Mastering Digital Design

with Verilog on FPGAs

John Wickerson

Lecture 4

This lecture

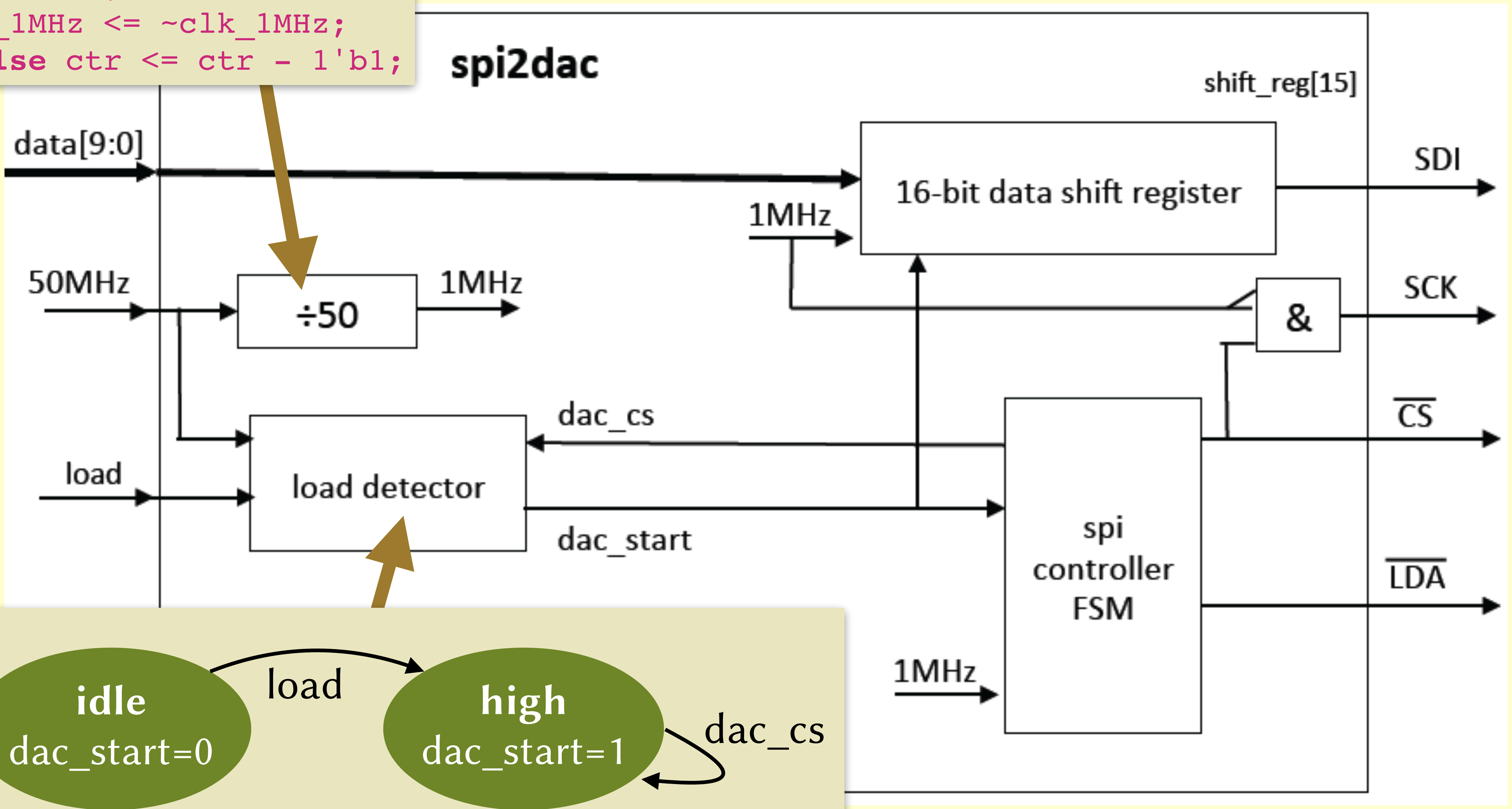
- The design of the spi2dac module
- How the ADC works
- Wrapping up

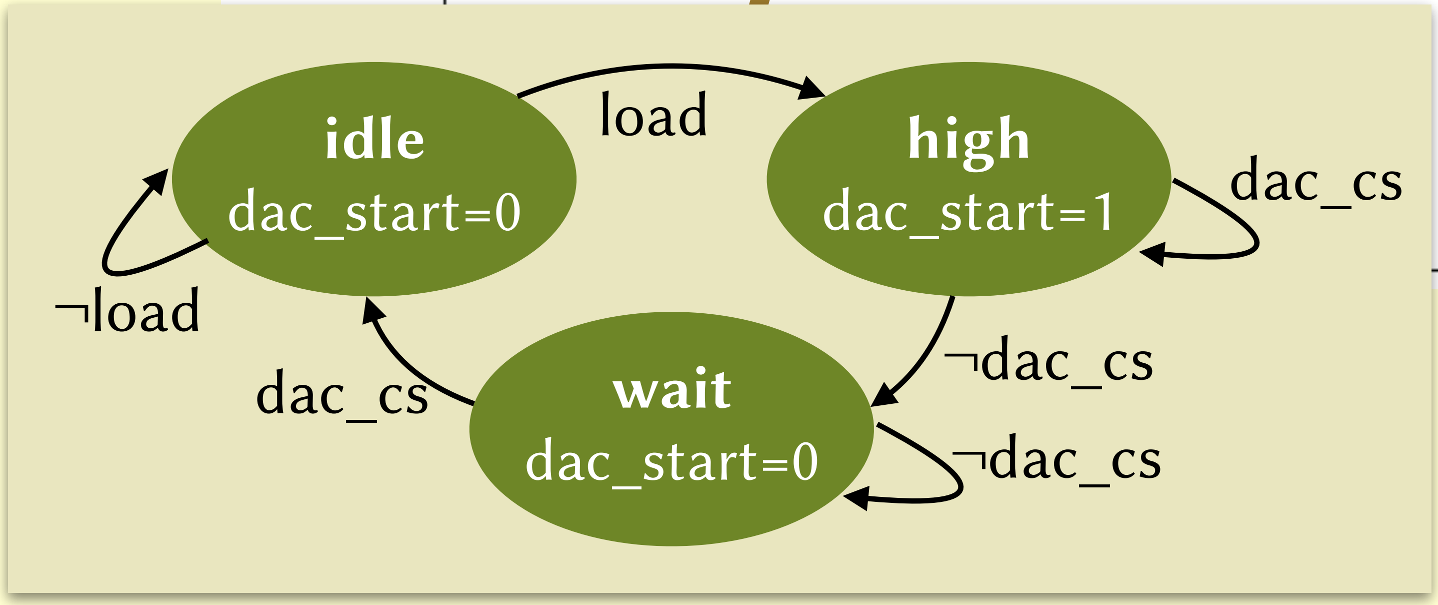
This lecture

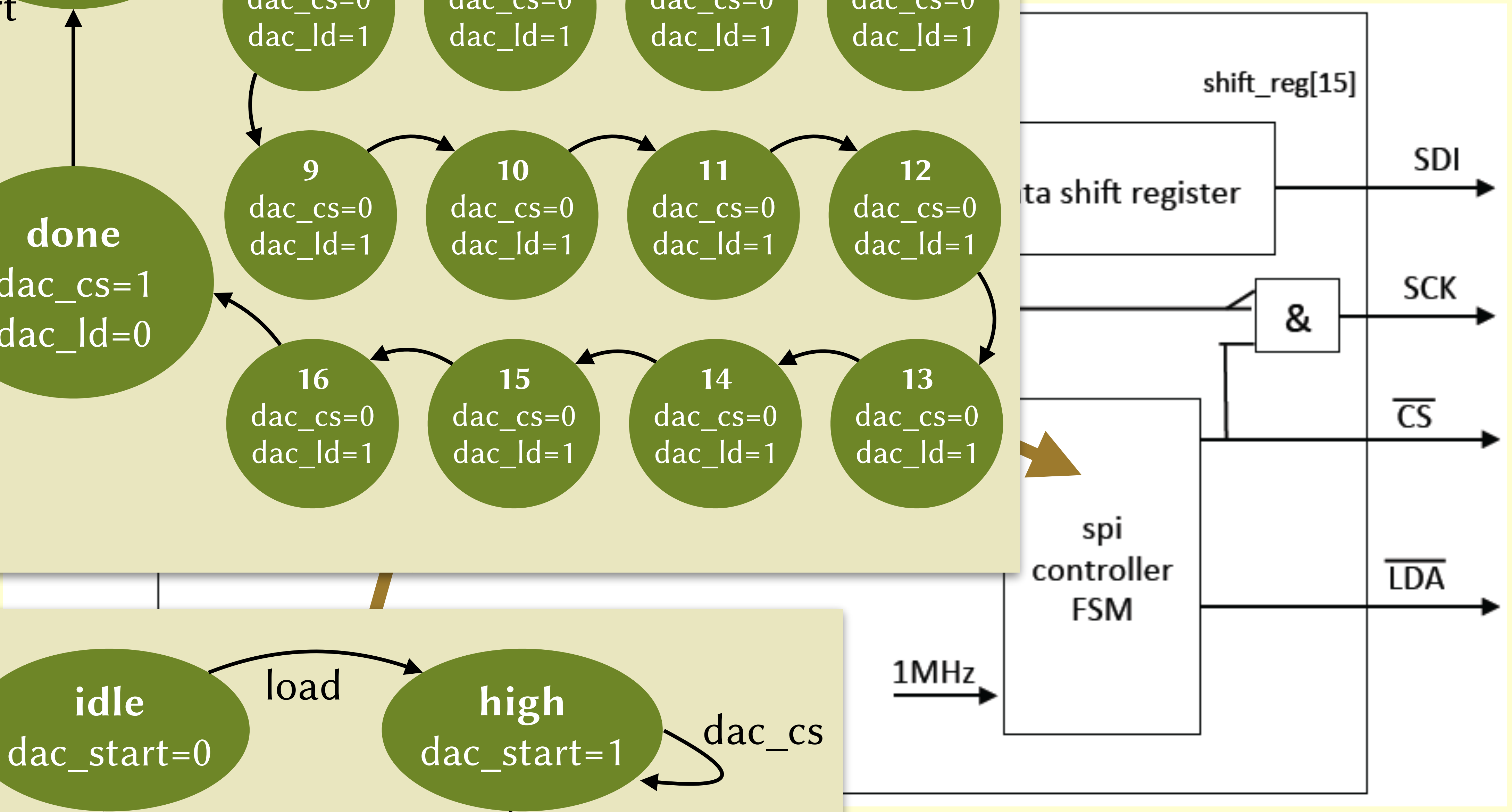
- **The design of the spi2dac module**
- How the ADC works
- Wrapping up

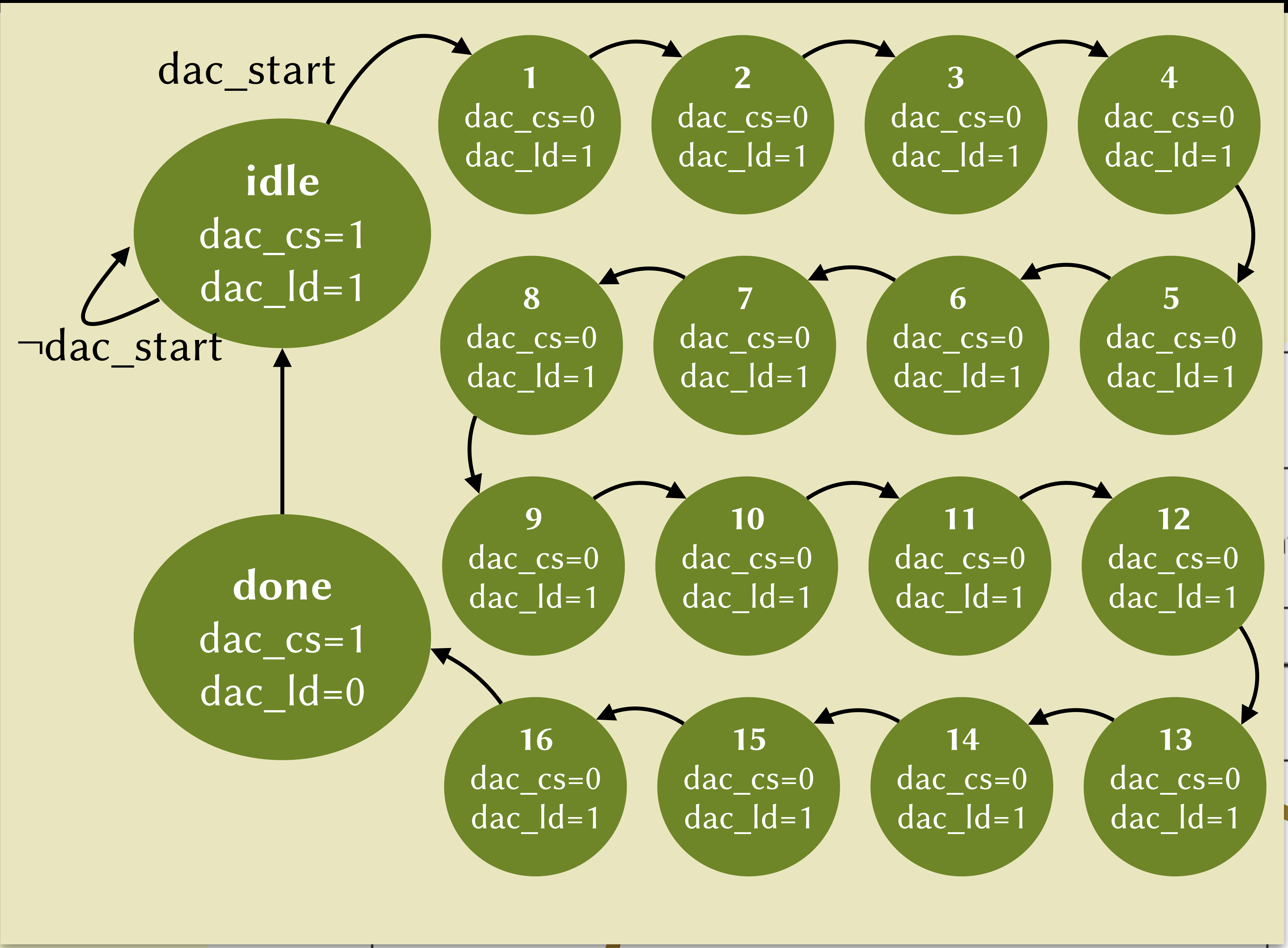
spi2dac

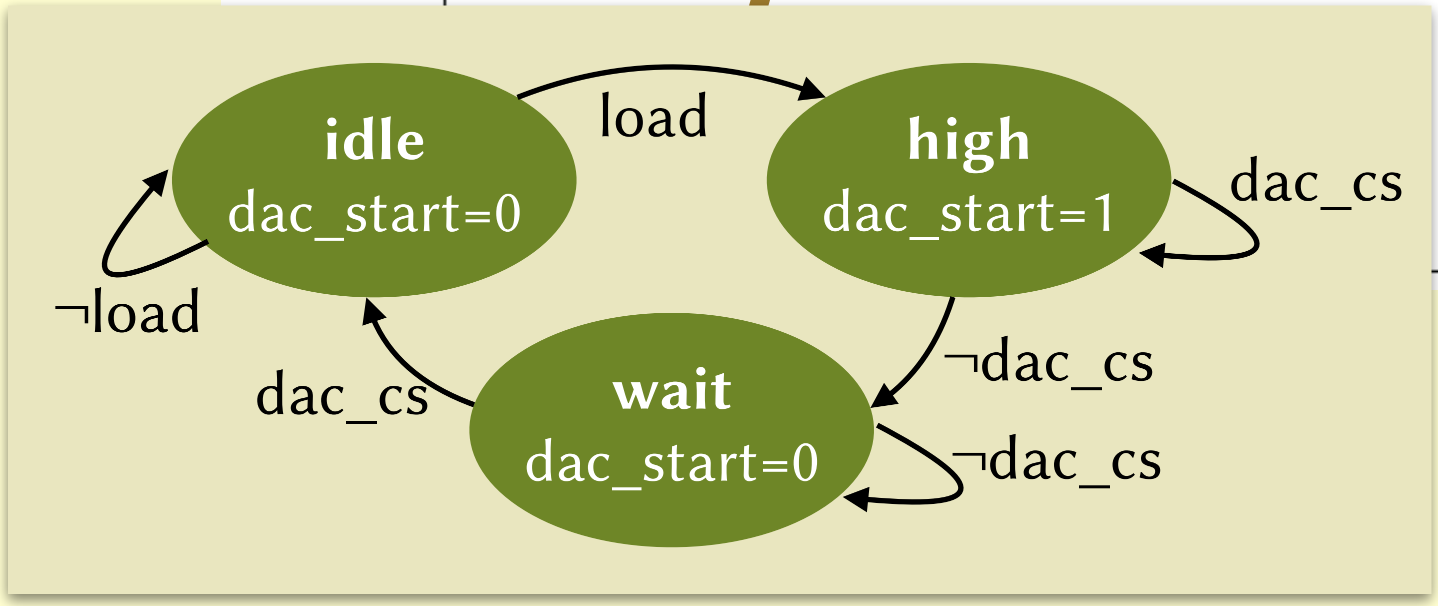
```
parameter TC = 5'd24;
reg [4:0] ctr;
always @ (posedge clk_50MHz)
  if (ctr==0) begin
    ctr <= TC;
    clk_1MHz <= ~clk_1MHz;
  end else ctr <= ctr - 1'b1;
```






shift_reg[15]
ta shift register
SDI
&
SCK
CS
spi controller FSM
LDA
1MHz

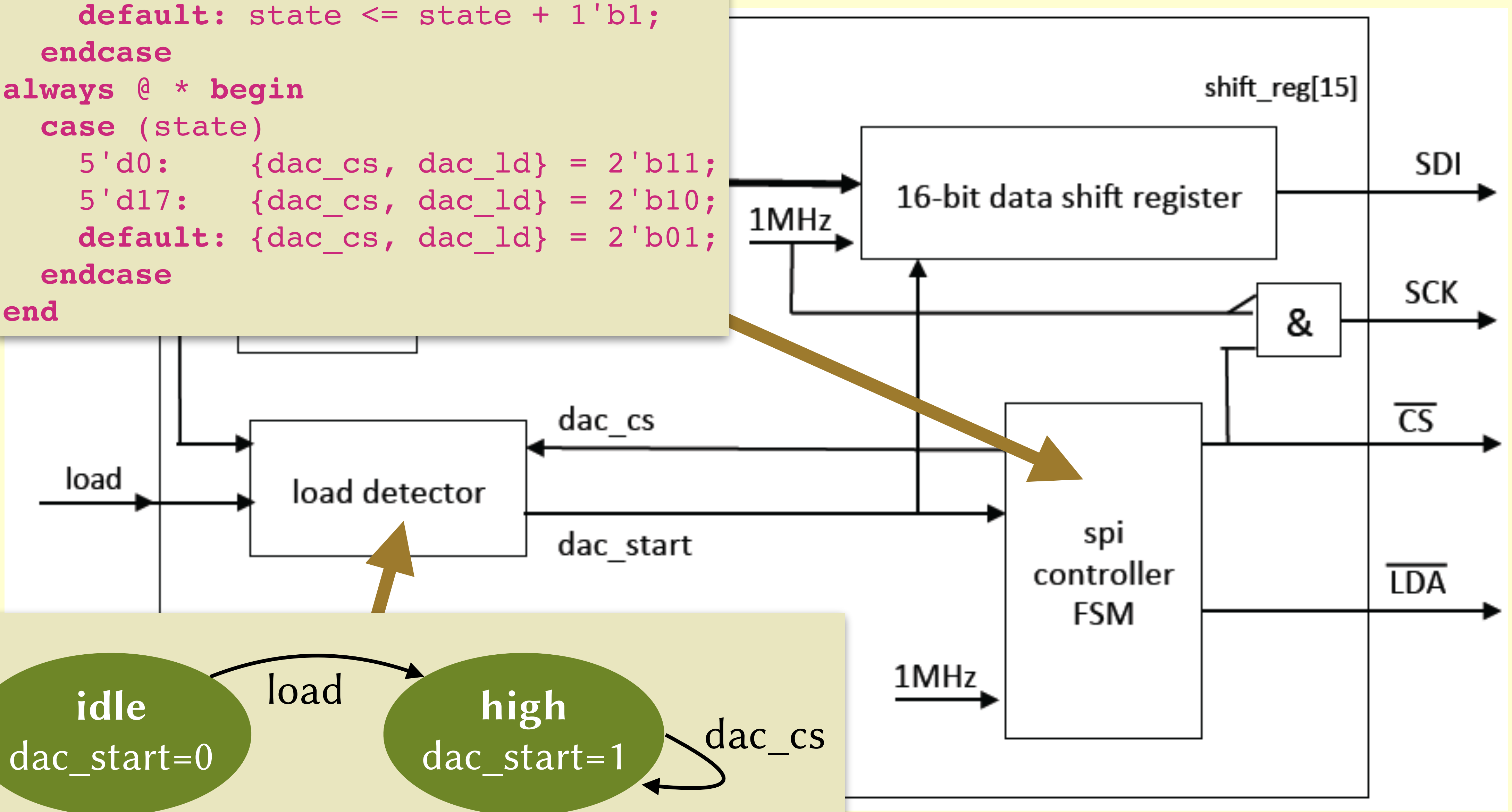

dac_start
¬dac_start
idle dac_cs=1 dac_ld=1
1 dac_cs=0 dac_ld=1
2 dac_cs=0 dac_ld=1
3 dac_cs=0 dac_ld=1
4 dac_cs=0 dac_ld=1
5 dac_cs=0 dac_ld=1
6 dac_cs=0 dac_ld=1
7 dac_cs=0 dac_ld=1
8 dac_cs=0 dac_ld=1
9 dac_cs=0 dac_ld=1
10 dac_cs=0 dac_ld=1
11 dac_cs=0 dac_ld=1
12 dac_cs=0 dac_ld=1
13 dac_cs=0 dac_ld=1
14 dac_cs=0 dac_ld=1
15 dac_cs=0 dac_ld=1
16 dac_cs=0 dac_ld=1
done dac_cs=1 dac_ld=0

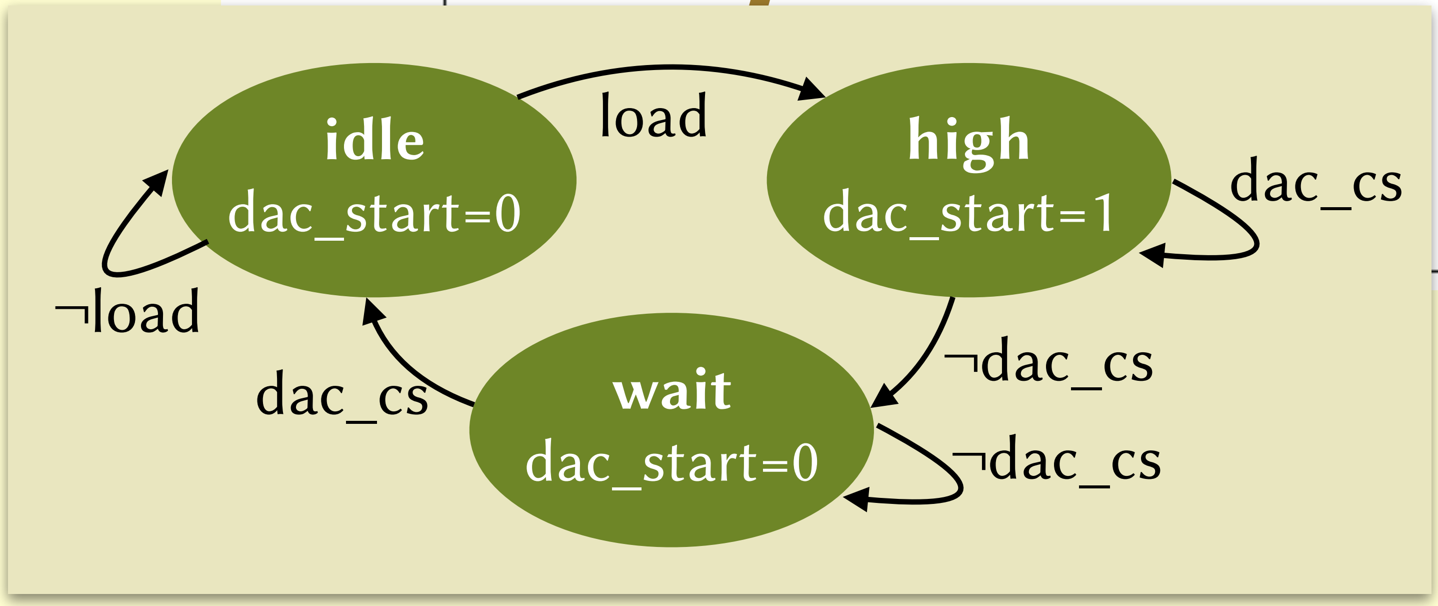

idle dac_start=0
load
high dac_start=1
dac_cs
¬load
dac_cs
¬dac_cs
wait dac_start=0
¬dac_cs

2dac

```
paramet
reg [4:
always
  if (c
    ctr
    clk
  end e
```

```
reg [4:0] state;
always @ (posedge clk_1MHz)
  case (state)
    5'd0:     if (dac_start == 1'b1)
                state <= state + 1'b1;
    5'd17:    state <= 5'd0;
    default: state <= state + 1'b1;
  endcase
always @ * begin
  case (state)
    5'd0:     {dac_cs, dac_ld} = 2'b11;
    5'd17:    {dac_cs, dac_ld} = 2'b10;
    default: {dac_cs, dac_ld} = 2'b01;
  endcase
end
```

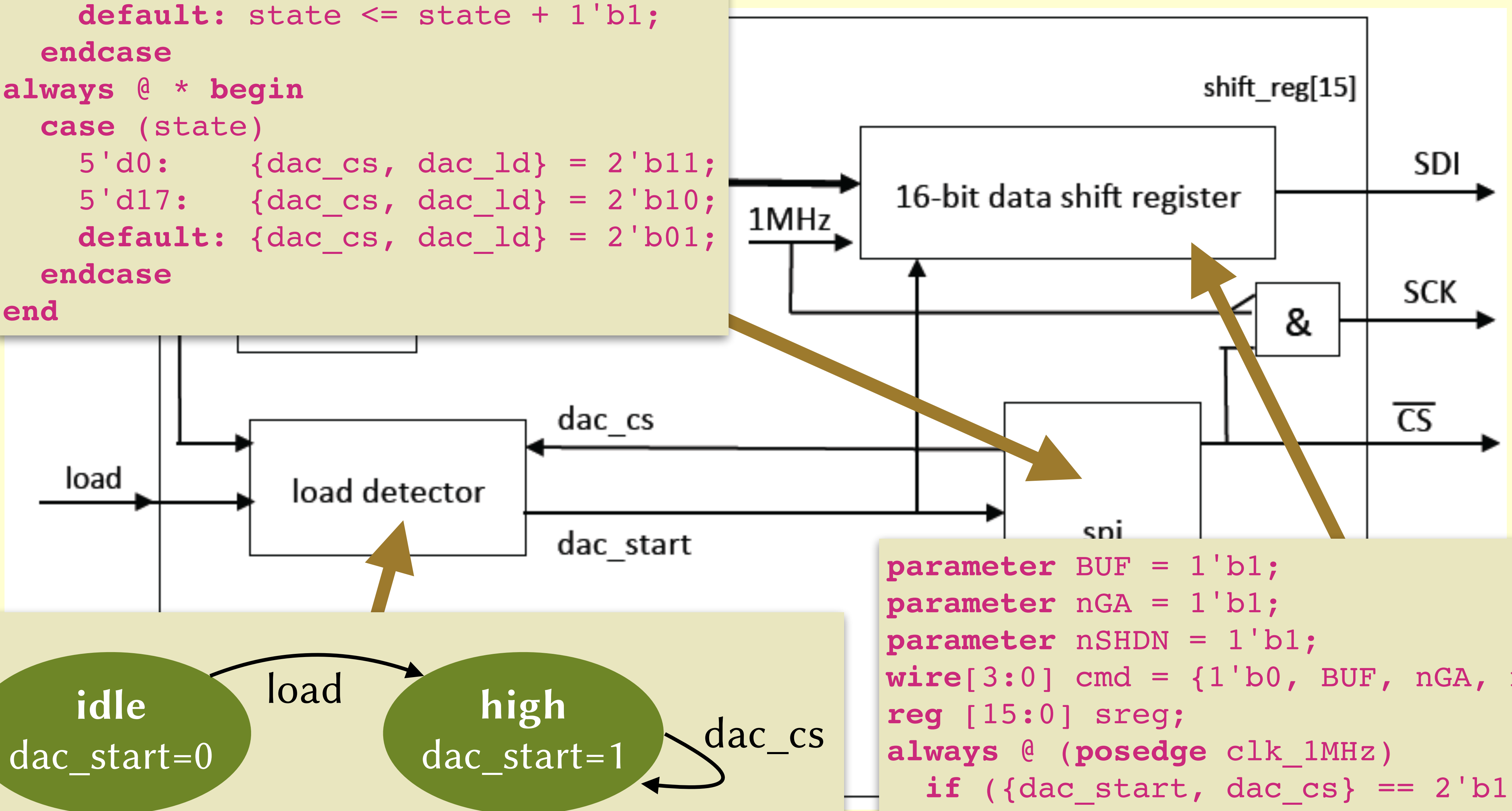


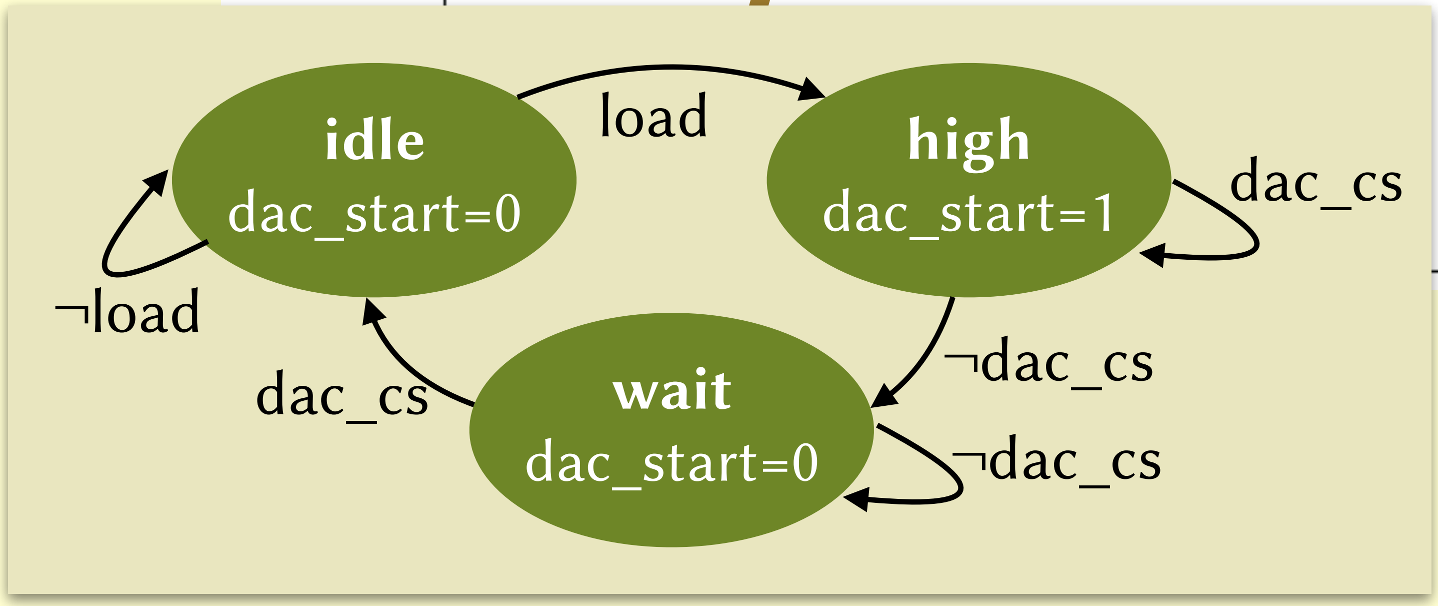


2dac

```
paramet
reg [4:(
always (
  if (c
    ctr
    clk
  end e
```

```
reg [4:0] state;
always @ (posedge clk_1MHz)
  case (state)
    5'd0:     if (dac_start == 1'b1)
                state <= state + 1'b1;
    5'd17:    state <= 5'd0;
    default: state <= state + 1'b1;
  endcase
always @ * begin
  case (state)
    5'd0:     {dac_cs, dac_ld} = 2'b11;
    5'd17:    {dac_cs, dac_ld} = 2'b10;
    default: {dac_cs, dac_ld} = 2'b01;
  endcase
end
```





```
parameter BUF = 1'b1;
parameter nGA = 1'b1;
parameter nSHDN = 1'b1;
wire[3:0] cmd = {1'b0, BUF, nGA, nSHDN};
reg [15:0] sreg;
always @ (posedge clk_1MHz)
  if ({dac_start, dac_cs} == 2'b11)
    sreg <= {cmd, data_in, 2'b00};
  else
    sreg <= {sreg[14:0], 1'b0};
assign dac_sck = !clk_1MHz & !dac_cs;
assign dac_sdi = sreg[15];
```

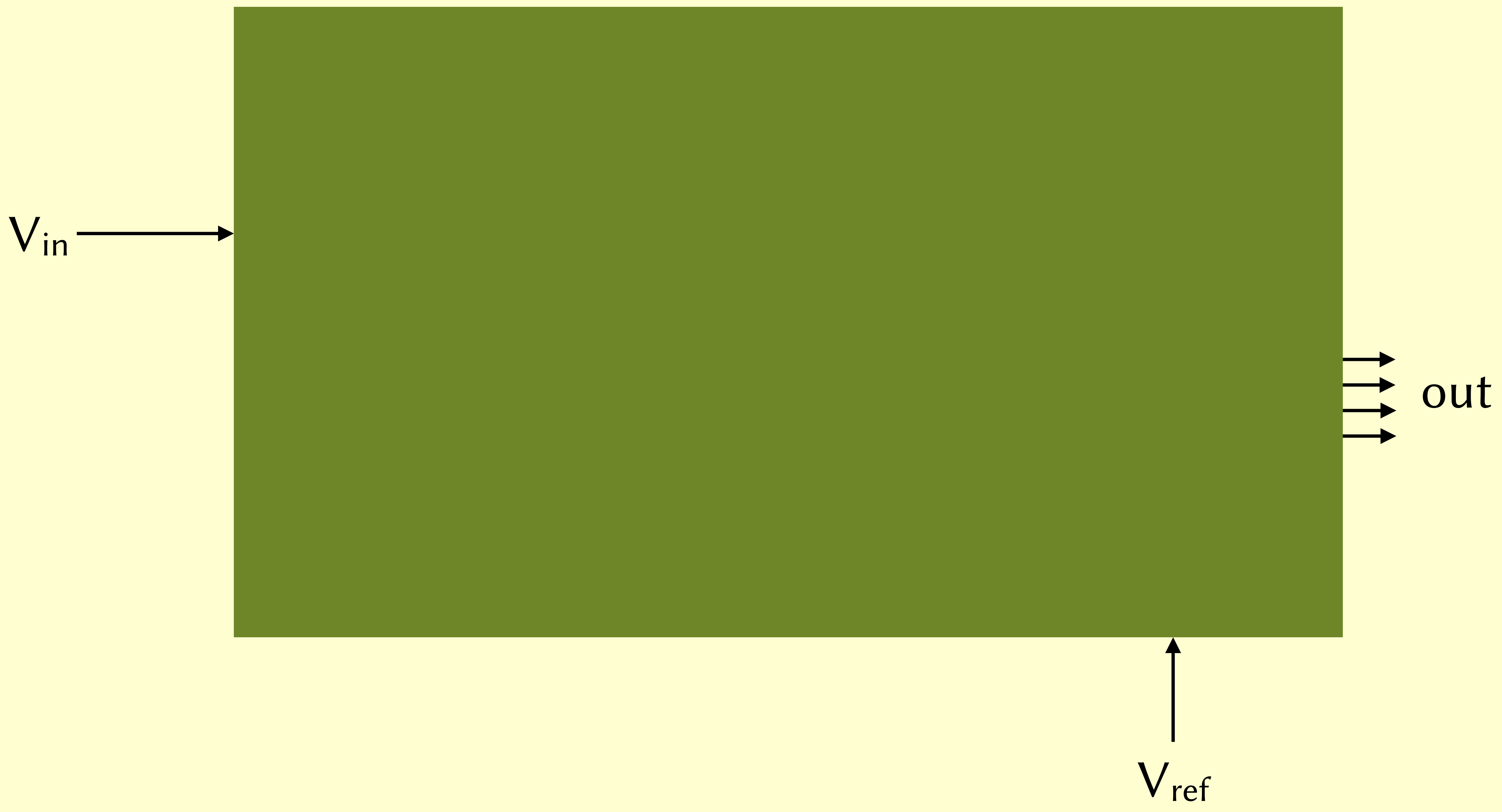
This lecture

- The design of the spi2dac module
- How the ADC works
- Wrapping up

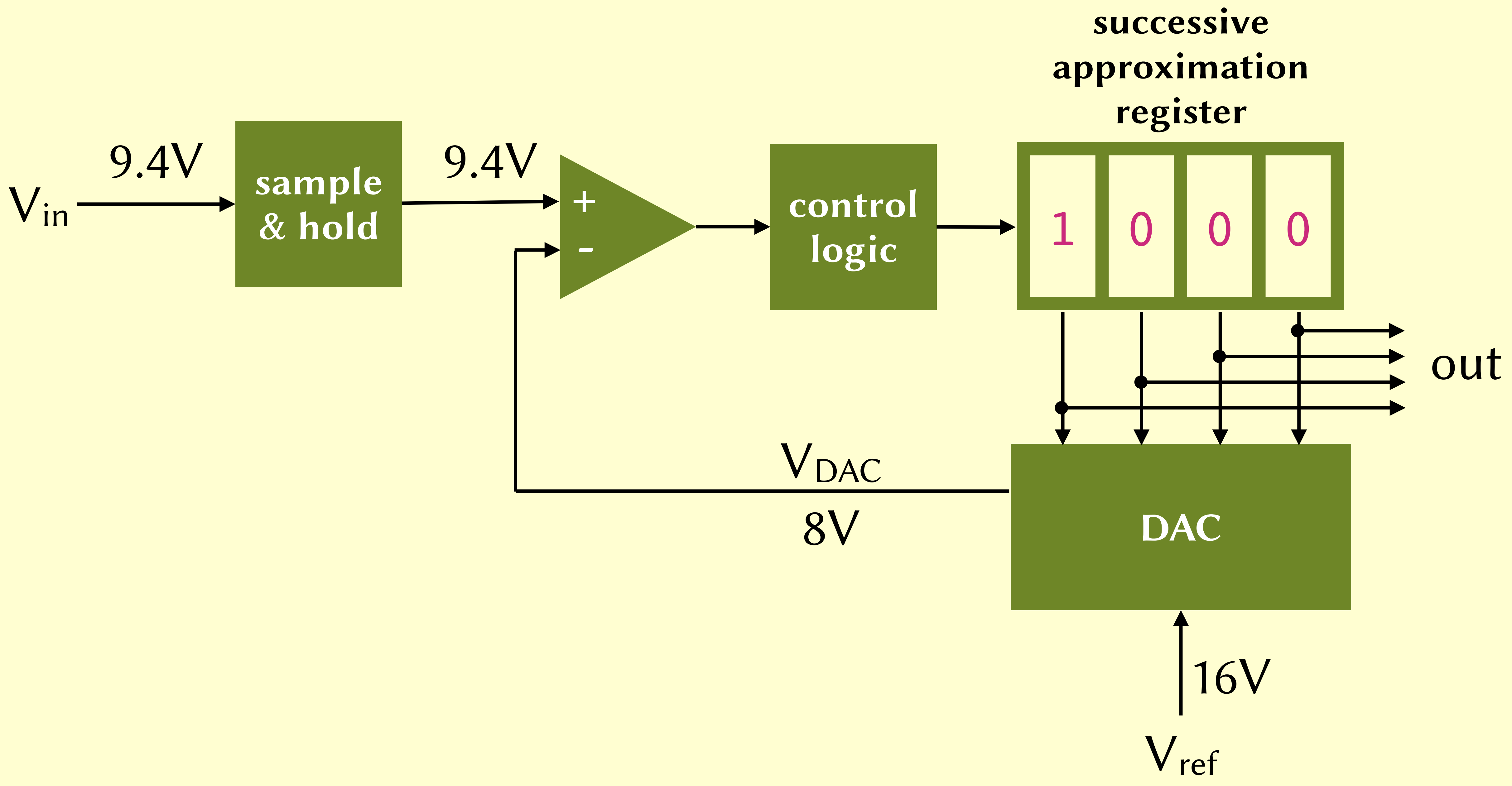
This lecture

- The design of the spi2dac module
- **How the ADC works**
- Wrapping up

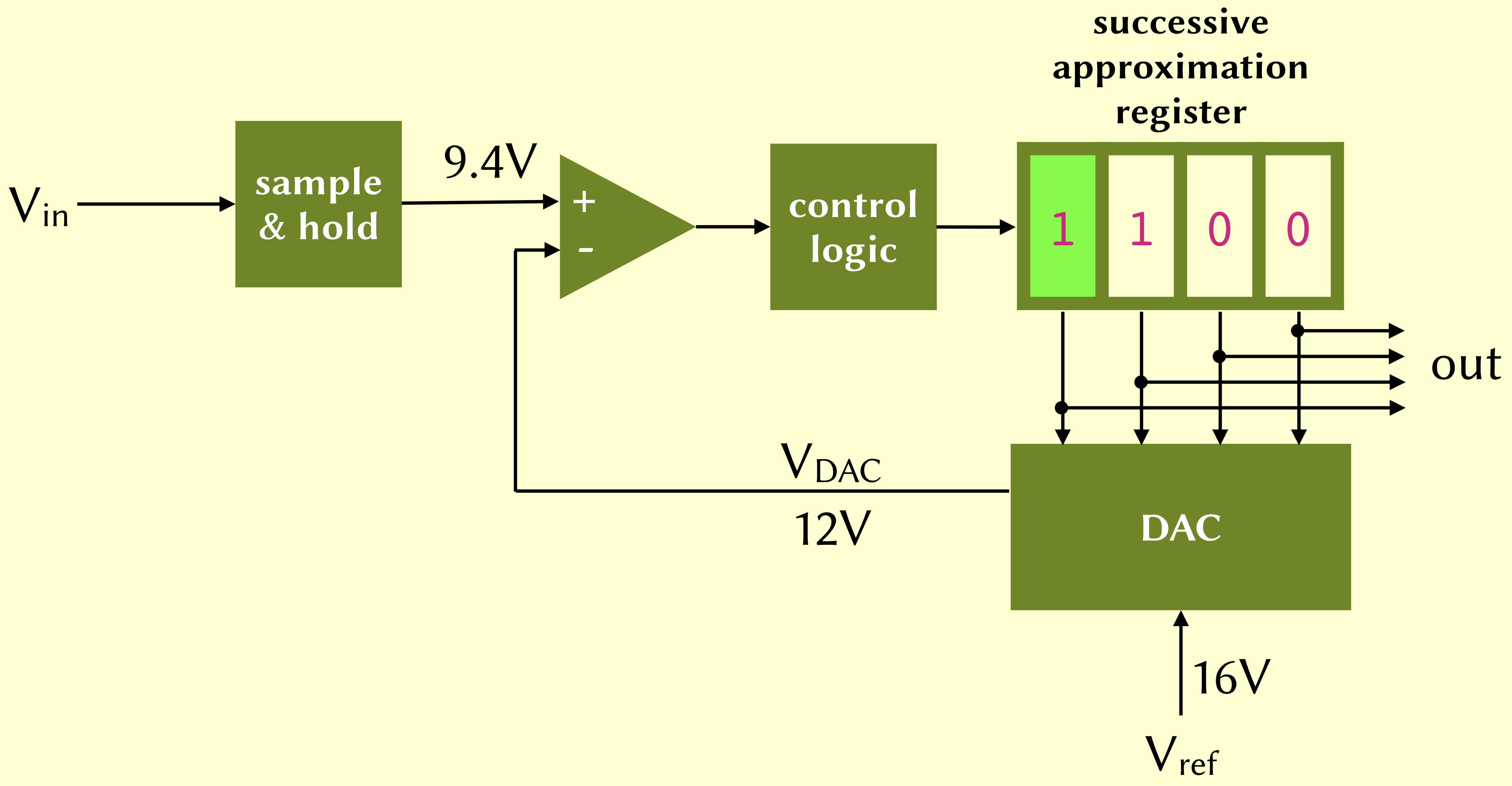
ADC using SAR



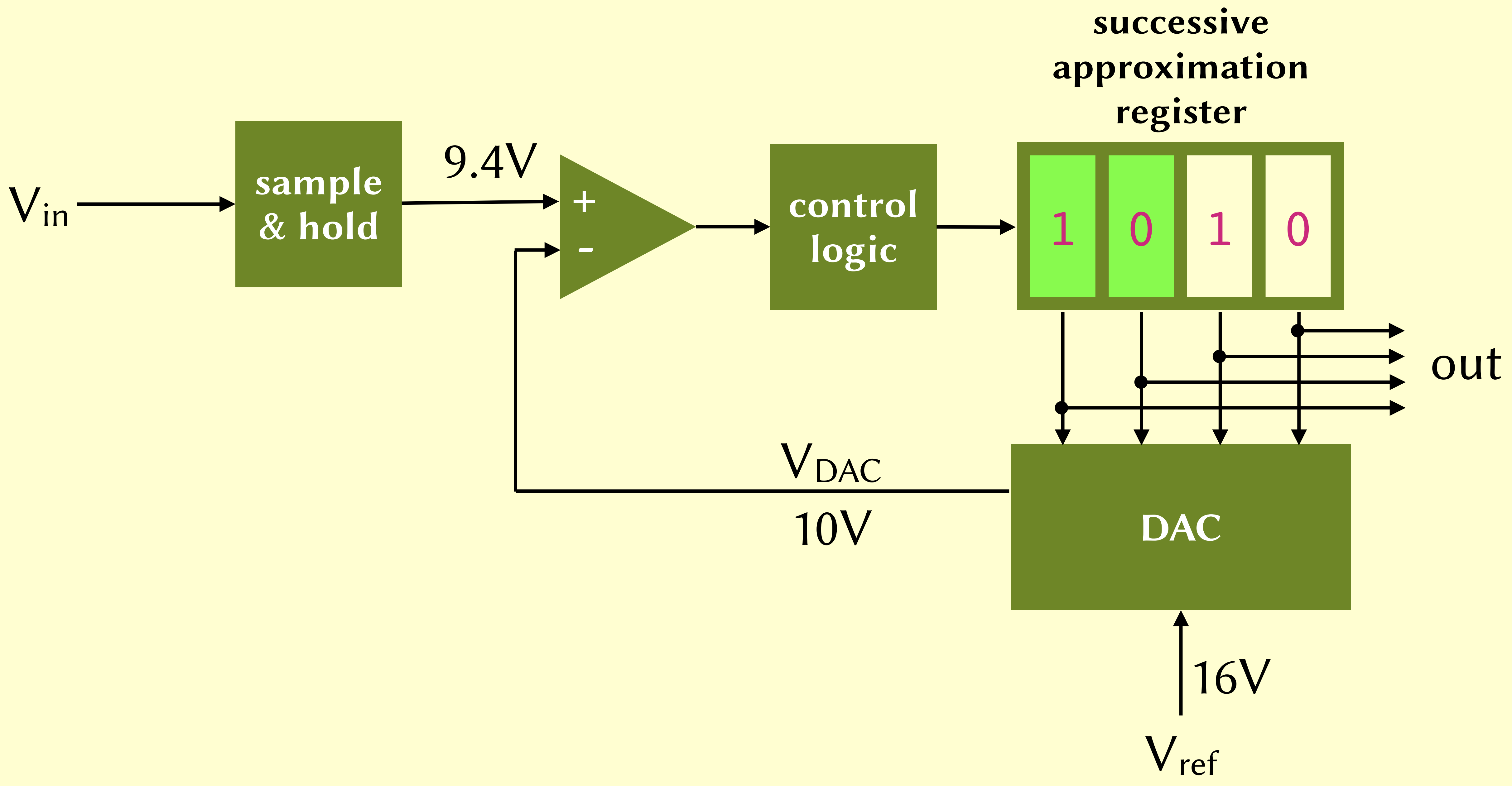
ADC using SAR



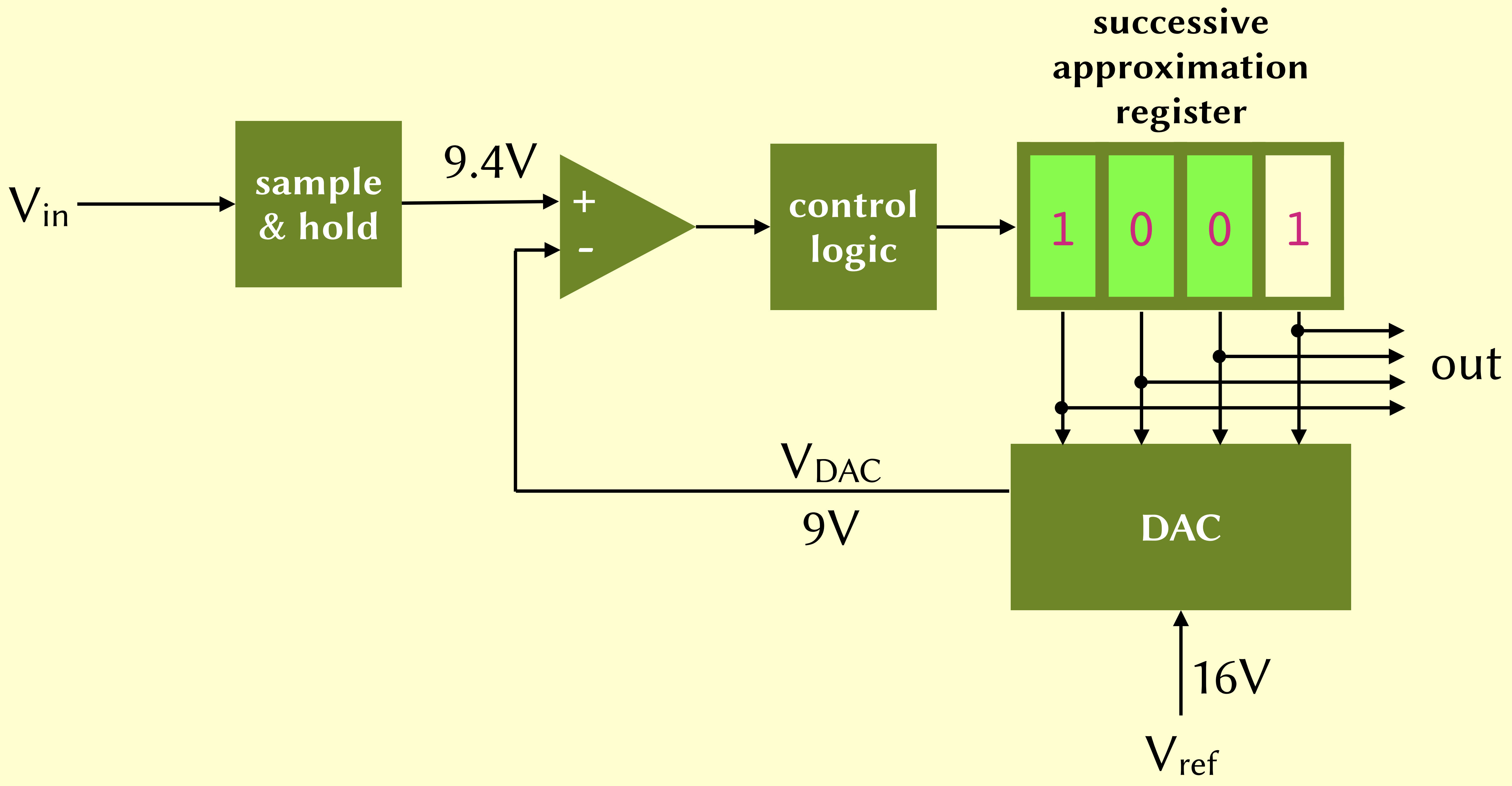
ADC using SAR



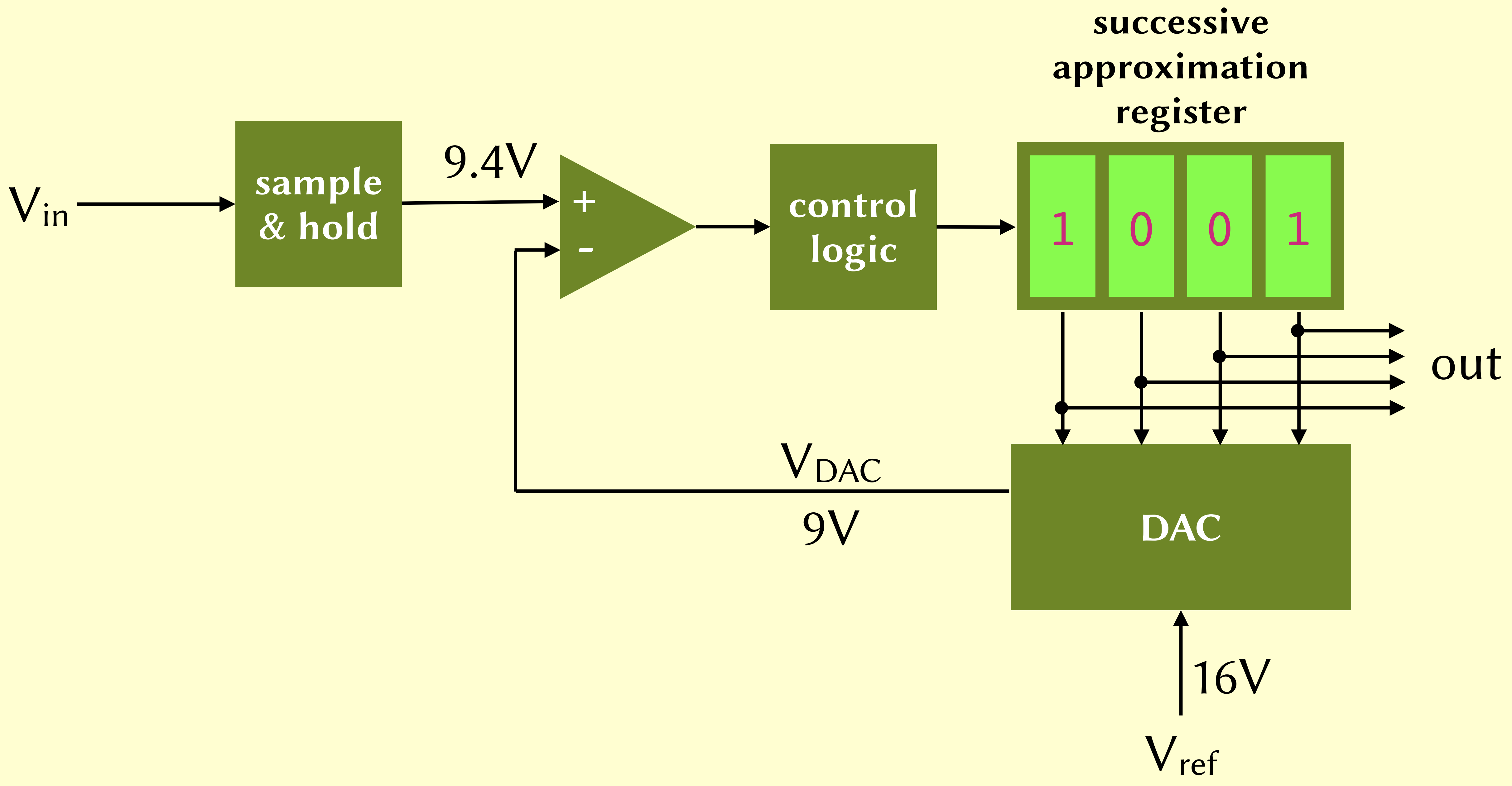
ADC using SAR



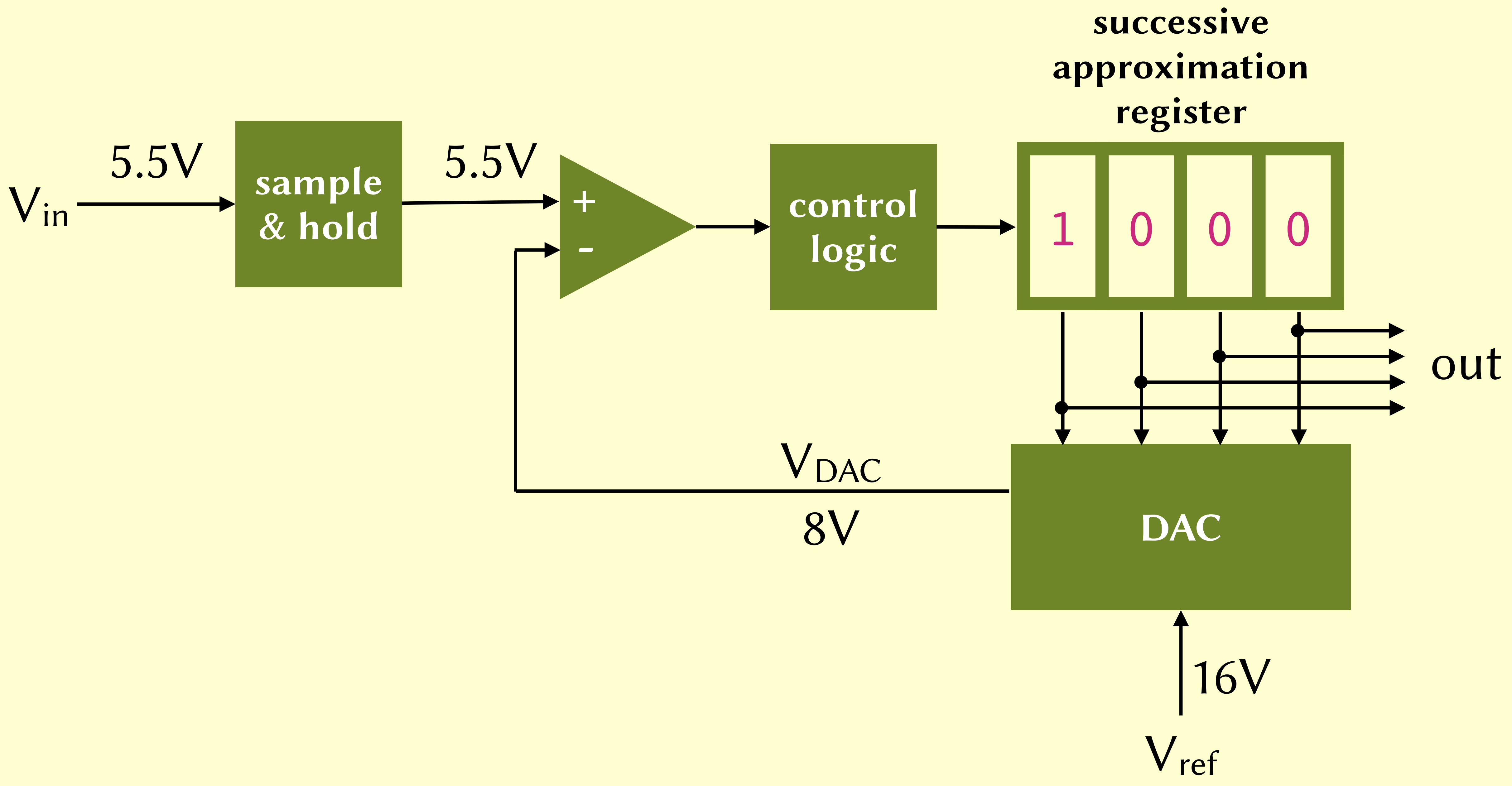
ADC using SAR



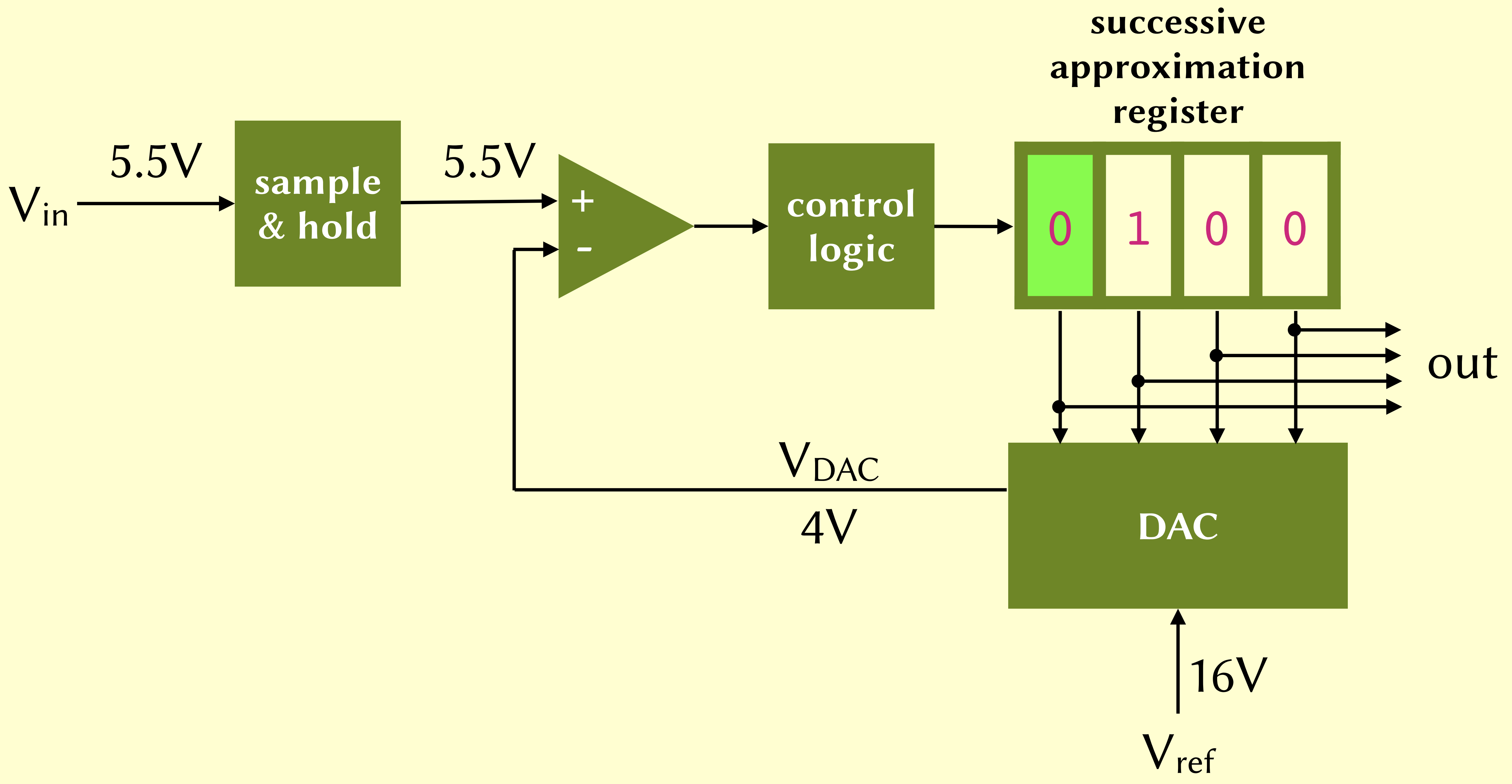
ADC using SAR



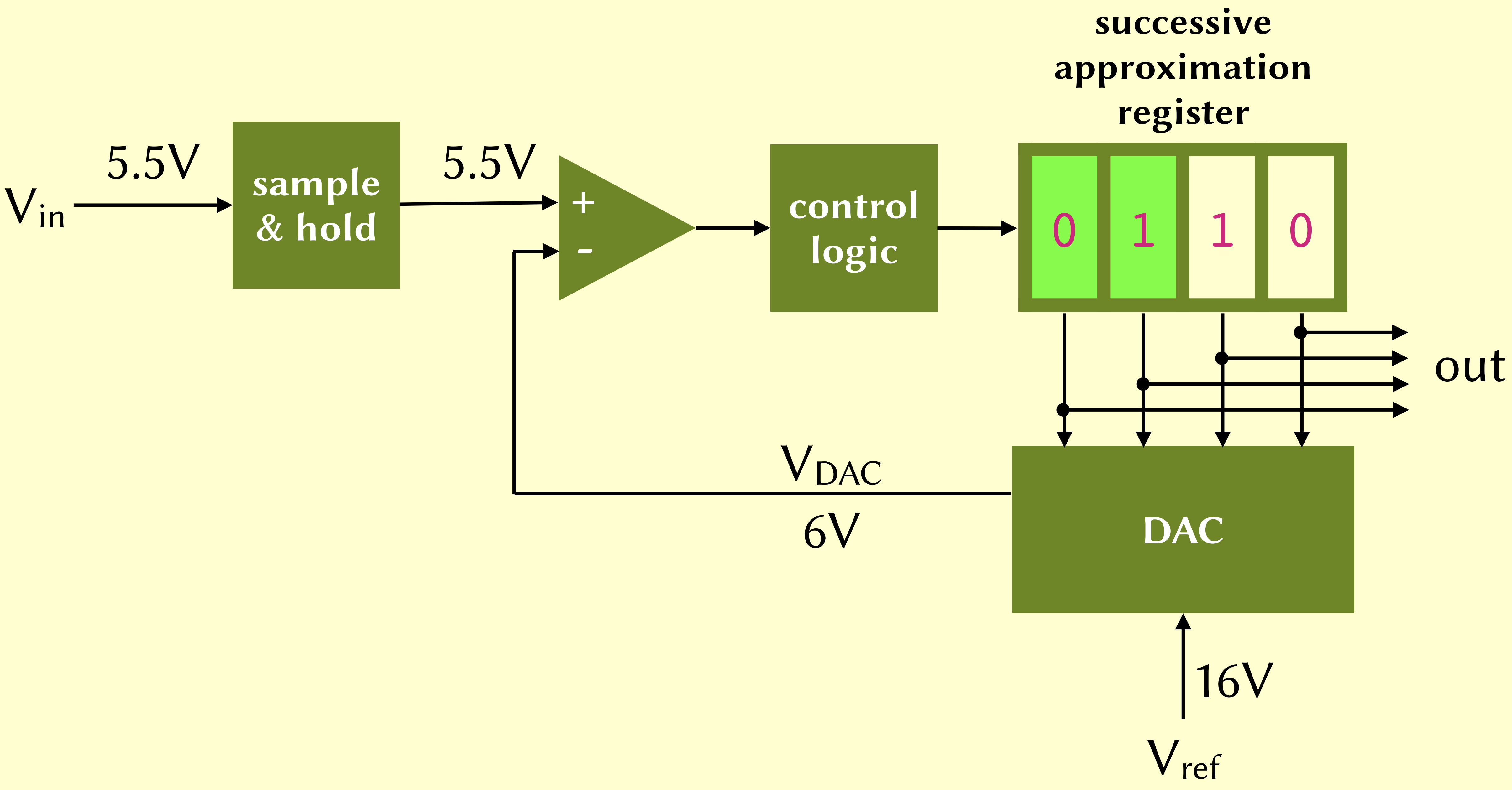
ADC using SAR



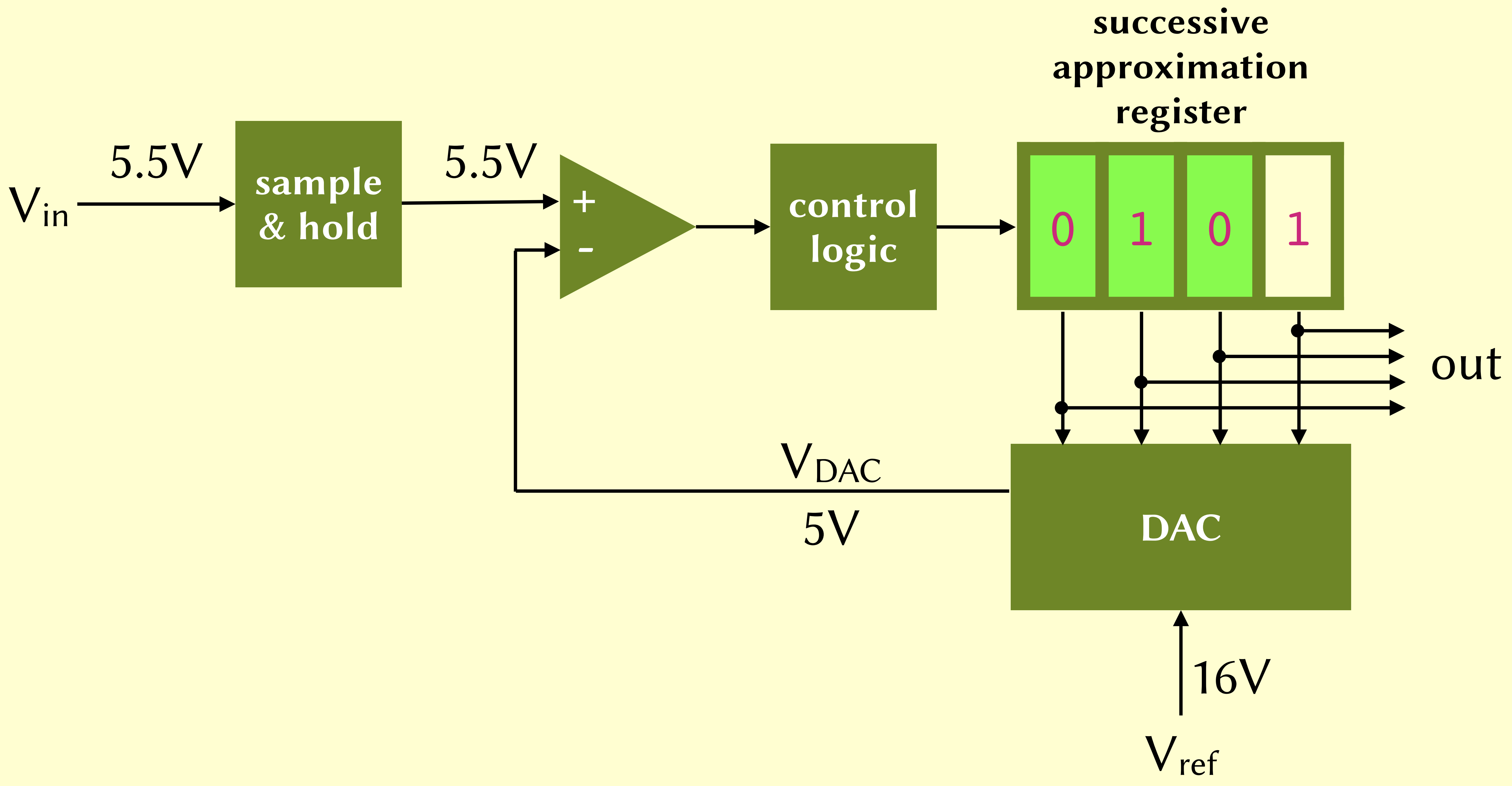
ADC using SAR



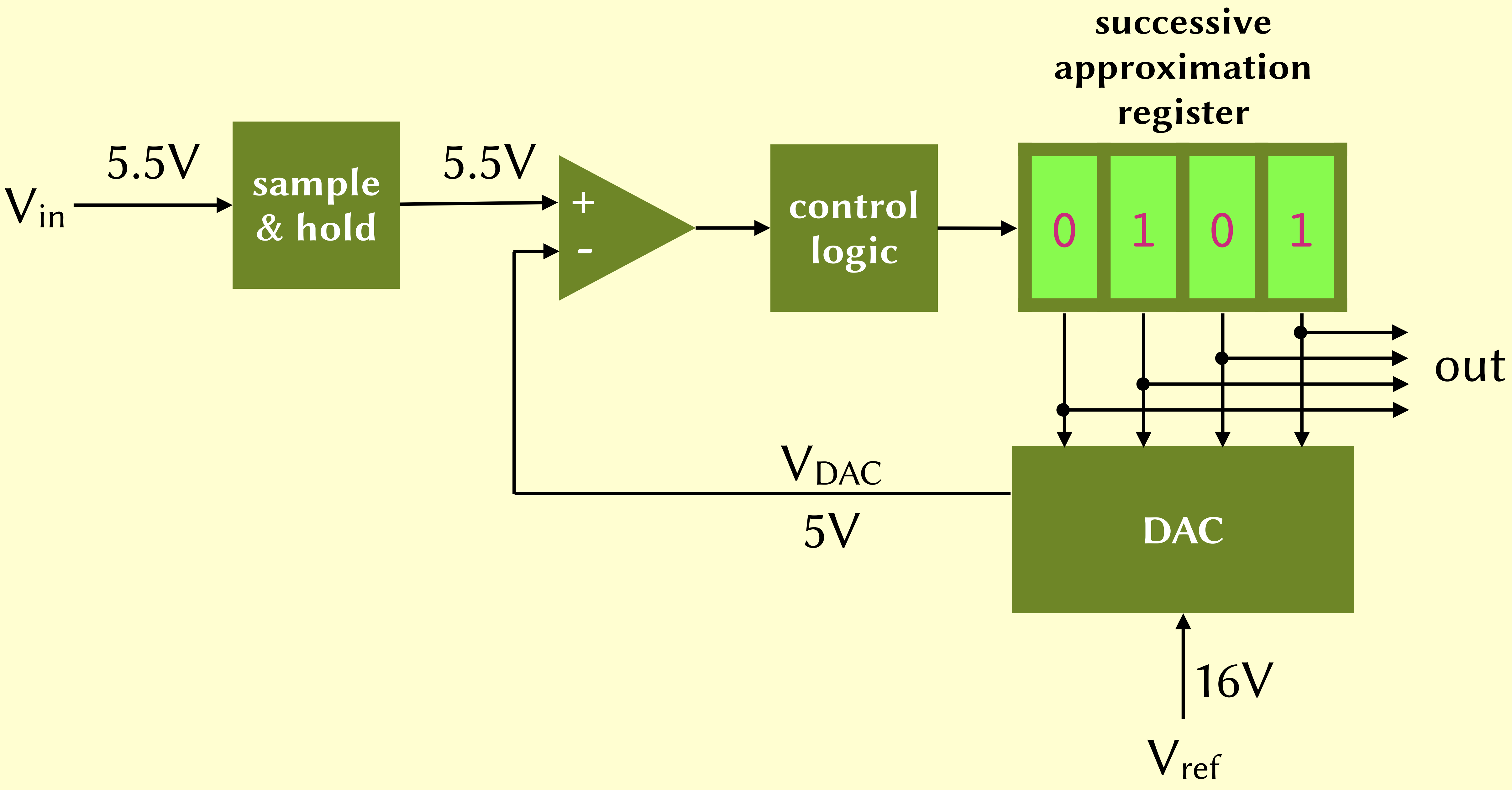
ADC using SAR



ADC using SAR



ADC using SAR



ADC using SAR

- An N-bit ADC takes N cycles to convert one sample
- Low power consumption
- Simple design with small circuit area
- Low latency
- More advanced designs are more efficient when the sampling rate is above 10MHz or $N > 16$.

This lecture

- The design of the spi2dac module
- How the ADC works
- **Wrapping up**

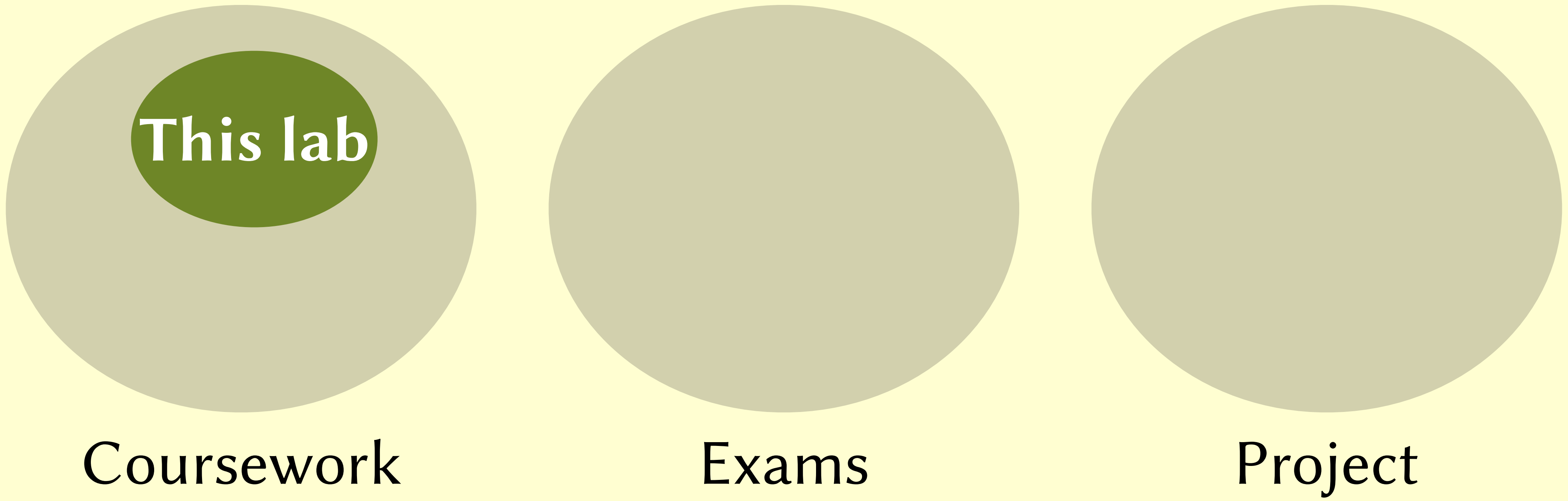
Module aims

- To ensure that all MSc students reach a **common competence level** in RTL design using FPGAs in a hardware description language.

- To act as a **revision exercise** for those already competent in Verilog and FPGAs.

Grading

- This lab is part of the Coursework component of the MSc course.
- You must pass this component, but it does not count towards your final MSc grade.



Assessment

- Assessment will be via a 15-minute **oral interview**. This will assess:
    - how many parts of the experiment you have completed,
    - the extent to which you have understood the underlying principles of digital design, and
    - whether you have used a logbook (electronic or paper) to help you learn through planning and reflection.
- Please have your equipment set up and ready to demo.