# Universidad de Costa Rica

## Sede Rodrigo Facio
## Facultad de Ingeniería
## Escuela de Ingeniería Eléctrica

---

IE-0523 Circuitos Digitales II

Tarea 3

---

Prof. Jorge Soto

*Andrés Alvarado Velázquez*

B30316

*11 de abril*
*I Ciclo 2018.*

# 1. Resumen

Se programa por comportamiento una compuerta not, nand y nor; además de un mux de 2 a 1 y un flip flop tipo D. A estos se le programa los tiempo de propagación, en el caso de las compuertas. Para comprobar el comportamiento lógico de las compuertas y lógica secuencial se propone pruebas que consisten en varias las entradas de tal forma que la salida cumpla con la tabla de verdad; para lo cual el modelo programado resultó exitoso. Con respecto al comportamiento temporal es evidente con el uso de la herramienta GtkWave.

Con respecto al flip flop, no se logra implementar los tiempos ni el conteo de actividad en este por cuestiones de tiempo.

# 2. Descripción Arquitectónica

Se pretende programar el comportamiento de 5 componentes, 3 de estos compuertas lógicas y 2 secuenciales, sean estos los que se presentan a continuación:

## 2.1. NOT

La compuerta *NOT*, el comportamiento deseado es que la señal $AN$ sea lo contrario a la señal $A$.

## 2.2. NAND

La compuerta *NAND* tiene un comportamiento deseado es que la señal $C$ esté en bajo cuando las señales $A$ y $B$ estén ambas en alto, caso contrario C debería estar en alto.

## 2.3. NOR

La compuerta *NOR*, el comportamiento deseado es que la señal $C$ esté en alto cuando las señales $A$ y $B$ estén ambas en bajo, caso contrario C debería estar en bajo.

## 2.4. MUX

El componente secuencial *MUX*, el compotamiento deseado es que la señal $C$ sea el valor de la señal $A$ cuando la senñal *SEL* esté en alto, mientras tanto cuando *SEL* está en bajo la senñal $C$ deberá contener el valor de $B$.

## 2.5. FFD

La figura 1 muestra un *flip flop tipo D*, donde las señales $D$, *CLK*, *Preset negado* y *CLR negado* son los input y las señales $Q$ y $Q$ *negado* son las salidas.

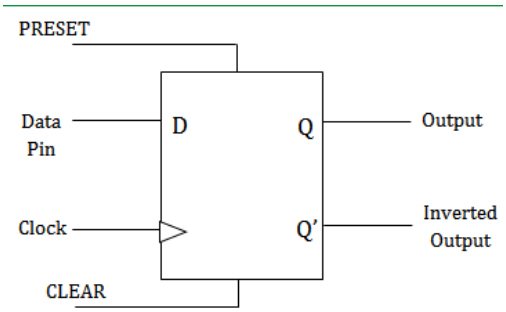


Figura 1: Flip flop tipo D

El comportamiento deseado lo describe la tabla de verdad en la figura 2.

| INPUTS | | | | OUTPUTS | |
|---|---|---|---|---|---|
| $\overline{PR}$ | $\overline{CLR}$ | CLK | D | Q | $\overline{Q}$ |
| 0 | 1 | X | X | 1 | 0 |
| 1 | 0 | X | X | 0 | 1 |
| 0 | 0 | X | X | X | X |
| 1 | 1 | ↑ | 1 | 1 | 0 |
| 1 | 1 | ↑ | 0 | 0 | 1 |
| 1 | 1 | 0 | X | $Q_0$ | $\overline{Q}_0$ |

Figura 2: Tabla de verdad de un flip flop tipo D

### 2.6. Consideraciones

Con respecto a los tiempo de propagación, la mayoría de los datasheet sólo presentan datos de los tiempo mínimos y máximos, siendo necesario el valor típico ya que verilog no toma dos especificaciones por tiempo de subida o caída; por lo tanto, para solucionar esto se considera que el tiempo típico sea un promedio de los anteriores.

## 3. Plan de pruebas

### 3.1. Pruebas de comportamiento

Estas pruebas tienen como objetivo verificar el cumplimiento del comportamiento deseado, además se puede verificar con estas mismas los tiempos de transición con ayuda de la herramienta GtkWave.

#### NAND y NOR

Se plantea la misma prueba para la compuerta NAND y NOR

- Establecer A=0, B=0.

- Enviar flancos activos en CLK, para cargar el valor de C.
- Establecer A=0, B=1.
- Enviar flancos activos en CLK, para cargar el valor de C.
- Establecer A=1, B=0.
- Enviar flancos activos en CLK, para cargar el valor de C.
- Establecer A=1, B=1.
- Enviar flancos activos en CLK, para cargar el valor de C.

**NOT**

- Establecer A=0.
- Enviar flancos activos en CLK, para cargar un 1 en AN.
- Establecer A=1.
- Enviar flancos activos en CLK, para cargar un 0 en AN.
- Establecer A=0.
- Enviar flancos activos en CLK, para cargar un 1 en AN.
- Establecer A=1.
- Enviar flancos activos en CLK, para cargar un 0 en AN.

**MUX**

- Establecer SEL=0.
- Enviar flancos activos en CLK.
- Establecer A=0, B=0.
- Enviar flancos activos en CLK, para cargar el valor de B en C.
- Establecer A=0, B=1.
- Enviar flancos activos en CLK, para cargar el valor de B en C.
- Establecer A=1, B=0.
- Enviar flancos activos en CLK, para cargar el valor de B en C.
- Establecer A=1, B=1.
- Enviar flancos activos en CLK, para cargar el valor de B en C.
- Establecer SEL=1.
- Enviar flancos activos en CLK.

- Establecer A=0, B=0.
- Enviar flancos activos en CLK, para cargar el valor de A en C.
- Establecer A=0, B=1.
- Enviar flancos activos en CLK, para cargar el valor de A en C.
- Establecer A=1, B=0.
- Enviar flancos activos en CLK, para cargar el valor de A en C.
- Establecer A=1, B=1.
- Enviar flancos activos en CLK, para cargar el valor de A en C.

### FLIP FLOP D

Considerar que en la figura 2 se considera los negados de CLK y PRE.

- Establecer CLK=0, D=0, PRE=1, CLR=0.
- Enviar flancos activos en CLK para generar un 1 en Q.
- Establecer PRE=0, CLR=1.
- Enviar flancos activos en CLK para generar un 0 en Q.
- Establecer PRE=1, CLR=1.
- Enviar flancos activos en CLK para generar un 1 en Q y en QN también.
- Establecer PRE=0, CLR=0.
- Enviar flancos activos en CLK para generar un 0 en Q y en QN un 1.
- Establecer D=1.
- Enviar flancos activos en CLK para generar un 1 en Q y en QN un 0.

### Conteo de actividad

Considerar que en la figura 2 se considera los negados de CLK y PRE.

- Establecer LE=0, Contador=0.
- Establecer cada Contador=0 haciendo uso de un for.
- Establecer un ciclo finito de 5000 veces.
- Establecer A[3]=0;
- Esperar un delay y establecer A[0]=0, B[0]=0.
- Esperar un delay y establecer A[0]=1, B[0]=1.
- Esperar un delay y establecer A[2]=0, B[2]=0.
- Esperar un delay y establecer A[2]=1, B[2]=1.
- Esperar un delay y establecer A[3]=1, con este último el conteo de la actividad será impar.

## 4. Instrucciones de utilización de la simulación

Por cuestiones de orden se posee dos archivos probadores, uno para el comportamiento lógico y temporal de los componentes y el otro para el conteo de la actividad de cambio de estado de estas. Para correr las respuestas del comportamiento debe ingresar los siguientes comandos:

```
make
make gtk
```

Los anteriores comandos imprimirán el comportamiento lógico en la terminal. Considerar que estas salidas presentan un sesgo al trabajarse con delays, se trató a prueba y error tratar de mostrar los cambios de una manera evidente y limpia.

```
make all_act
make gtk
```

La respuesta del conteo de actividades se imprime en la terminal.

## 5. Ejemplos de los resultados

Las figuras 3, 4 y 5 muestran el comportamiento de las compuertas lógicas nand, nor y not respectivamente, contemplando delay en el momento del cambio de valor.

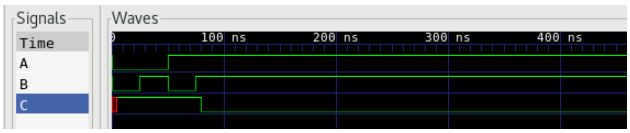


Figura 3: Comportamiento de la compuerta NAND

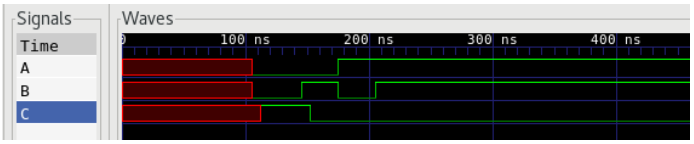


Figura 4: Comportamiento de la compuerta NOR

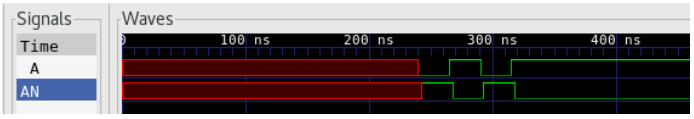


Figura 5: Comportamiento de la compuerta NOT

La siguiente figura muestra el comportamiento del mux, es evidente que cuando la señal SEL se encuentra en *bajo* la señal C sigue el valor de B, en caso contrario sigue el valor de A.

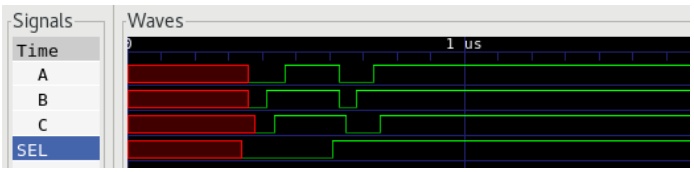


Figura 6: Comportamiento de un mux 2 a 1

La siguiente imagen muestra el comportamiento del flip flop tipo D. Vale mencionar que las señales PRE y CLR son los negados de las señales con el mismo nombre en el probador, ver en el código que estas se pasan por una compuerta *not*; por lo tanto, se puede verificar el comportamiento contra la figura 2.

La siguiente figura muestra el resultado de correr el comando *make all_act*.

## 6. Tiempo

| Actividad | Tiempo | Descripción |
|---|---|---|
| Buscar información y estudiarla | 40 minutos | Esto comprende buscar información de los delay. |
| Ejecución | 1 hora con 20 minutos | Programación por comportamiento |
| Ejecución | 2 hora con 25 minutos | Confección de pruebas y verificación de las mismas |
| Reporte y presentación | 2 horas con 35 minutos | |

Cuadro 1: Tabla de tiempo.

## 7. Conclusiones y recomendaciones

- Se logra programar con éxito el comportamiento una compuerta not, nand, nor; además de un mux de 2 a 1 como un flip flop tipo D.
- Se logra observar la respuesta de los componentes con valores típicos en retardos; para lo que se recomienda tomar en consideración estos valores al imprimir los cambios utilizando *monitor* para seleccionar un delay apropiado.
- Se considera una ventaja utilizar las compuertas predefinidas por verilog (not, nand, etc) en el momento de programar unas con valores de delay.
- Se logra realizar el conteo de la actividad de las compuertas lógicas utilizando código de instrumentación.