```
// ------------------------

módulo shift_register #(
  parámetro norte = 10,
  parámetro M = 8
  ) (
  cable de entrada [N-1:0] din,
  cable de entrada,
  cable de entrada primero,
  habilitación del cable de entrada,
  carga del cable de entrada,
  valor del cable de entrada [N-1:0],
  cable de salida [N-1:0] salida
  );


  registro [N-1:0] sr [M-1:0];
  entero i;

  cable [N-1:0] sr_0;

  asignar sr_0 = sr[0];

  siempre @(posedge clk, posedge rst) comenzar

    si (primero) comienza
      para (i=0;i<M;i=i+1) comenzar
        Sr[i] <= {N{1'b0}};
      fin
    fin
    si no, empezar
      si (habilitar) comenzar
        si (carga) comienza
          para (i=0;i<M;i=i+1) comenzar
            sr[i] <= valor;
          fin
        fin
        si no, empezar
          sr[0] <= estruendo;
          para (i=1; i<M;i=i+1) comenzar
            señor[i] <= señor[i-1];
          fin
        fin
      fin
    fin
  fin

  asignar salida = sr[M-1];

módulo final


// ------------------------

módulo seq_det (
  cable de entrada,
  cable de entrada primero,
  cable de entrada dinar,
  detección de cable de salida
);


  parámetro local S3 = 2'b0;
  parámetro local S1 = 2'b01;
  parámetro local S2 = 2'b10;
  parámetro local S0 = 2'b11;

  estado de registro [1:0];
```

```
// -------------------------
// Uno caliente
// -------------------------
// parámetro local S3 = 4'b0001;
// parámetro local S1 = 4'b0010;
// parámetro local S2 = 4'b0100;
// parámetro local S0 = 4'b1000;

// registro [3:0] estado;
// -------------------------

siempre @(posedge clk, posedge rst) comenzar
  si (primero) comienza
    estado <= S0;
  fin
  si no, empezar
    caso (estado)
      S0: si (din) comienza
        estado <= S1;
      fin
      S1: si (din) comienza
        estado <= S2;
      fin
      si no, empezar
        estado <= S0;
      fin
      S2: si (!din) comienza
        estado <= S3;
      fin
      S3: si (din) comienza
        estado <= S1;
      fin
      si no, empezar
        estado <= S0;
      fin

    final
  fin

  asignar det = (estado==S3);

  // ----------------------
  // Otra opción
  // ----------------------
  // reg [1:0] siguiente_estado;
  //
  // siempre @(posedge clk, posedge rst) comienza
  // si (primero) comienza
  // estado <= S0;
  // fin
  // si no, comenzar
  // estado <= next_state;
  // fin
  // fin
  //
  // siempre @(*) comienza
  // caso(estado)
  // S0: si(din) bgin
  // estado_siguiente = S1;
  // fin
  // si no, comenzar
  // estado_siguiente = S0;
  // fin
  // S1: si (din) comienza
  // estado_siguiente = S2;
  // fin
  // si no, comenzar
  // estado_siguiente = S0;
  // fin
```

```
    // S2: si (!din) comienza
    // estado_siguiente = S3;
    // fin
    // si no, comenzar
    // estado_siguiente = S2;
    // fin
    // S3: si (din) comienza
    // estado_siguiente = S1;
    // fin
    // si no, comenzar
    // estado_siguiente = S0;
    // fin
    // caso final
    // fin
    // ---------------------

    // ---------------------
    // Otra opción
    // ---------------------
    // siempre @(*) comienza
    // estado_siguiente = estado;
    // caso(estado)
    // S0: si(din) bgin
    // estado_siguiente = S1;
    // fin
    // S1: si (din) comienza
    // estado_siguiente = S2;
    // fin
    // S2: si (!din) comienza
    // estado_siguiente = S3;
    // fin
    // S3: si (din) comienza
    // estado_siguiente = S1;
    // fin
    // caso final
    // fin
    // ---------------------

  módulo final
```