Laboratorio 5 plataformas abiertas

Mario Enrique Brenes Arroyo
Universidad de Costa Rica
Fecha: 20 de septiembre de 2024
Carnet: C11194
Grupo: 1
Curso: Circuitos Digitales II
Profesor(a): : Enrique Coen Alfaro

Índice

1. Introducción **1**

2. Proceso de Síntesis **2**
2.1. Descripción estructural genérica (RTLIL) 2
2.2. Comprovación de funcionamiento sintetizado 2

3. Observaciones **4**

4. modificaciones **4**

5. Número de componentes usados **5**

6. Retardo de propagación de las entradas hasta las salidas. **5**

7. Uso del Makefile **5**
7.1. Reglas Principales 6
7.2. Reglas de Limpieza 6
7.3. Notas 7

1. Introducción

Para esta tarea se hará una pequeña sintetis de nuestro trabajo de la tarea 1 para conocer las diferencias entre un modelo conductual y uno estructural

de un archivo verilog per viendo que son igualmente sintetizados para el mismo objetivo, en este caso un control de acceso vehicular.

2. Proceso de Síntesis

A continuación se describe cómo obtener una descripción estructural a partir del programa de síntesis Yosys, utilizando la librería de elementos lógicos proporcionada en clase para realizar el mapeo tecnológico correspondiente. En detalle, se deben completar las siguientes tareas:

2.1. Descripción estructural genérica (RTLIL)

con nuestro codigo primero ejecutaremos un archivo sintetizado de neustro programa $control - de - acceso.v$ pero antes vamos a hacer unas modificaciones a los parametros de la clave correcta del mismo programa para establecerlos como parametros locales.

```
module control_acceso (
    input wire llegado_vehiculo, paso_vehiculo, tercer_intento, boton_reset, clk, reset,
    input wire [15:0] clave_ingresada,
    output reg abriendo_compuerta, cerrando_compuerta, alarm_bloqueo, alarm_pin_incorrecto
);

reg [1:0] EstPresente, ProxEstado, contador_intentos;

// Definir CLAVE_CORRECTA como constante
localparam [15:0] CLAVE_CORRECTA = 16'h1194; // Clave correcta fija por numero de carnet c11194
```

Figura 1: Clave correcta como contante

Creamos nuestro archivo sintesis.ys parsintetizar nuestro programa de la tarea 1.

2.2. Comprovación de funcionamiento sintetizado

Verificamos que la descripción estructural genérica funciona usando la infraestructura de pruebas diseñada para la tarea 1.

Diagrama de flujo conductual

este es el diagrama conductual de la tarea 1.

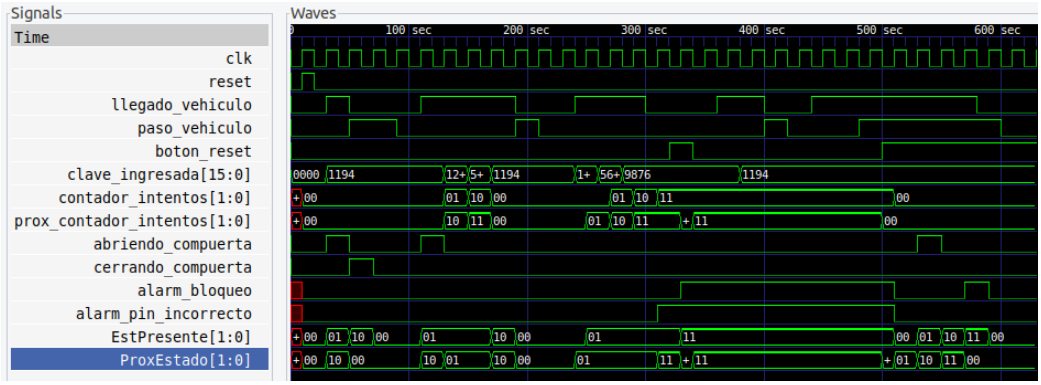

Figura 2: diagrama_c*onductual*

Diagrama de flujo rtlil

Este es el diagrama rtlil generado.

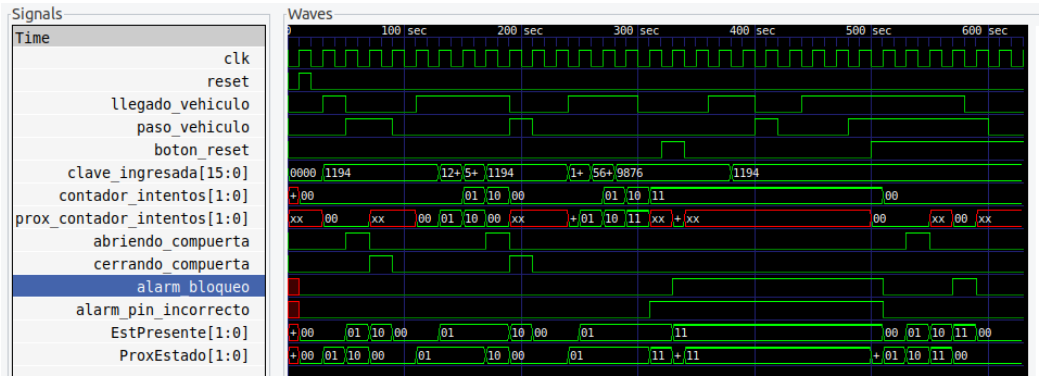

Figura 3: Diagrama_r*tlil*

Diagrama de flujo sintetizado

Este es el diagrama rtlil generado.

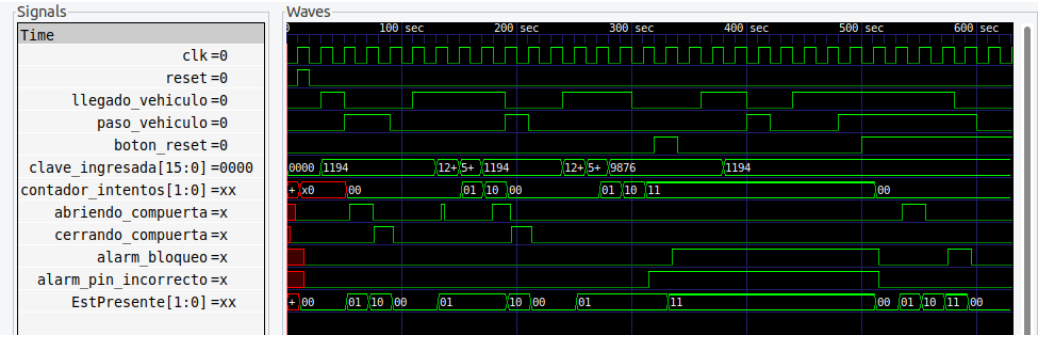

Figura 4: Diagrama$_s$*intetizado*

3. Observaciones

- El diagrama rtlil genera condiciones no iporta en el proximo contador de intentos cuando la contraseña es correcta.
- En los diagramas sintetizados se pueden ver los retardos en las salidas.
- En el diagrama sintetizado se abre por un momento pequeño la compuerta debido al retraso de 2 unidades de tiempo.
- Al parecer los diagramas sintetizados fueron mas eficientes a la hora de crear los diagramas de flujo de gtkwave que el mismo diagrama conductual

4. modificaciones

- Se ampliaron los rangos de las entradas de 10 a 20 unidades paa considerar bien los retardos
- Se pusieron relojes sincronicos con el reset y el clock para la sintetización porque si no se producian latches
- Se añadieron retardos en la libreria *cmos_cells.v* para adaptarlo a un sistema casi de la vida real al gusto.
- Se cambio el sistema de latches del contador de intentos por registros mas especializados.

5. Número de componentes usados

Nuestro archivo `control_de_accesso.v` sintetizado tuvo los siguientes resultados:

- NOR cells: 27
- NAND cells: 19
- NOT cells: 12
- DFF cells: 4

6. Retardo de propagación de las entradas hasta las salidas.

- **BUF**:
  - Retardo: 1
- **NOT**:
  - Retardo: 1
- **NAND**:
  - Retardo: 2
- **NOR**:
  - Retardo: 2
- **DFF**:
  - Retardo: 2

7. Uso del Makefile

El Makefile proporciona varias reglas para automatizar la simulación y limpieza de archivos. A continuación se describen las reglas disponibles:

7.1. Reglas Principales

- **conductual**: Compila y ejecuta la simulación conductual.

```
make conductual
```

Esto ejecuta los siguientes comandos:

```
iverilog testbench.v
vvp a.out
gtkwave resultados.vcd
```

- **rtlil**: Ejecuta la síntesis RTLIL y luego la simulación.

```
make rtlil
```

Esto ejecuta los siguientes comandos:

```
yosys sintesis_rtlil.ys
iverilog testbench_rtlil.v
vvp a.out
gtkwave resultados.vcd
```

- **synth**: Ejecuta la síntesis y luego la simulación sintetizada.

```
make synth
```

Esto ejecuta los siguientes comandos:

```
yosys sintetis.ys
iverilog testbench_synth.v
vvp a.out
gtkwave resultados.vcd
```

En caso de que quiera correr un diseño sintetizado sin retardo abra `testbench_synth.v` y cambie antes de ejecutar el make synth:

`include cmos_cells.v"` por `include cmos_cells_sin_retardo.v"`

7.2. Reglas de Limpieza

- **clean**: Elimina los archivos generados por las simulaciones.

```
make clean
```

Esto ejecuta los siguientes comandos:

```
    del a.out resultados.vcd || true
    rm -f a.out resultados.vcd
```

- **clean-rtlil**: Limpia los archivos generados por la simulación RTLIL.

```
    make clean-rtlil
```

Esto ejecuta los siguientes comandos:

```
    make clean
    del control_rtlil.v || true
    rm -f control_rtlil.v
```

- **clean-synth**: Limpia los archivos generados por la simulación sintetizada.

```
    make clean-synth
```

Esto ejecuta los siguientes comandos:

```
    make clean
    del control_synth.v || true
    rm -f control_synth.v
```

7.3. Notas

- Siempre después de ejecutar cualquier `make`, ejecutar el `make clean` correspondiente.
  - Para `make conductual` después ejecute `make clean`.
  - Para `make rtlil` después ejecute `make clean-rtlil`.
  - Para `make synth` después ejecute `make clean-synth`.
- Asegúrate de tener instalados `iverilog`, `vvp`, `gtkwave` y `yosys` para ejecutar las simulaciones y síntesis correctamente.
- Los comandos `del` y `rm` están incluidos para compatibilidad con Windows y Unix respectivamente.

Todo el trabajo lo puedes ver en el repositorio de github aquí