Example: Successive approximation analog to digital converter (ADC)

Description of operation

A successive approximation ADC works by using a digital to analog converter (DAC) and a comparator to perform a binary search to find the input voltage.

A sample and hold circuit (S&H) is used to sample the analog input voltage and hold (i.e. keep a non-changing copy) the sampled value whilst the binary search is performed.

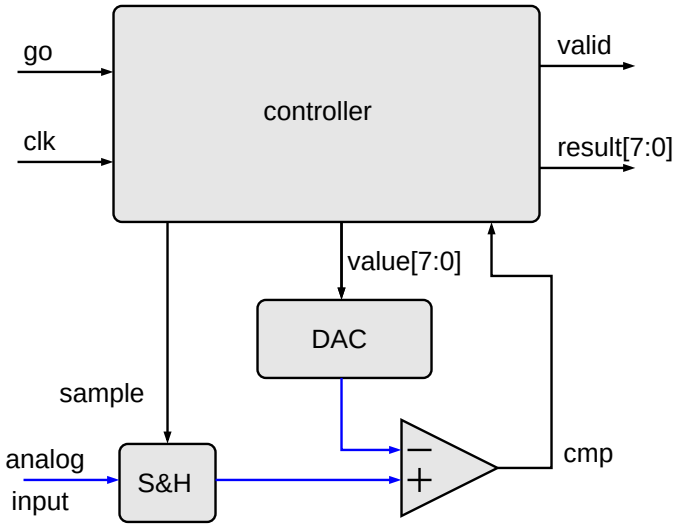
The binary search starts with the most significant bit (MSB) and works towards the least significant bit (LSB). For a 8-bit output resolution, 8 comparisons are needed in the binary search, taking a least 8 clock cycles.

The sample and hold circuit samples the analog input on a rising edge of the `sample` signal. The comparator output `cmp` is a logic 1 if the sampled analog voltage is greater than the output of the DAC, 0 otherwise.

The circuit in this example has two control signals `go` and `done`. At any point if `go` is 0 the circuit is reset. When `go` becomes a 1 the process of sampling and converting takes place. When a conversion if finished `valid` is set to 1 and the result is available.

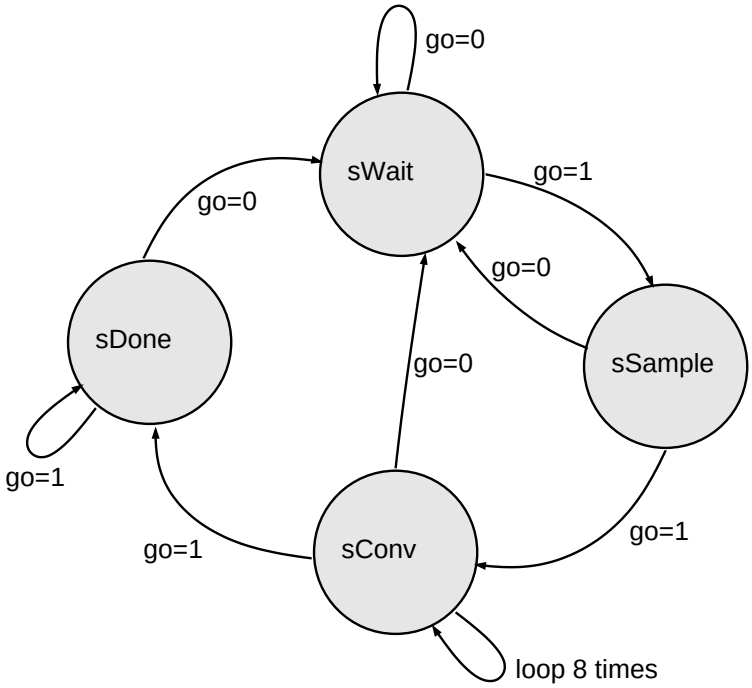
Of relevance to the ECAD course is the state machine controlling this process.

Block diagram



(analog signals are shown in blue)

State diagram



Verilog description

```
// ADC controller
module controller(clk,go,valid,result, sample,value,cmp);
  input        clk; // clock input
  input        go; // go=1 to perform conversion
  output       valid; // valid=1 when conversion finished
  output [7:0] result; // 8 bit result output
  output       sample; // to S&H circuit
  output [7:0] value; // to DAC
  input        cmp; // from comparitor

  reg    [1:0] state; // current state in state machine
  reg    [7:0] mask; // bit to test in binary search
  reg    [7:0] result; // hold partially converted result

  // state assignment
  parameter    sWait=0, sSample=1, sConv=2, sDone=3;

  // synchronous design
  always @(posedge clk) begin

    if (!go) state <= sWait; // stop and reset if go=0
    else case (state)           // choose next state in state machine
      sWait   : state <= sSample;
      sSample :
        begin                     // start new conversion so
          state  <= sConv;        // enter convert state next
          mask   <= 8'b10000000;  // reset mask to MSB only
          result <= 8'b0;         // clear result
        end
      sConv   :
        begin
          // set bit if comparitor indicates input larger than
          // value currently under consideration, else leave bit clear
          if (cmp) result <= result | mask;
          // shift mask to try next bit next time
          mask <= mask>>1;
          // finished once LSB has been done
          if (mask[0]) state <= sDone;
        end
      sDone   :;
    endcase
  end

  assign sample = state==sSample;   // drive sample and hold
  assign value  = result | mask;    // (result so far) OR (bit to try)
  assign valid  = state==sDone;     // indicate when finished
endmodule
```

Example of Verilog simulation testbench

(Note: This simulation code requires a simulator which supports such features.)

```
module testbench();

  // registers to hold inputs to circuit under test, wires for outputs
  reg clk,go;
  wire valid,sample,cmp;
  wire [7:0] result;
  wire [7:0] value;

  // instance controller circuit
  controller c(clk,go,valid,result, sample,value,cmp);

  // generate a clock with period of 20 time units
  always begin
    #10;
    clk=~clk;
  end
  initial  clk=0;

  // simulate analogue circuit with a digital model
  reg [7:0] hold;
  always @(posedge sample) hold = 8'b01000110;
  assign cmp = ( hold >= value);

  // monitor some signals and provide input stimuli
  initial begin
    $monitor($time, " go=%b valid=%b result=%b sample=%b value=%b cmp=%b
                     state=%b mask=%b",
                     go,valid,result,sample,value,cmp,c.state,c.mask);
    #100; go=0;
    #100; go=1;
    #5000; go=0;
    #5000; go=1;
    #40;   go=0;
    #5000;
    $stop;
  end

endmodule
```

Verilog simulation output

Note the use of x for undefined values.

```
    0 go=x valid=x result=xxxxxxxx sample=x value=xxxxxxxx
      cmp=x state=xx mask=xxxxxxxx
  100 go=0 valid=x result=xxxxxxxx sample=x value=xxxxxxxx
      cmp=x state=xx mask=xxxxxxxx
  110 go=0 valid=0 result=xxxxxxxx sample=0 value=xxxxxxxx
      cmp=x state=00 mask=xxxxxxxx
  200 go=1 valid=0 result=xxxxxxxx sample=0 value=xxxxxxxx
      cmp=x state=00 mask=xxxxxxxx
  210 go=1 valid=0 result=xxxxxxxx sample=1 value=xxxxxxxx
      cmp=x state=01 mask=xxxxxxxx
  230 go=1 valid=0 result=00000000 sample=0 value=10000000
      cmp=0 state=10 mask=10000000
  250 go=1 valid=0 result=00000000 sample=0 value=01000000
      cmp=1 state=10 mask=01000000
  270 go=1 valid=0 result=01000000 sample=0 value=01100000
      cmp=0 state=10 mask=00100000
  290 go=1 valid=0 result=01000000 sample=0 value=01010000
      cmp=0 state=10 mask=00010000
  310 go=1 valid=0 result=01000000 sample=0 value=01001000
      cmp=0 state=10 mask=00001000
  330 go=1 valid=0 result=01000000 sample=0 value=01000100
      cmp=1 state=10 mask=00000100
  350 go=1 valid=0 result=01000100 sample=0 value=01000110
      cmp=1 state=10 mask=00000010
  370 go=1 valid=0 result=01000110 sample=0 value=01000111
      cmp=0 state=10 mask=00000001
  390 go=1 valid=1 result=01000110 sample=0 value=01000110
      cmp=1 state=11 mask=00000000
 5200 go=0 valid=1 result=01000110 sample=0 value=01000110
      cmp=1 state=11 mask=00000000
 5210 go=0 valid=0 result=01000110 sample=0 value=01000110
      cmp=1 state=00 mask=00000000
10200 go=1 valid=0 result=01000110 sample=0 value=01000110
      cmp=1 state=00 mask=00000000
10210 go=1 valid=0 result=01000110 sample=1 value=01000110
      cmp=1 state=01 mask=00000000
10230 go=1 valid=0 result=00000000 sample=0 value=10000000
      cmp=0 state=10 mask=10000000
10240 go=0 valid=0 result=00000000 sample=0 value=10000000
      cmp=0 state=10 mask=10000000
10250 go=0 valid=0 result=00000000 sample=0 value=10000000
      cmp=0 state=00 mask=10000000
```

Note that at time 390 the result output is the same as the value loaded into the sample and hold model in the test bench circuit.