
Universidad de Buenos Aires
Facultad de Ingeniería

Especialización en Microelectrónica

Diseño Lógico

# Simulación y Diseño de una Máquina de Estados

*Detección de la palabra "casa"*

**Alumno:** Agustín Galdeman
**Fecha:** 14 de abril de 2025

Buenos Aires, Argentina

# Introducción

En este documento se presenta el diseño e implementación de una máquina de estados finita (FSM) en Verilog capaz de detectar cuántas veces aparece la palabra **"casa"** en un flujo secuencial de caracteres. La entrada está formada exclusivamente por los siguientes símbolos: las letras minúsculas del alfabeto español (incluyendo la letra "ñ") y el espacio vacío. Cada símbolo se representa como un número de 5 bits.

# Codificación de símbolos

Dado que hay 27 letras (de la "a" a la "z" incluyendo la "ñ") más el espacio, se utilizan 28 símbolos codificados con 5 bits. Una posible asignación es la siguiente:

| Símbolo | Valor (decimal) | Símbolo | Valor (decimal) |
|---|---|---|---|
| a | 1 | ñ | 15 |
| b | 2 | o | 16 |
| c | 3 | p | 17 |
| d | 4 | q | 18 |
| e | 5 | r | 19 |
| f | 6 | s | 20 |
| g | 7 | t | 21 |
| h | 8 | u | 22 |
| i | 9 | v | 23 |
| j | 10 | w | 24 |
| k | 11 | x | 25 |
| l | 12 | y | 26 |
| m | 13 | z | 27 |
| n | 14 | espacio | 0 |

## Definición de estados

La FSM tiene 5 estados definidos para reconocer la secuencia "casa":

- **IDLE (0):** Estado inicial.
- **S1 (1):** Se ha leído una `c`.
- **S2 (2):** Se ha leído `ca`.
- **S3 (3):** Se ha leído `cas`.
- **S4 (4):** Se ha leído `casa`, se incrementa el contador.

## Transiciones de estado

- Desde **IDLE**:
  - Si se lee `c`, se pasa a **S1**.
  - Otro símbolo: se permanece en **IDLE**.
- Desde **S1**:
  - Si se lee `a`, se pasa a **S2**.
  - Si se lee `c`, se permanece en **S1**.
  - Otro símbolo: se vuelve a **IDLE**.
- Desde **S2**:
  - Si se lee `s`, se pasa a **S3**.
  - Si se lee `c`, se vuelve a **S1**.
  - Otro símbolo: se vuelve a **IDLE**.
- Desde **S3**:
  - Si se lee `a`, se pasa a **S4**.
  - Si se lee `c`, se vuelve a **S1**.
  - Otro símbolo: se vuelve a **IDLE**.
- Desde **S4**:
  - Se incrementa el contador.
  - Si se lee `c`, se pasa a **S1**.
  - Otro símbolo: se vuelve a **IDLE**.

# Explicación del código

# Entradas y salidas

- `clk`: señal de reloj.
- `reset`: reinicia el estado de la máquina y el contador.
- `simbolo` [4:0]: símbolo de entrada (codificado).
- `contador` [31:0]: salida que contabiliza cuántas veces se detectó 'casa'.

# Estados definidos

Se utilizan cinco estados codificados con 3 bits:

- **IDLE (0)**: estado inicial.
- **S1 (1)**: se ha leído la letra 'c'.
- **S2 (2)**: se leyó 'ca'.
- **S3 (3)**: se ha leído 'cas'.
- **S4 (4)**: se leyó 'casa'.

# Codificación de letras

Para simplificar, se definen solo los símbolos necesarios:

- `A = 5'd1`
- `C = 5'd3`
- `S = 5'd20`

# Bloque secuencial

El siguiente bloque actualiza el estado y el contador:

```
always @(posedge clk or posedge reset) begin
    if (reset) begin
        estado <= IDLE;
        contador <= 0;
    end else begin
        estado <= proximo_estado;
        if (proximo_estado == S4 && estado != S4)
            contador <= contador + 1;
    end
end
```

**Explicación**:

- Si se activa `reset`, se reinicia el sistema.
- En cada flanco positivo de `clk`, se actualiza el estado.
- Si se entra en el estado `S4`, se incrementa el contador.

# Lógica combinacional de transición

Este bloque determina el próximo estado a partir del estado actual y el símbolo recibido:

```
always @(*) begin
    case (estado)
        IDLE:      proximo_estado = (simbolo == C) ? S1 : IDLE;
        S1:        proximo_estado = (simbolo == A) ? S2 :
                                    (simbolo == C) ? S1 : IDLE;
        S2:        proximo_estado = (simbolo == S) ? S3 :
                                    (simbolo == C) ? S1 : IDLE;
        S3:        proximo_estado = (simbolo == A) ? S4 :
                                    (simbolo == C) ? S1 : IDLE;
        S4:        proximo_estado = (simbolo == C) ? S1 : IDLE;
        default:   proximo_estado = IDLE;
    endcase
end
```

**Descripción de transiciones**:

- Si el símbolo coincide con la siguiente letra esperada, se avanza.
- Si se recibe una 'c' fuera de lugar, se reinicia desde `S1`.
- Cualquier otro símbolo no válido reinicia al estado `IDLE`.

# Simulación funcional de la máquina de estados

Para verificar el funcionamiento del módulo `detector_casa`, se realizó una simulación temporal del sistema utilizando la librería GTKWave de Python. Se graficó el comportamiento mediante matplotLib.

## Secuencia de entrada

Se utilizó la siguiente secuencia codificada:

`c a s a  c a s a  x x  c a s a`

## Estrategia de simulación

1. Cada letra fue representada por un valor de 5 bits:
   - `c` = 3, `a` = 1, `s` = 20, `x` = 25
2. Se programó la lógica de transición de la FSM manualmente:
   - Si el símbolo era el esperado, se avanzaba al siguiente estado.
   - Si se alcanzaba el estado `S4`, se incrementaba el contador.
3. Se graficaron tres señales clave:
   - El símbolo leído en cada ciclo.
   - El estado actual de la FSM.
   - El valor acumulado del contador.

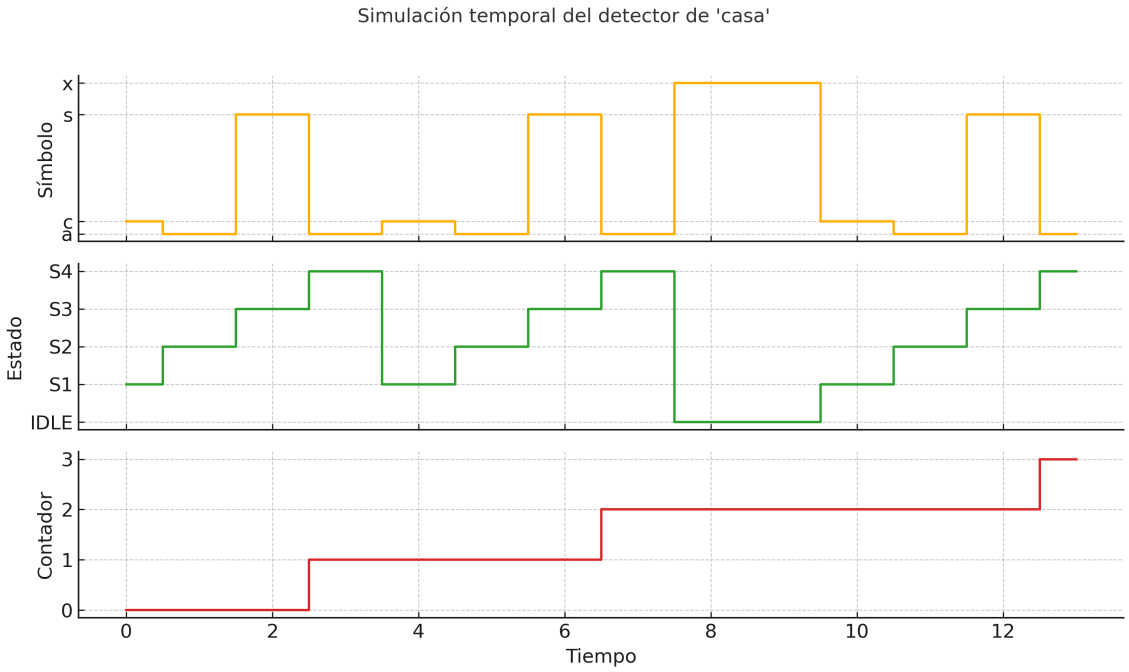


Figura 1: Simulación funcional del detector de la palabra “casa”.

# Simulación de señales

Por último se adjunta el resultado de la simulación obtenida mediante el testbench entregado. Se utilizó EDAPlayground con Icarus Verilog 12.0 y la siguiente opción de compilador:

$$-Wall - g2012$$

para compilar y simular el módulo. Para su visualización se usó EPWave.

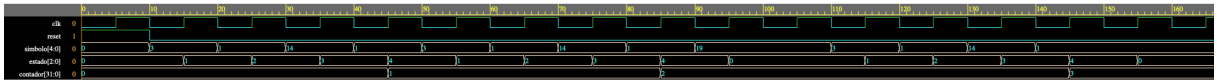


Figura 2: Simulación de señales del módulo.