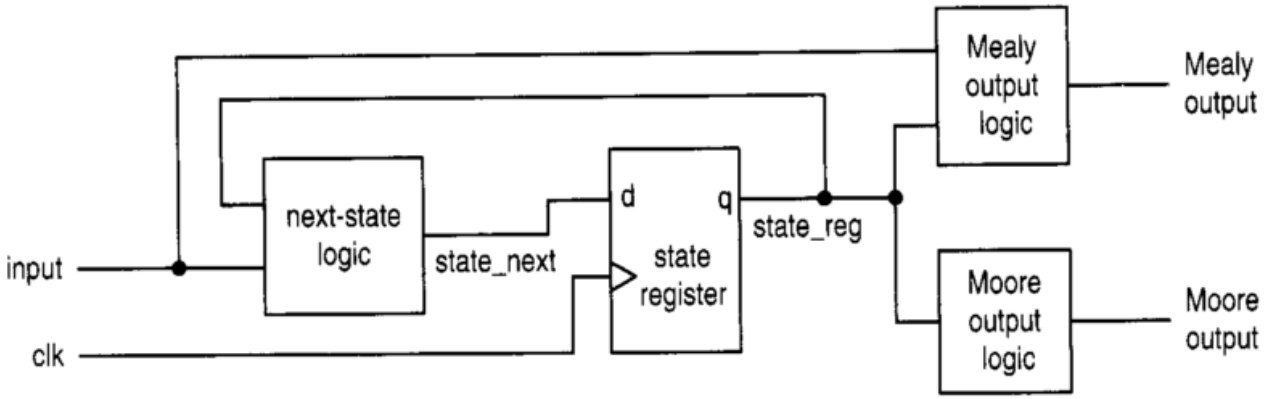
Circuitos Secuenciales en FPGA
Parte III: Finite State Machine con Datapath

Introducción a la Microfabricación y las FPGA

Instituto Balseiro

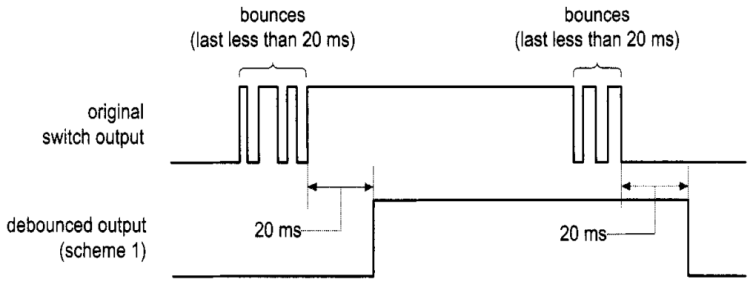
16 de Septiembre 2013

La clase pasada

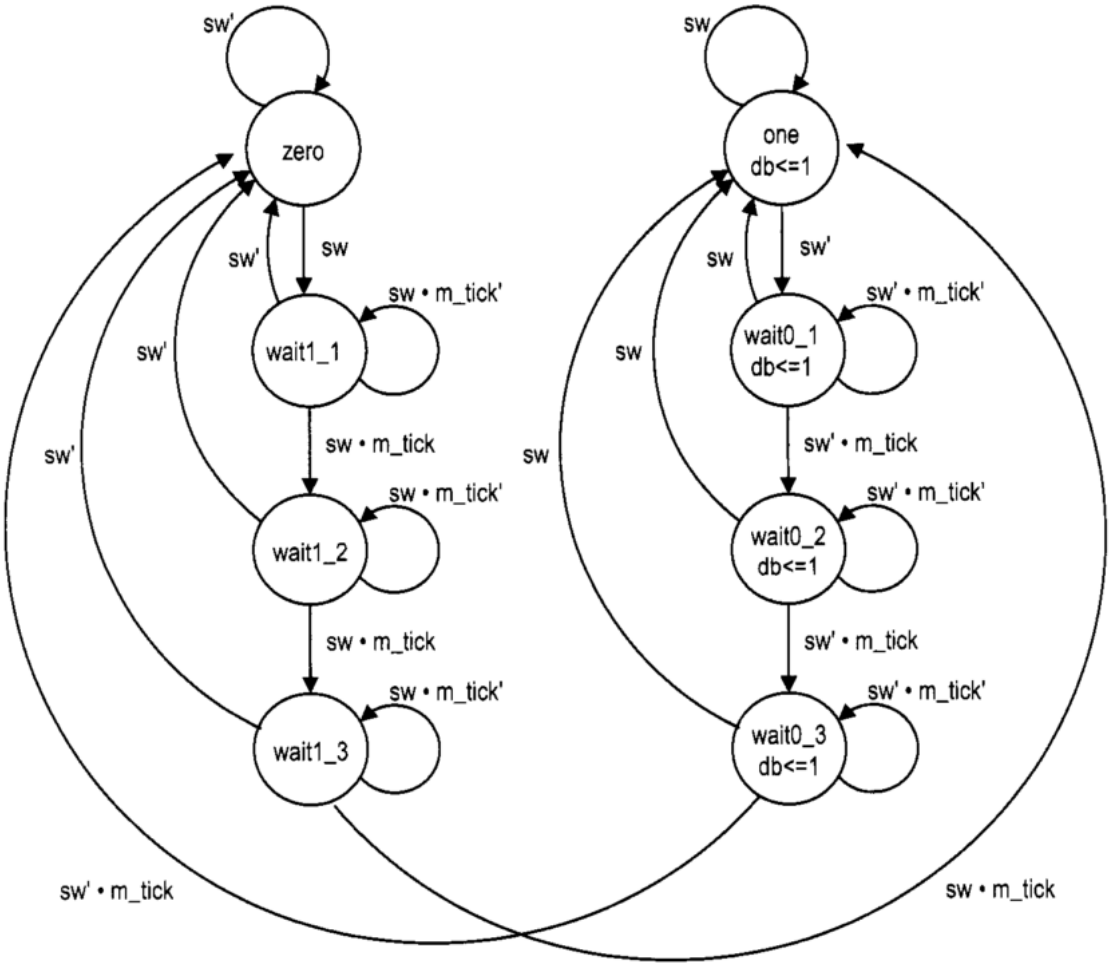


- FSM: Seguir el mismo método de diseño sincrónico, dividiendo claramente el problema en 4 pasos.
- Especificación con diagramas de FSM o ASM.
- Seguir guidelines de codificación para cada una de las cuatro partes.
- Diseño jerárquico.

Circuito de Estabilización



Contador free-running que genera `m_tick` cada 10 ms. Por lo menos 20 ms pero no sabes exactamente cuánto.



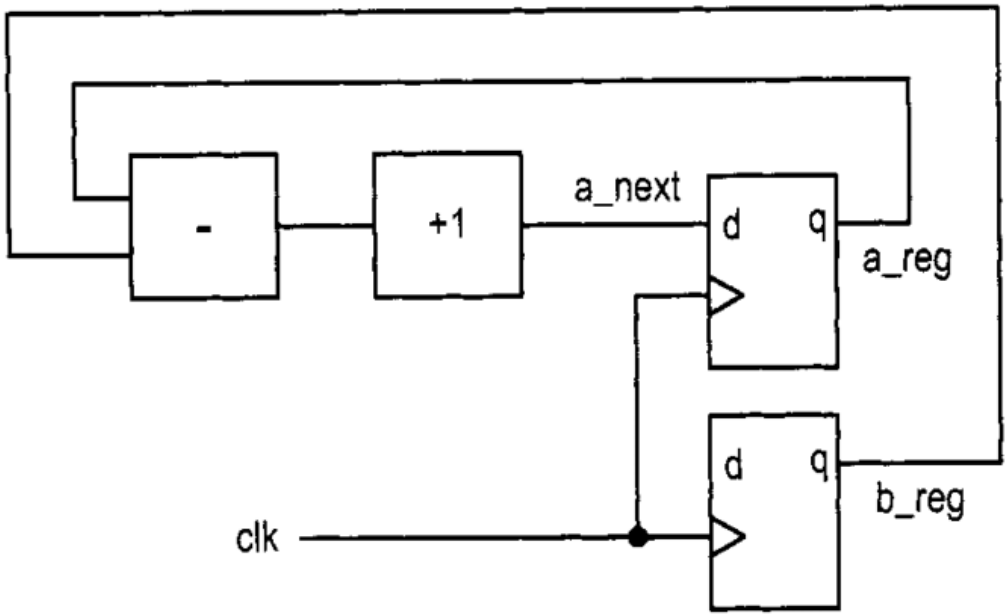
FSMD

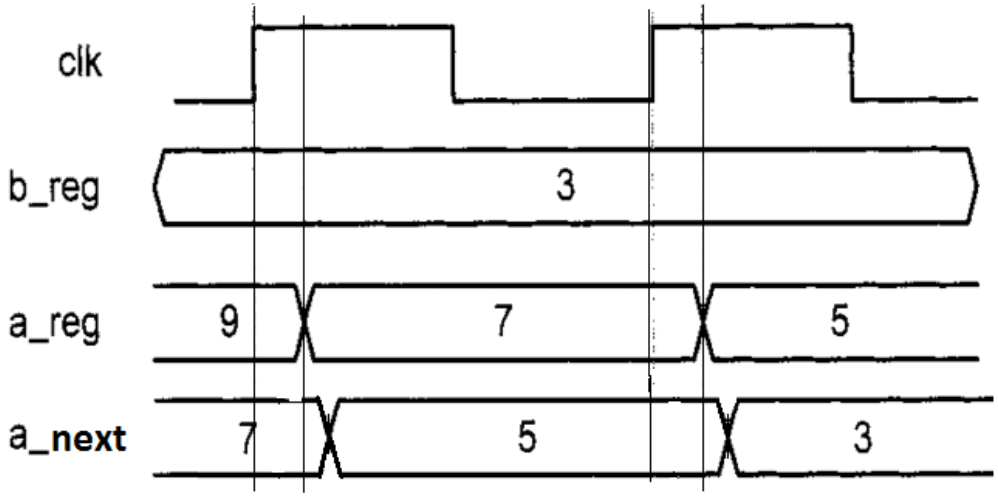
- Finite State Machine con Datapath. Combina una FSM y uno o varios circuitos regulares secuenciales.
  - *FSM o Control-path*: examina las entradas y el estado, y genera las señales de control para especificar las operaciones de los circuitos regulares secuenciales.
  - *Data path*: circuitos secuenciales regulares que hacen operaciones entre registros.
- Se usa para implementar sistemas descritos con la metodología de Register Transfer. Manipulaciones de datos entre registros.

Metodología Register Transfer

$r_{dest} \leftarrow f(r_{src1}, r_{src2}, ..., r_{srcn})$
Una operación RT puede implementarse con un circuito combinacional para $f()$, conectando la entrada y la salida de los registros.

Ejemplo
$a \leftarrow a - b + 1$
El resultado no se guarda en a hasta el proximo clk.
Está sincronizado.
$a_{next} \leftarrow a_{reg} - b_{reg} + 1$

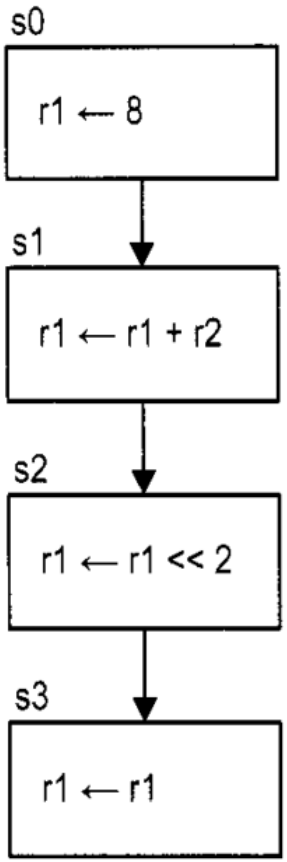


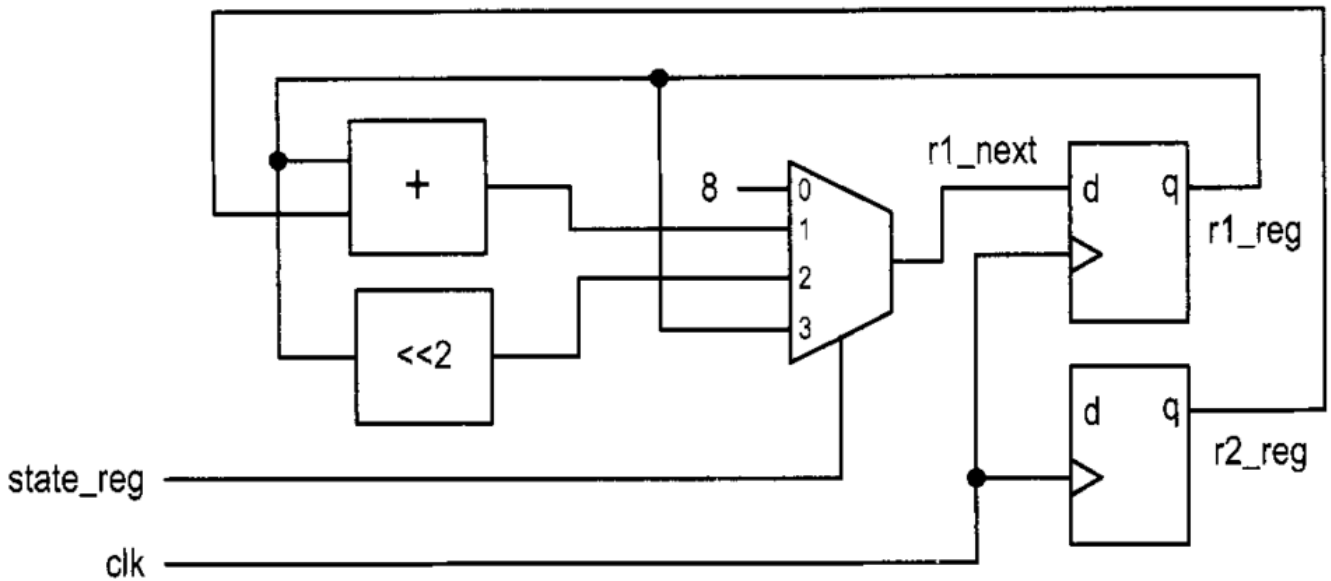


Representación: ASMD y FSMD

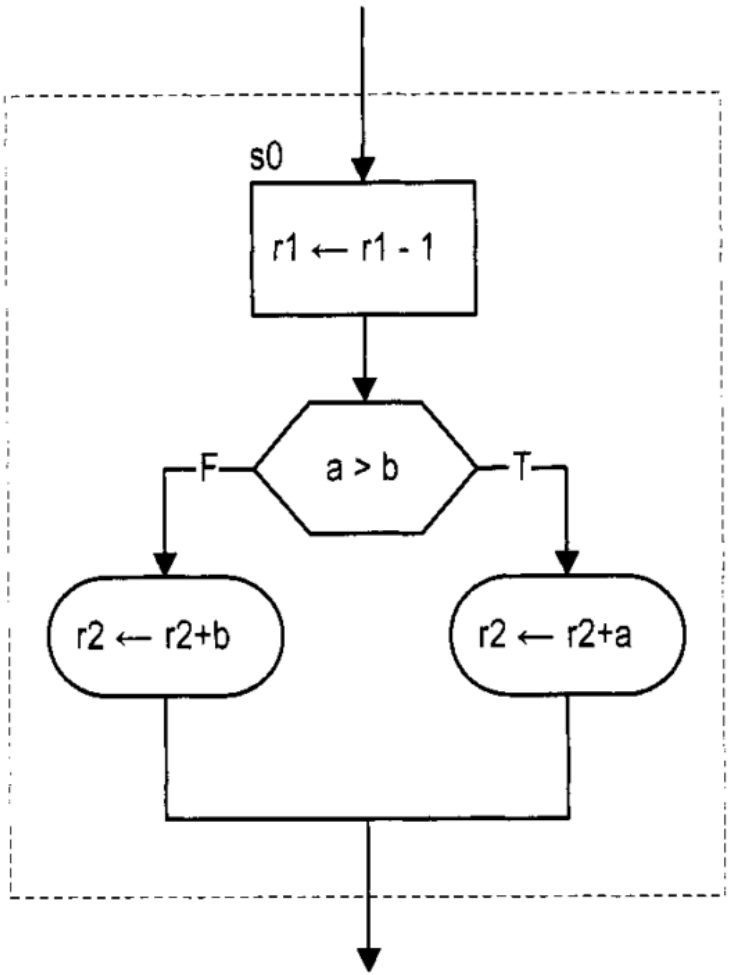
- Un circuito basado en la metodología RT tiene que especificar qué operaciones RT se van a hacer en cada paso (i.e, en cada estado).
- Diseño: extensión de ASM o FSM para incluir esta información. ASMD/FSMD: con Data Path.
- Las operaciones RT son tratadas como otro tipo de actividad, y se ponen donde van los outputs: o en los estados (Moore) o como dependiendo de los estados y las entradas (Mealy).

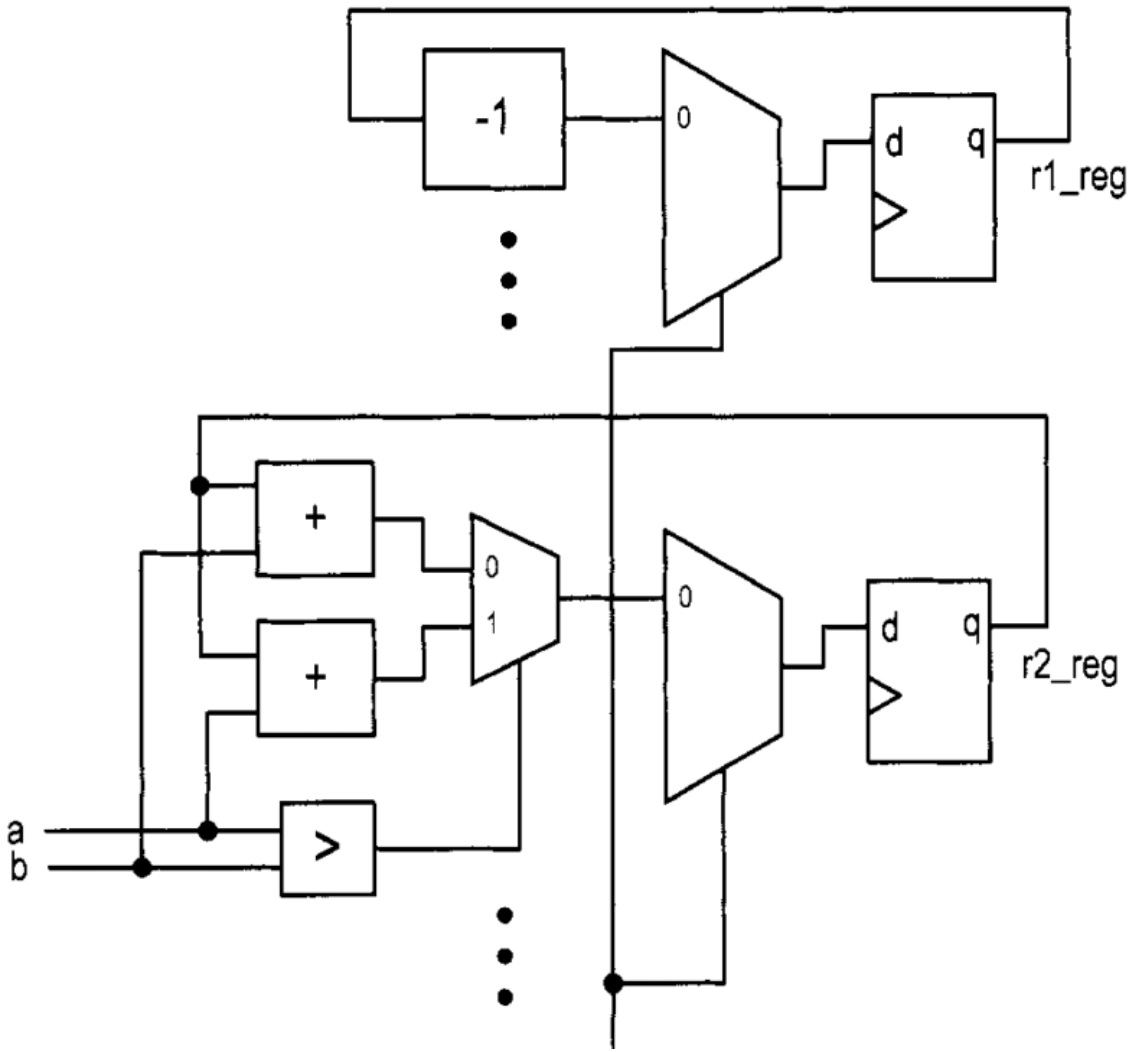
ASMD - Moore


s0
r1 ← 8
s1
r1 ← r1 + r2
s2
r1 ← r1 << 2
s3
r1 ← r1


+
<<2
8
0
1
2
3
r1_next
d
q
r1_reg
d
q
r2_reg
state_reg
clk

ASMD - Mealy





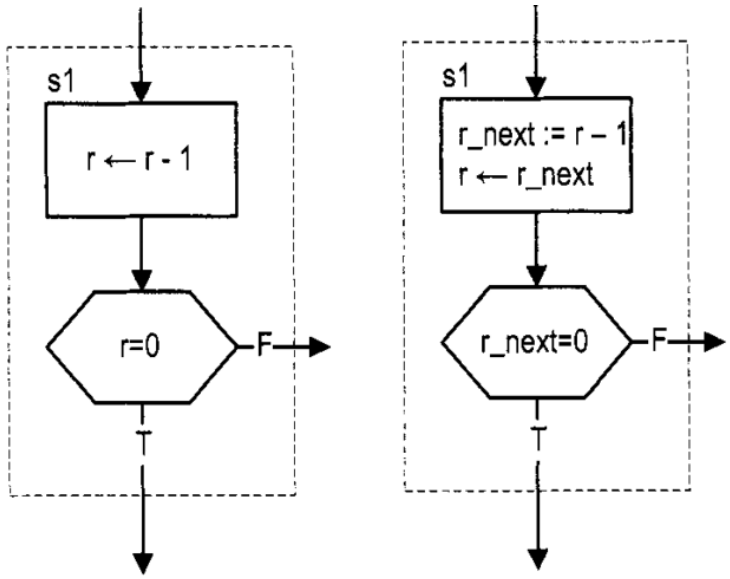
Diseño Sincrónico

Las operaciones RT en un ASMD están controladas por un clock, porque son transferencias entre registros. Entonces, el nuevo valor NO se actualiza hasta que se sale del bloque, i.e, se hace la trancision de estado.

$r \leftarrow r + 1$ es :

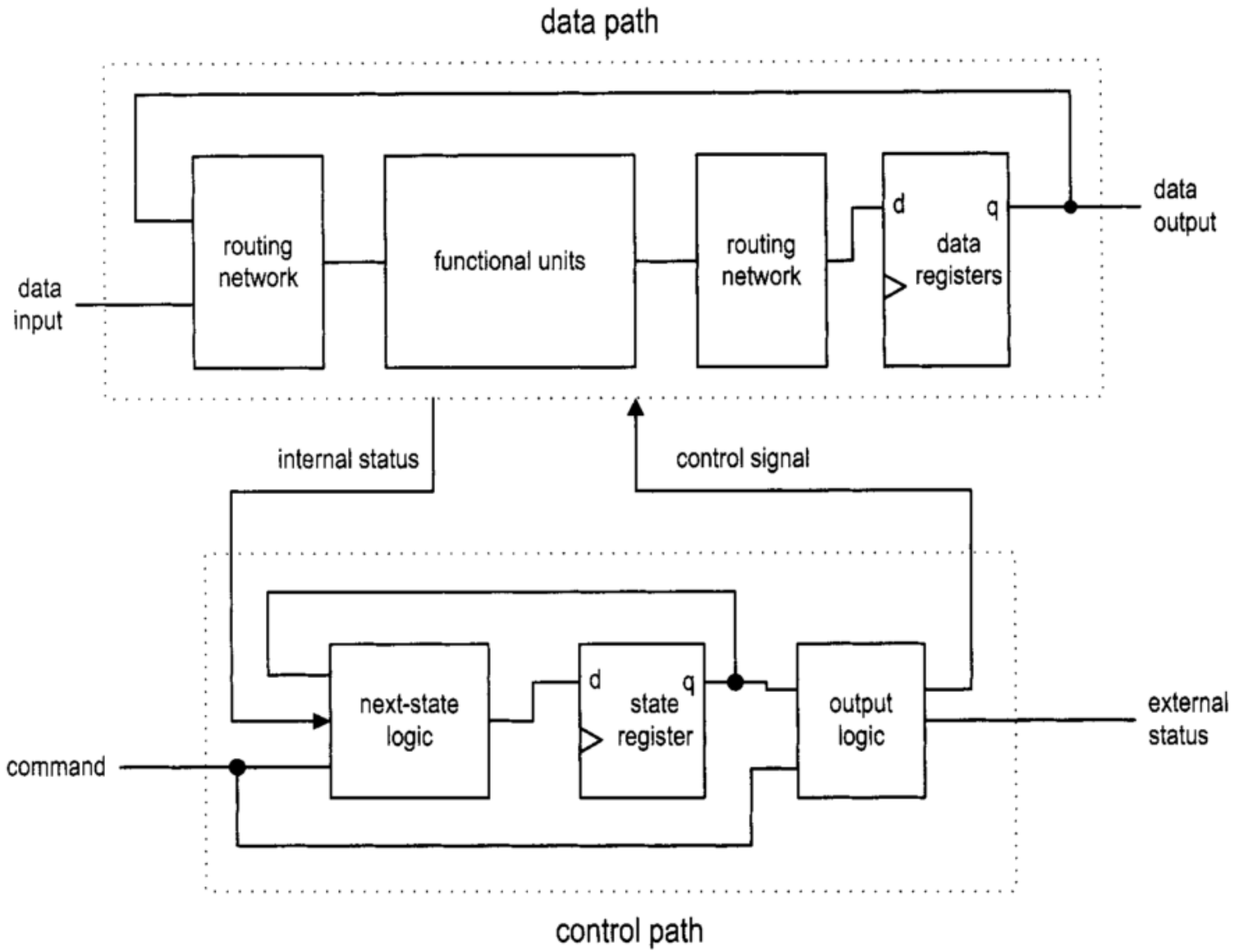
`r_next <= r_reg + 1`

`r_reg <= r_next` al momento de hacer la transición.



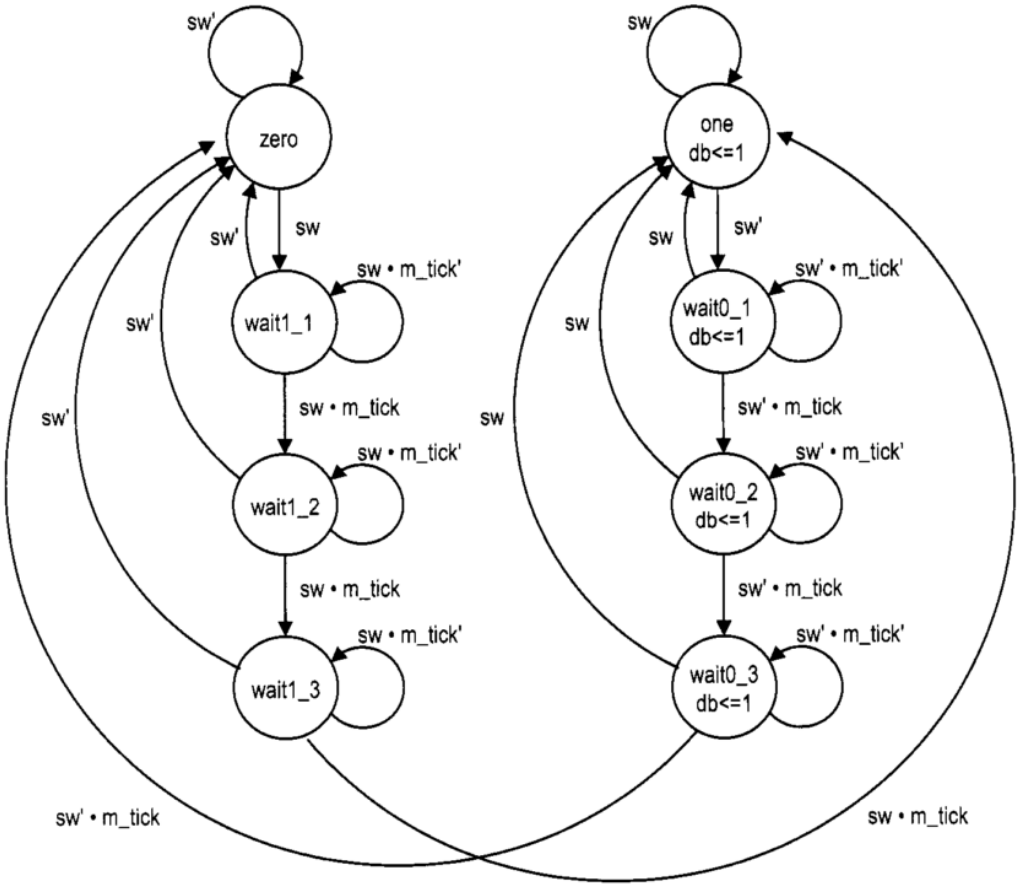
(a) Use old value of `r` (b) Use new value of `r`

Diagrama de Bloques



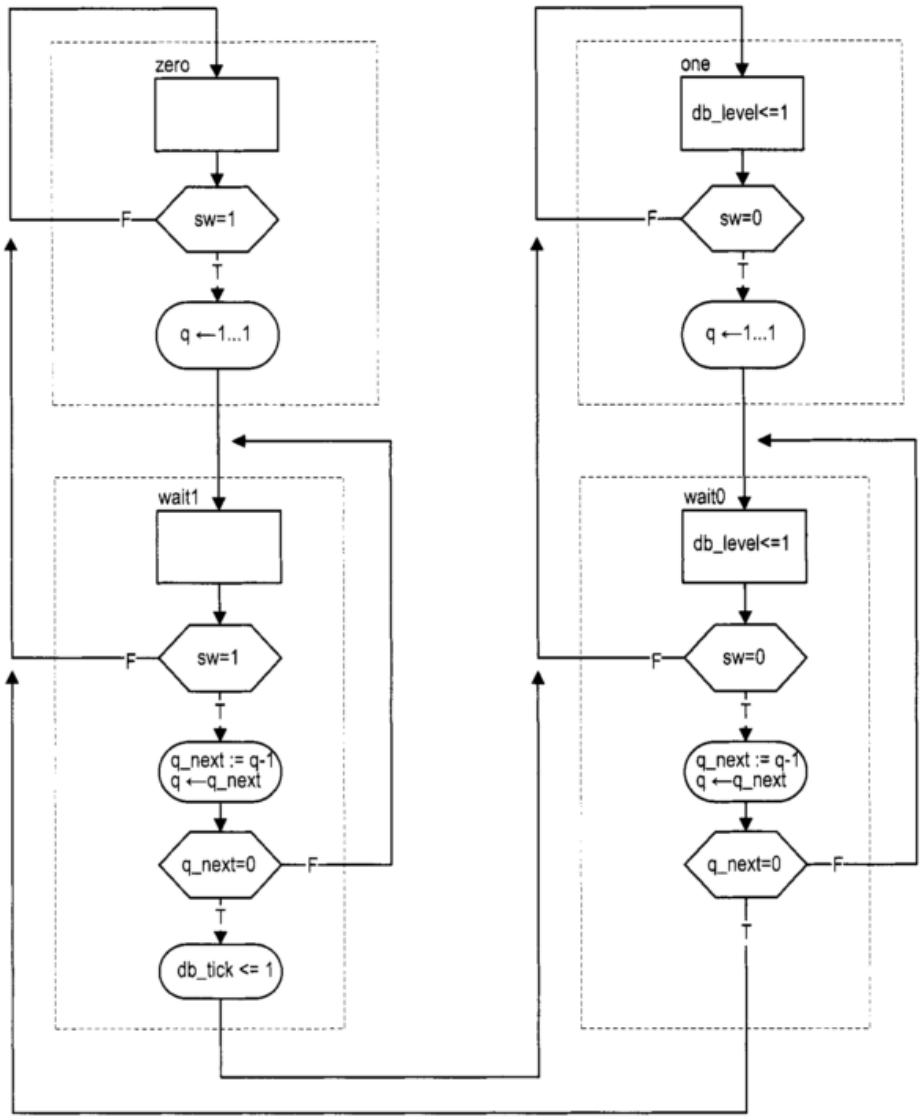
Debouncer

- Usa una FSM y un timer (sec. regular), pero no es metodología RT. La FSM no tiene ningún control del timer.
- Como en el estado `wait1_1` (idem `wait0_1`) la señal `m_tick` puede assertearse en cualqueir momento, no se puede medir exactamente cuánto tiempo pasa desde que se estabiliza la señal.
- Para solucionarlo: usar RT, que la FSM *controle* el timer.



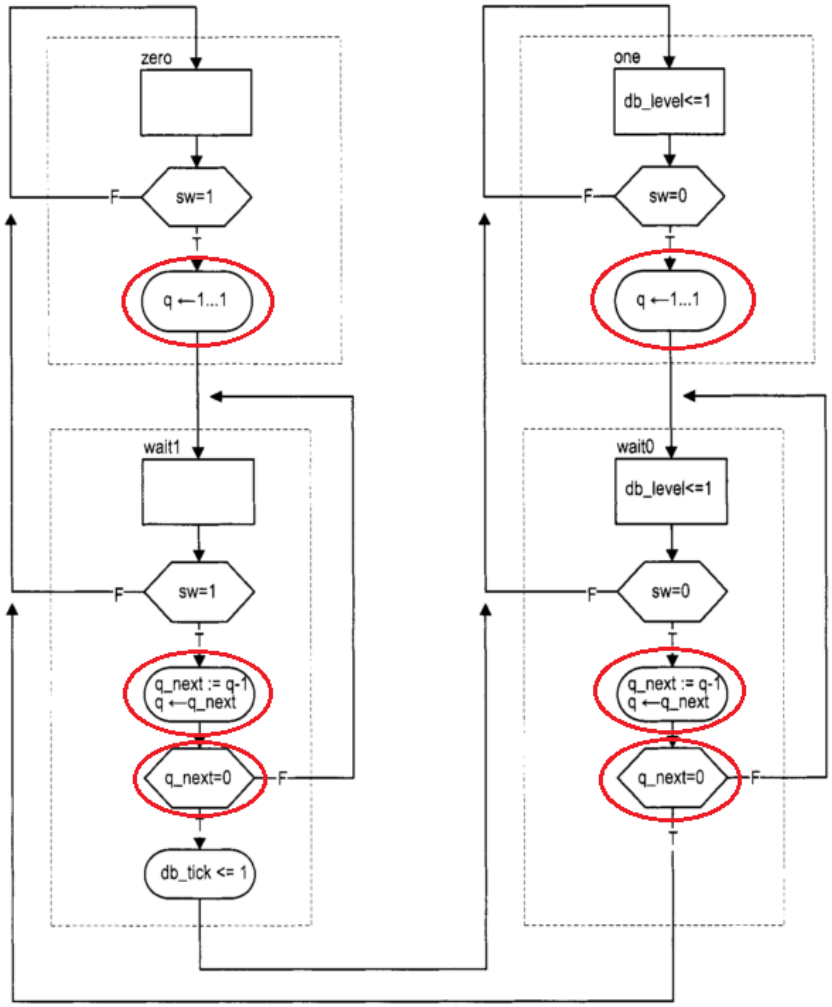
Diseño: FSMD

Diseño: ASMD


zero
sw=1
F
T
q ←1...1
wait1
sw=1
F
T
q_next := q-1
q ←q_next
q_next=0
F
T
db_tick <= 1
one
db_level<=1
sw=0
F
T
q ←1...1
wait0
db_level<=1
sw=0
F
T
q_next := q-1
q ←q_next
q_next=0
F
T

Diseño: Datapath en ASMD

- Identificar los componentes principales del DataPath y las señales que estado asociadas (que generan y que necesitan).
- Contador y:
  1. Cargarlo
  2. Decrementarlo
  3. Levantar una señal de estado cuando llega a cero.
- Entonces: contador con `q_load`, `q_dec` y que genera `q_zero`.



Código con Data Path explícito

.... ver en ISE ...

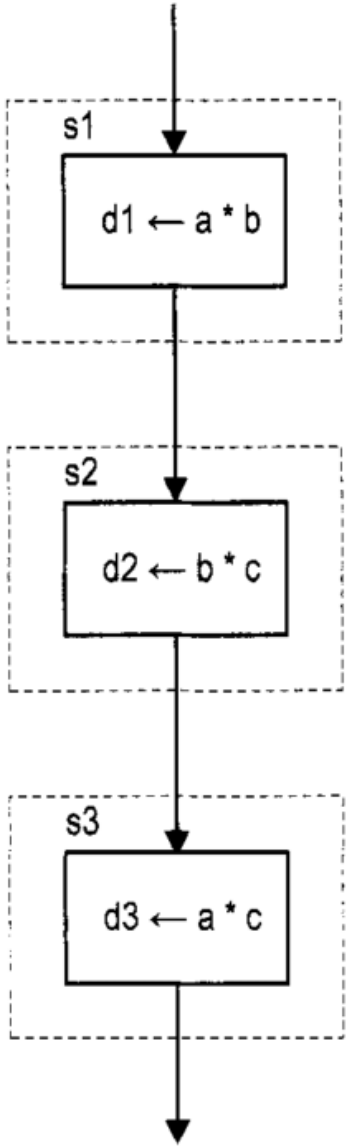
Código con Data Path implícito

```
architecture imp_fsmd_arch of debounce is
   constant N: integer:=21;  -- filter of 2^N * 20ns = 40ms
   type state_type is (zero, wait0, one, wait1);
   signal state_reg, state_next: state_type;
   signal q_reg, q_next: unsigned(N-1 downto 0);
begin
   -- FSMD state & data registers
   process(clk,reset)
   begin
      if reset='1' then
         state_reg <= zero;
         q_reg <= (others=>'0');
      elsif (clk'event and clk='1') then
         state_reg <= state_next;
         q_reg <= q_next;
      end if;
   end process;
```

```
-- next-state logic & data path func
process(state_reg,q_reg,sw,q_next)
begin
   state_next <= state_reg;
   q_next <= q_reg;
   db_tick <= '0';
   case state_reg is
      when zero =>
         db_level <= '0';
         if (sw='1') then
            state_next <= wait1;
            q_next <= (others=>'1');
         end if;
      when wait1=>
         db_level <= '0';
         if (sw='1') then
            q_next <= q_reg - 1;
            if (q_next=0) then
               state_next <= one;
               db_tick <= '1';
            end if;
         else -- sw='0'
            state_next <= zero;
         end if;
      when one =>
         db_level <= '1';
         if (sw='0') then
            state_next <= wait0;
            q_next <= (others=>'1')
         end if;
      when wait0=>
         db_level <= '1';
         if (sw='0') then
            q_next <= q_reg - 1;
            if (q_next=0) then
               state_next <= zero;
            end if;
         else -- sw='1'
```

Comparación

- El código con datapath implícito sigue mas cercanamente el ASMD/FSMD.
- Sin embargo, confiamos en que el soft de sintesis genere el datapath automáticamente, y por lo tanto, tenemos menos control.



Ejercicios mínimos

- Combinacional:
  - Dígito hexadecimal a 7-segmentos.
  - Sumador de signo y magnitud.
- Secuenciales Regulares:
  - Multiplexación en tiempo para mostrar 4 dígitos hexadecimales en el 7-segmentos.
  - Pulse Width Modulator (Ej 4.7.1 del Chino)
- FSM:
  - Estacionamiento completo.
- FSMD:
  - Fibonacci.
  - Binario a BCD.
  - Fibonacci con I/O BCD.
  - Auto-scaled low frequency counter.