescope.**
8. Cualquier intento de plagio parcial o total es sancionado por UTEC en todas sus atribuciones.

2 Ejercicios

Implementar para cada ejercicio sus módulos en Verilog y testbench respectivo. **Justificar todas sus implementaciones y respuestas** analizando las ecuaciones booleanas y K-maps. **Nota:** En todos los diseños usar jerarquía de módulos y submódulos. La implementación dentro de cada submódulo:

- En caso *structural* usará primitivas de logic gates.
- En el caso *behavioral* usará operadores funcionales.

1. Implementar los siguientes *MUX*. **OBS:** Usar submódulos *MUX2:1* (repetir) dentro del top.:
   - MUX 16:1 de `8-bits` (inputs y outputs): a) structural y b) behavioral.
2. Implementar los siguientes *DEMUX* (operación inversa al MUX). **OBS:** Usar submódulos *DEMUX1:2* (repetir) dentro del top.:
   - DEMUX 1:16 de 16-bits (inputs y outputs).
   - Responder: es posible implementar un un DEMUX de 1:13 de 8-bits?. Justificar.
3. Implementar un comparador de dos bits correspondiente a la Tabla 1. La Figura 2 muestra el circuito esquemático, donde AB y CD son las entradas de dos bits. La salida $F1$ es 1 cuando AB es igual a CD; $F2$ es 1 cuando $AB < CD$ y $F3$ es 1 $AB > CD$.

A	B	C	D	F1	F2	F3
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	1	0	0

Figure 1: *Truth-table* del comparador de dos bits.

4. Implementar un BCD (Binary Coded Decimal) correspondiente a la Tabla 3. Este codificador BCD genera dígitos codificados del 0 al 9, usando como entradas el rango de valores binarios de 0000 a 1001, valores binarios mayores generan salidas *"don't care"*.

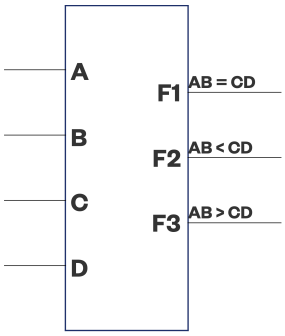

Figure 2: *Schematic* del comparador de dos bits.

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Figure 3: *Truth-table* del BCD.

3 Referencias

1. **Verilog Quick Reference:** Verilog Quick Ref.
2. Libros de referencia y material de clase (ver Sílabo).