LAB: Cerrojo

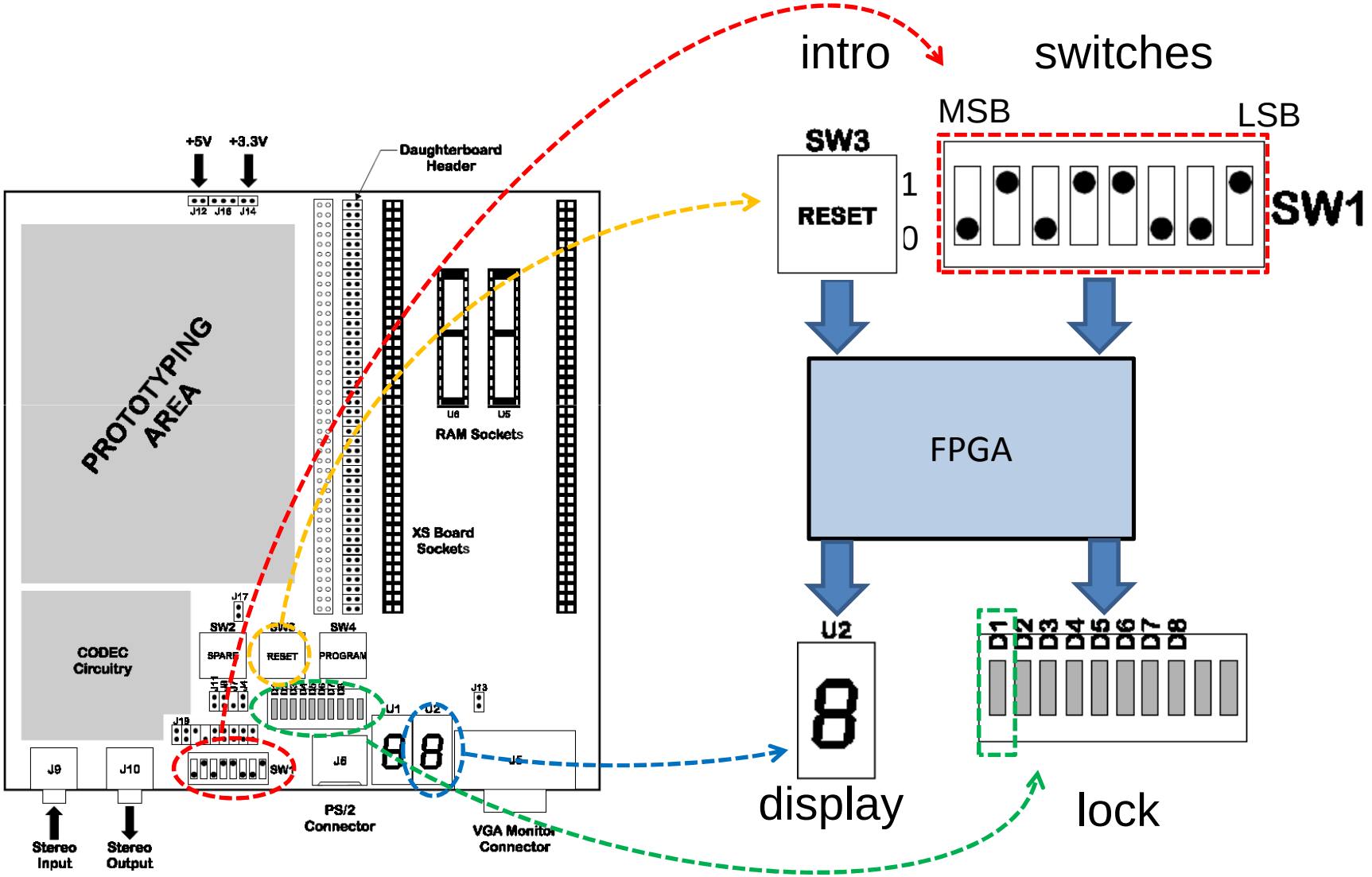
Carlos González Calvo
carlosgonzalez@fdi.ucm.es

Universidad Complutense Madrid

Objetivo

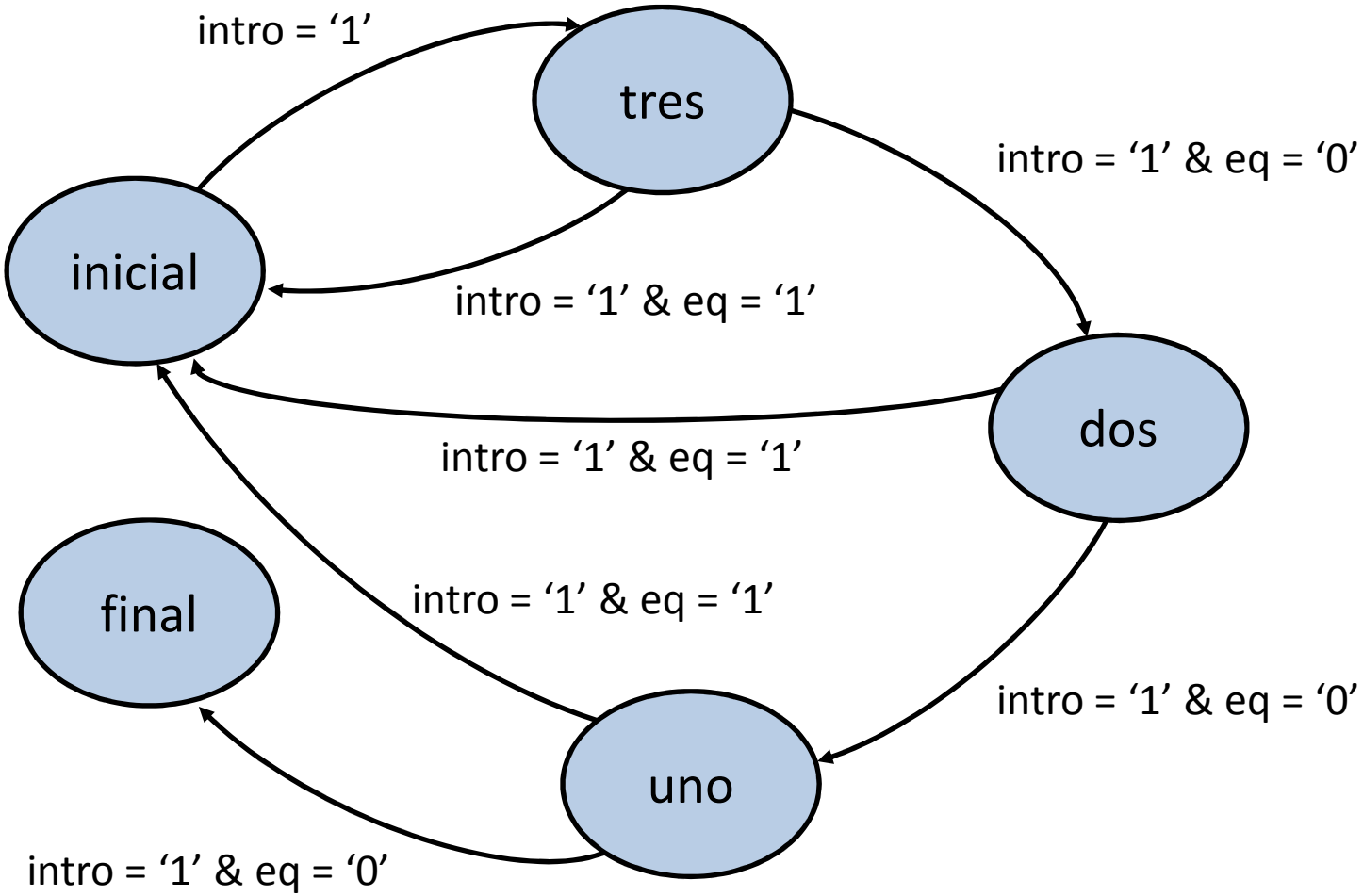
- Un cerrojo esta controlado por una clave de 8 bits. Inicialmente el cerrojo se encontrará abierto y al presionar un botón se almacenará la clave. A partir de entonces, se disponen de tres intentos (pulsando nuevamente el botón) para descubrir la clave, quedando para siempre bloqueado el cerrojo si se supera el número de intentos. Además, se debe mostrar el número de intentos restantes.

Implementación



FSM

- Solo se muestran las condiciones en las que se produce una transición de estado



UCF

Nombre	Pin
switches<0>	P12
switches<1>	J1
switches<2>	H1
switches<3>	H3
switches<4>	G2
switches<5>	K15
switches<6>	K16
switches<7>	F15

Nombre	Pin
display<0>	R10
display<1>	P10
display<2>	M11
display<3>	M6
display<4>	N6
display<5>	T7
display<6>	R7

Nombre	Pin
clk	T9
rst	K4
intro	H4
lock	L5

Flujo de trabajo

1) Importar el archivo '*cerrojo.vhd*'
2) Crear/añadir los componentes que se instancian
   - Crear el testbench
   - Simular
3) Implementar (ucf)
4) Descargar a la placa

Solución de la FSM

- Definición de tipos

```
TYPE states IS (inicial,uno,dos,tres,final);
SIGNAL currentState, nextState: states;
```

- Definición de la FSM

```
stateGen:
PROCESS (currentState, intro, eq)
BEGIN
  nextState <= currentState;
  CASE currentState IS
    WHEN inicial =>
      st <= "100";
      ld <= '1';
      lock <= '0';
      IF (intro='1') THEN
        nextState <= tres;
      END IF;
    WHEN tres =>
      st <= "011";
      ld <= '0';
      lock <= '1';
      IF (intro='1' AND eq='1') THEN
        nextState <= inicial;
       ELSIF (intro='1' AND eq='0') THEN
        nextState <= dos;
      END IF;
    WHEN dos =>
      st <= "010";
      ld <= '0';
      lock <= '1';
      IF (intro='1' AND eq='1') THEN
        nextState <= inicial;
       ELSIF (intro='1' AND eq='0') THEN
        nextState <= uno;
      END IF;
    WHEN uno =>
      st <= "001";
      ld <= '0';
      lock <= '1';
      IF (intro='1' AND eq='1') THEN
        nextState <= inicial;
       ELSIF (intro='1' AND eq='0') THEN
        nextState <= final;
      END IF;
    WHEN final =>
      st <= "000";
      ld <= '0';
      lock <= '1';
  END CASE;
END PROCESS stateGen;

state:
PROCESS (rst, clk)
BEGIN
  IF (rst = '0') THEN
    currentState <= inicial;
  ELSIF (clk'EVENT AND clk='1') THEN
    currentState <= nextState;
  END IF;
END PROCESS state;
```

Conclusión

Hemos implementado una
máquina de estados más compleja