EEL 4742 – Embedded Systems
Module 8 – ADC

Hadi M Kamali

Department of Electrical and Computer Engineering (**ECE**)
University of Central Florida

Office Location/phone: HEC435 – (407) 823-0764
webpage: https://www.ece.ucf.edu/~kamali/
e-mail: kamali@ucf.edu

HAVEN Research Group
https://haven.ece.ucf.edu/

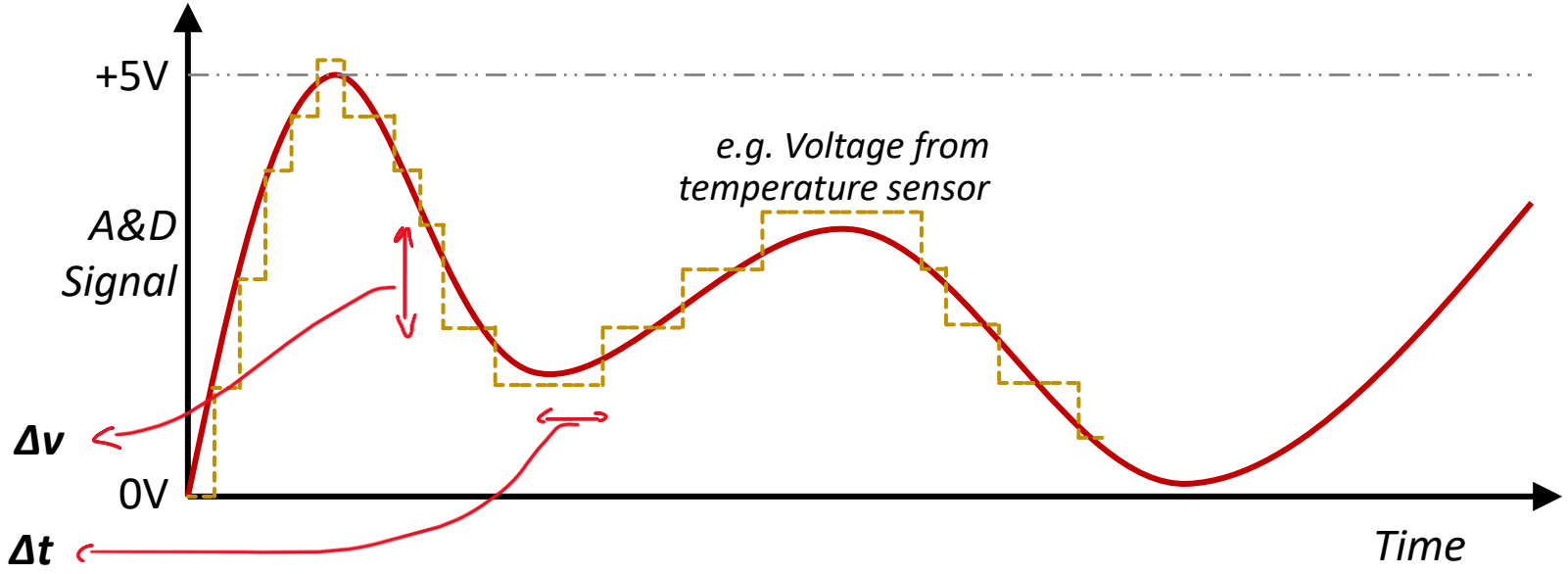
UNIVERSITY OF CENTRAL FLORIDA

Why ADC

- Analog-to-Digital-Converter (ADC)
  - For encoding analog signals into digital signals

analog: *continuously valued signal, such as temperature, speed, or voltage with infinite possible values in between*

digital: *discretely valued signal, such as integers encoded in binary*

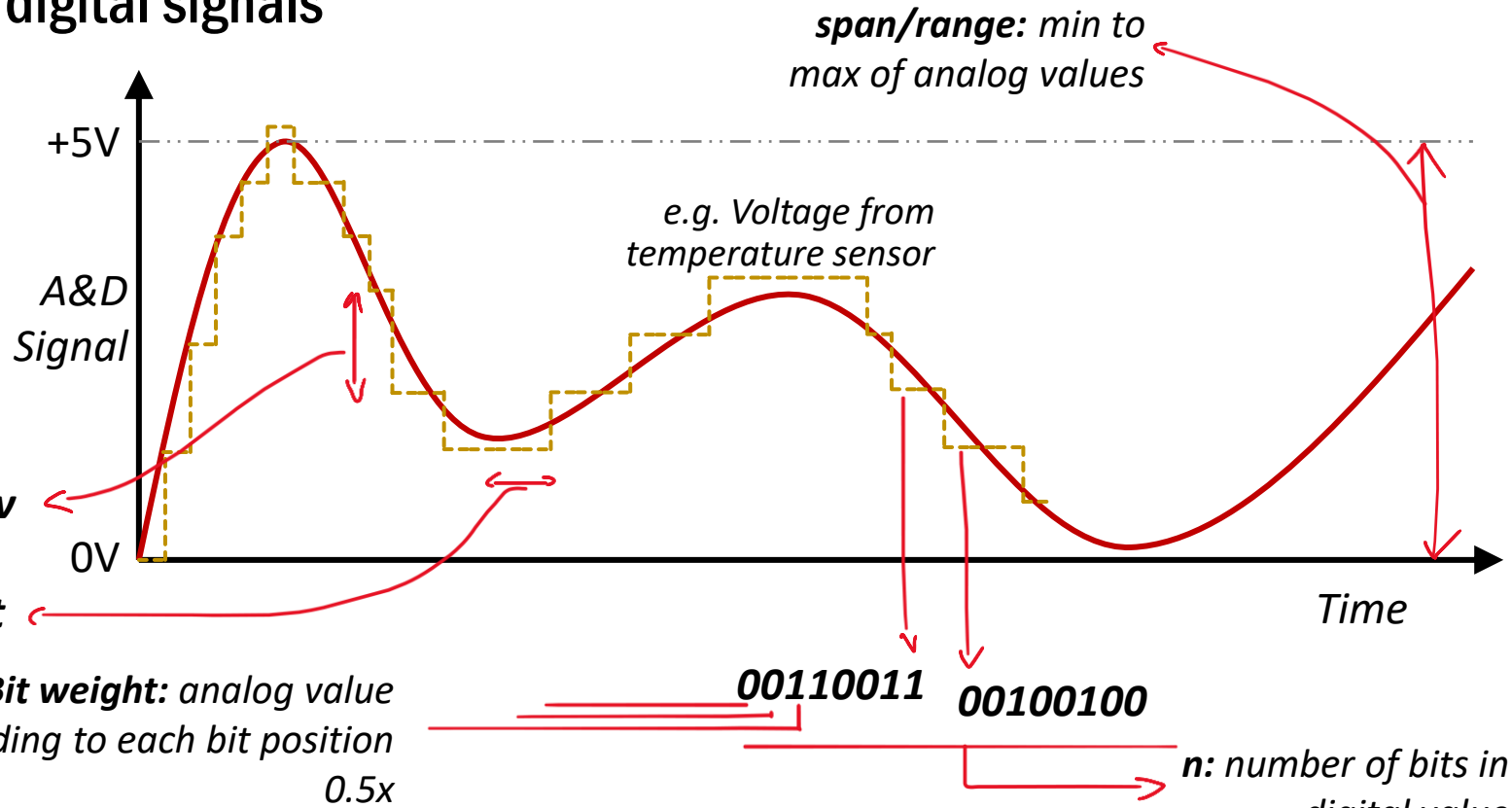


Terminology of ADC

- Analog-to-Digital-Converter (ADC)
  - For encoding analog signals into digital signals

analog: *continuously valued signal, such as temperature, speed, or voltage with infinite possible values in between*

digital: *discretely valued signal, such as integers encoded in binary*

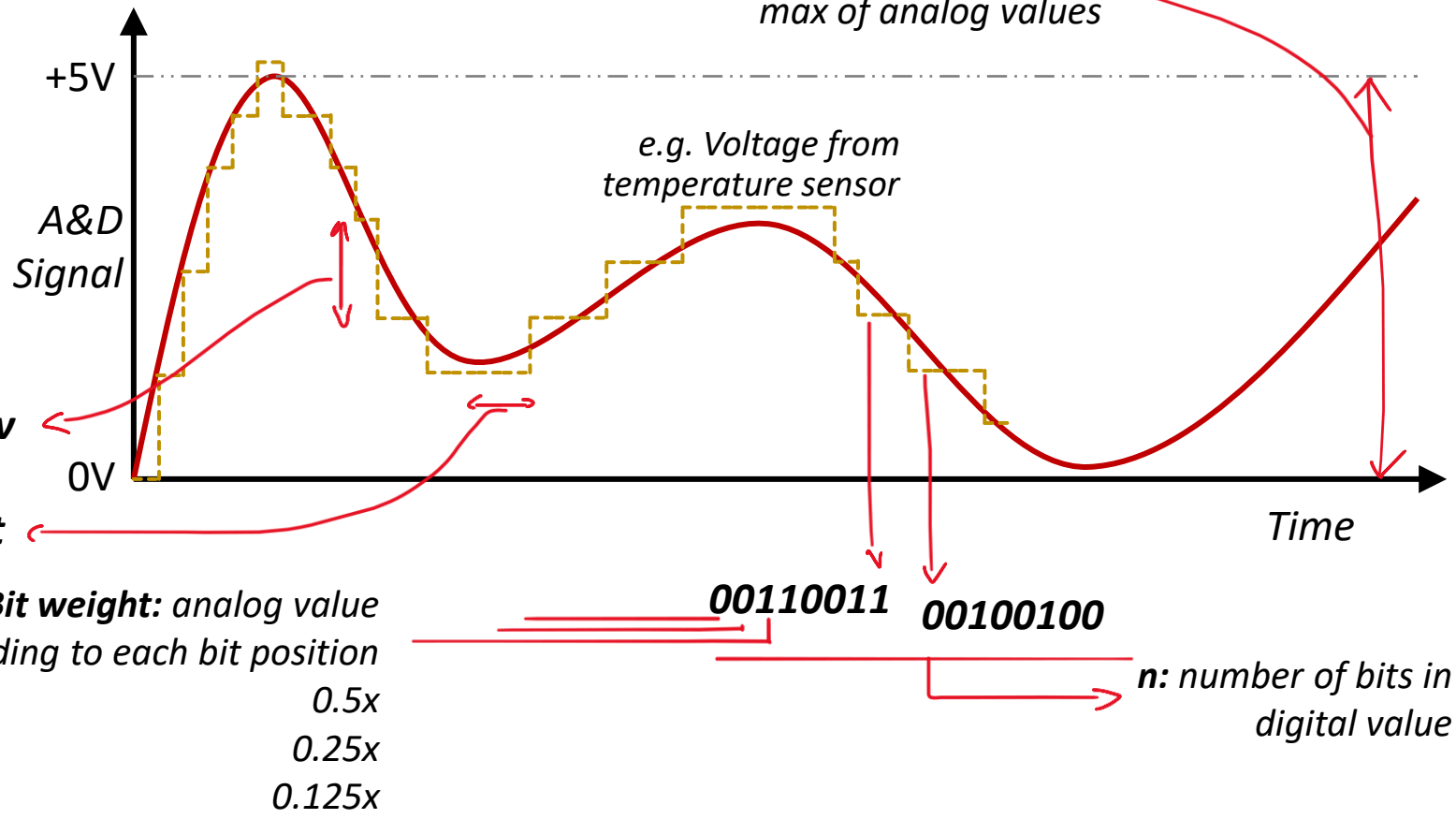


Step Size (Resolution)

- Analog-to-Digital-Converter (ADC)
  - For encoding analog signals into digital signals

Number of steps = 2^n

Resolution = Range / (Number of steps)



Step size or resolution: *smallest analog change resulting in one increment/decrement*

Sampling time: *minimum time required per each digital value representation*

Bit weight: *analog value corresponding to each bit position*
0.5x
0.25x
0.125x

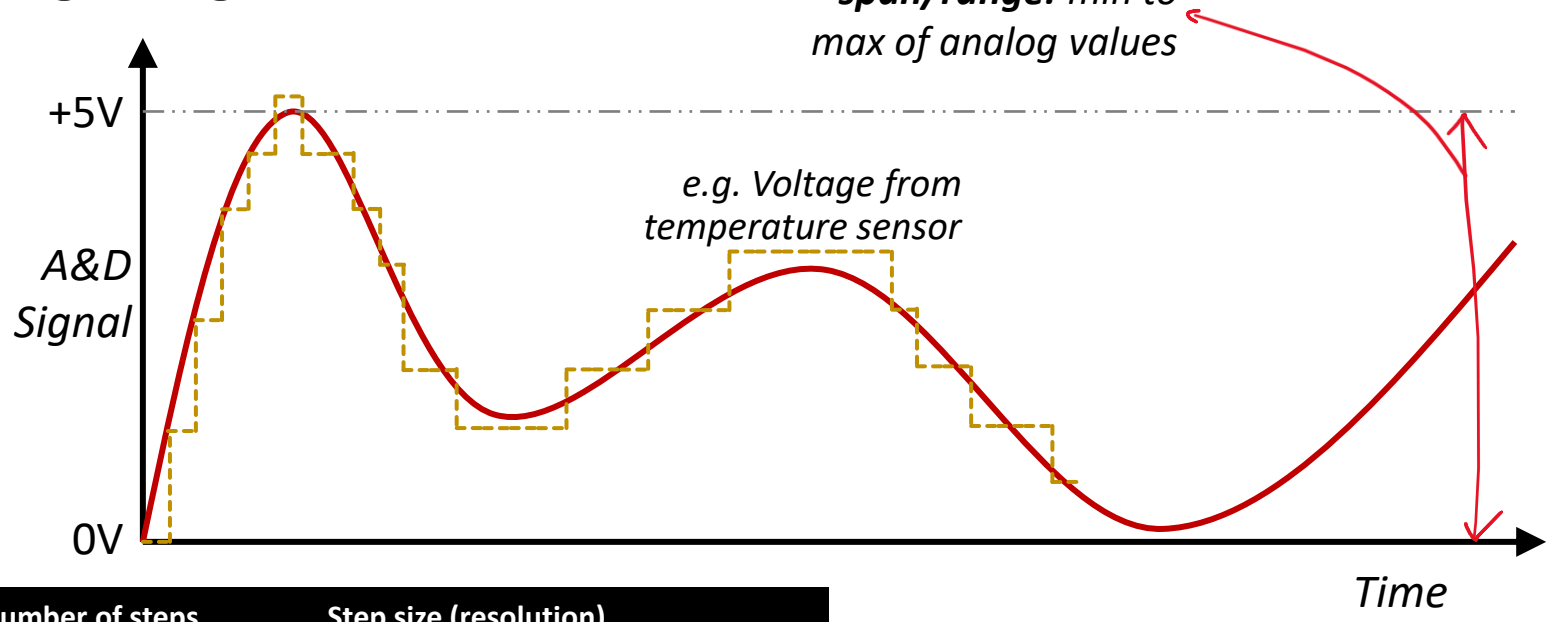
n: *number of bits in digital value*

Step Size (Resolution)

- Analog-to-Digital-Converter (ADC)
  - For encoding analog signals into digital signals

Number of steps = 2^n

Resolution = Range / (Number of steps)

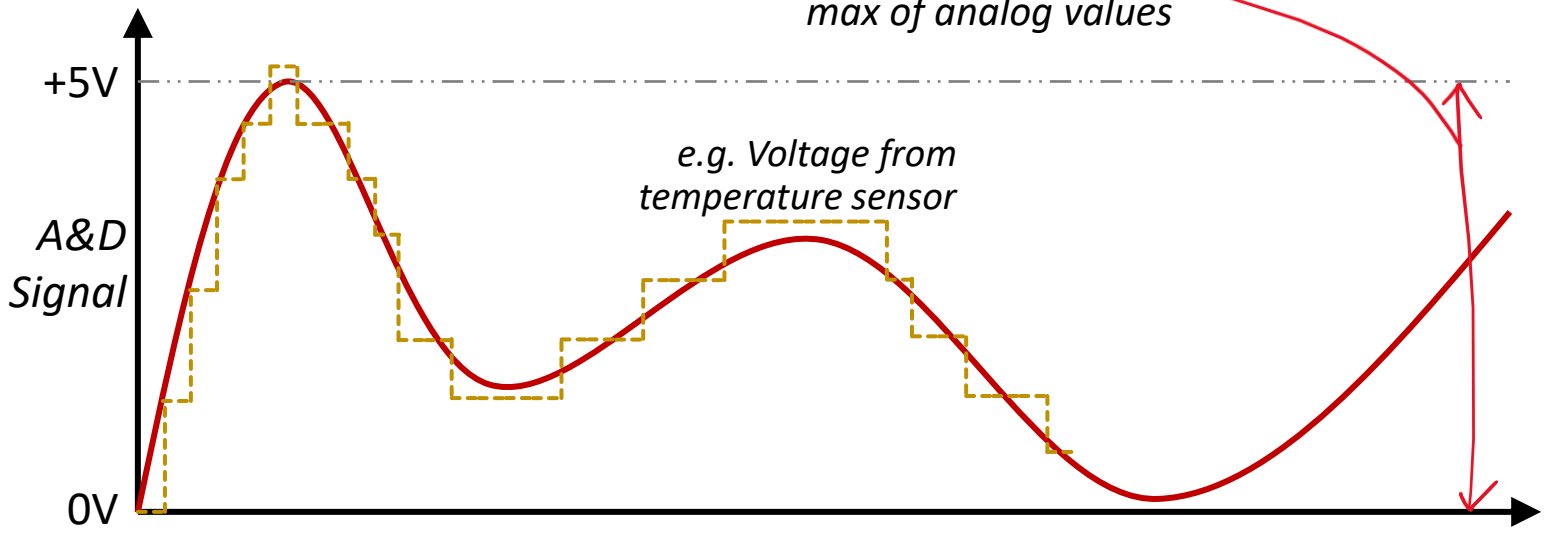


Number of bits (n)	Number of steps	Step size (resolution)
4	16	(5-0)/16 = 312.5 mV
8	256	(5-0)/256 = 19.53 mV
10	1,024	(5-0)/1,024 = 4.88 mV
12	4,096	(5-0)/4,096 = 1.2 mV
16	65,536	(5-0)/65536 = 0.076 mV

Step Size (Resolution)

- Analog-to-Digital-Converter (ADC)
  - For encoding analog signals into digital signals

span/range: *min to max of analog values*



Number of steps = 2^n

Resolution = Range / (Number of steps)

(5-0)/16 = 312.5 mV

0 V	0.3125 V	0.625 V	0.9375 V	1.25 V	1.5625 V	1.875 V	2.1875 V	2.5 V	2.8125 V	3.125 V	3.4375 V	3.75 V	4.0625 V	4.375 V	4.6875 V
0h0	0h1	0h2	0h3	0h4	0h5	0h6	0h7	0h8	0h9	0hA	0hB	0hC	0hD	0hE	0hF

(5-0)/32 = 156.25 mV

0 V	0.15625 V	0.3125 V	0.46875 V	0.625 V	0.78125 V	0.9375 V	1.09375 V	…	3.75 V	3.90625 V	4.0625 V	4.21875 V	4.375 V	4.53125 V	4.6875 V	4.84375 V
0h0	0h1	0h2	0h3	0h4	0h5	0h6	0h7	…	0h18	0h19	0h1A	0h1B	0h1C	0h1D	0h1E	0h1F

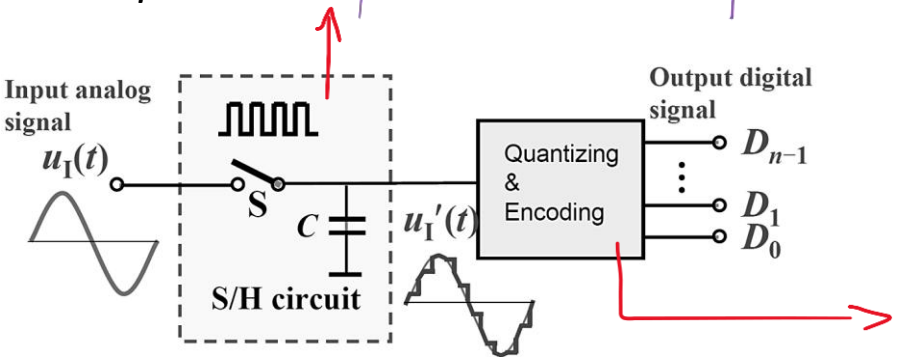
Step-by-step ADC Conceptual Flow

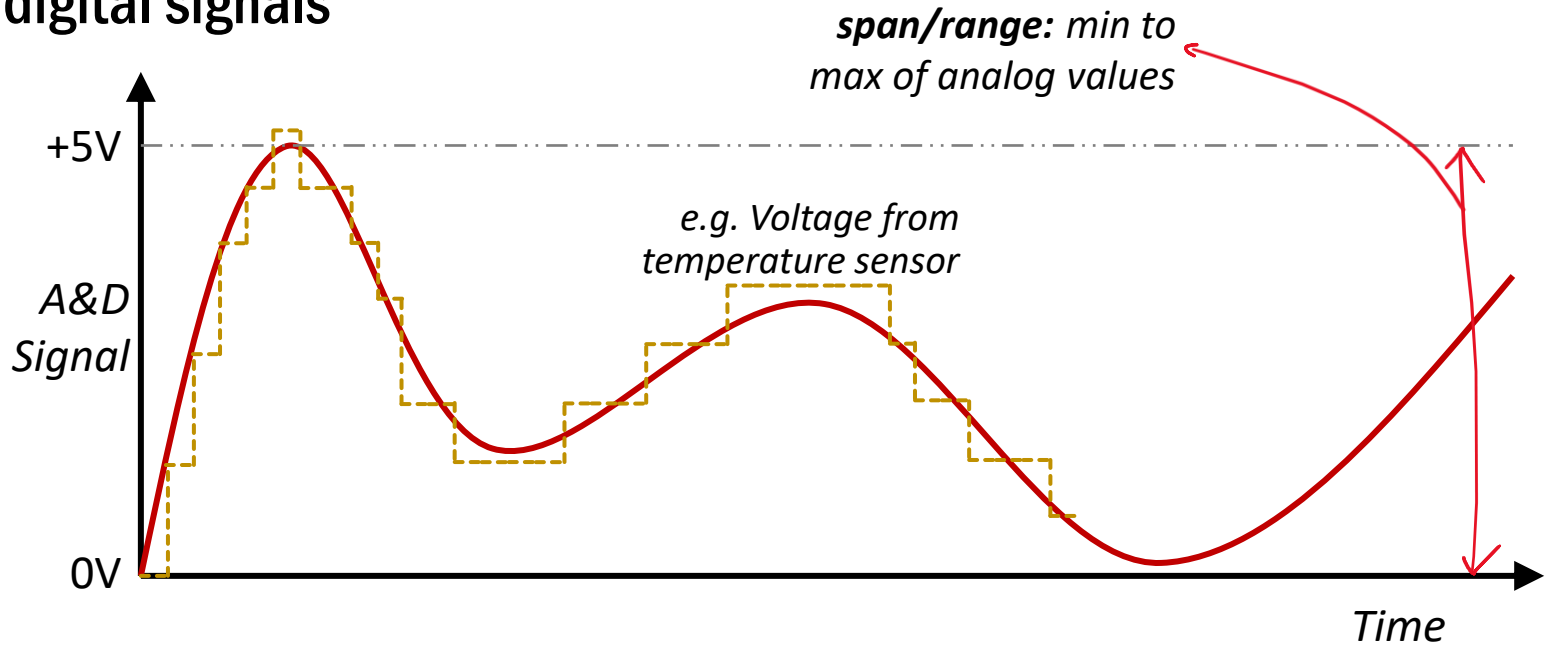
- Analog-to-Digital-Converter (ADC)
  - For encoding analog signals into digital signals

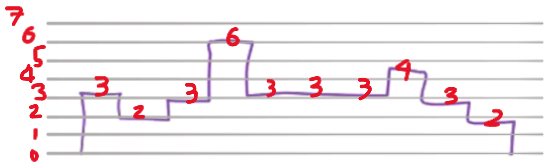
sampling

holding

digital capture







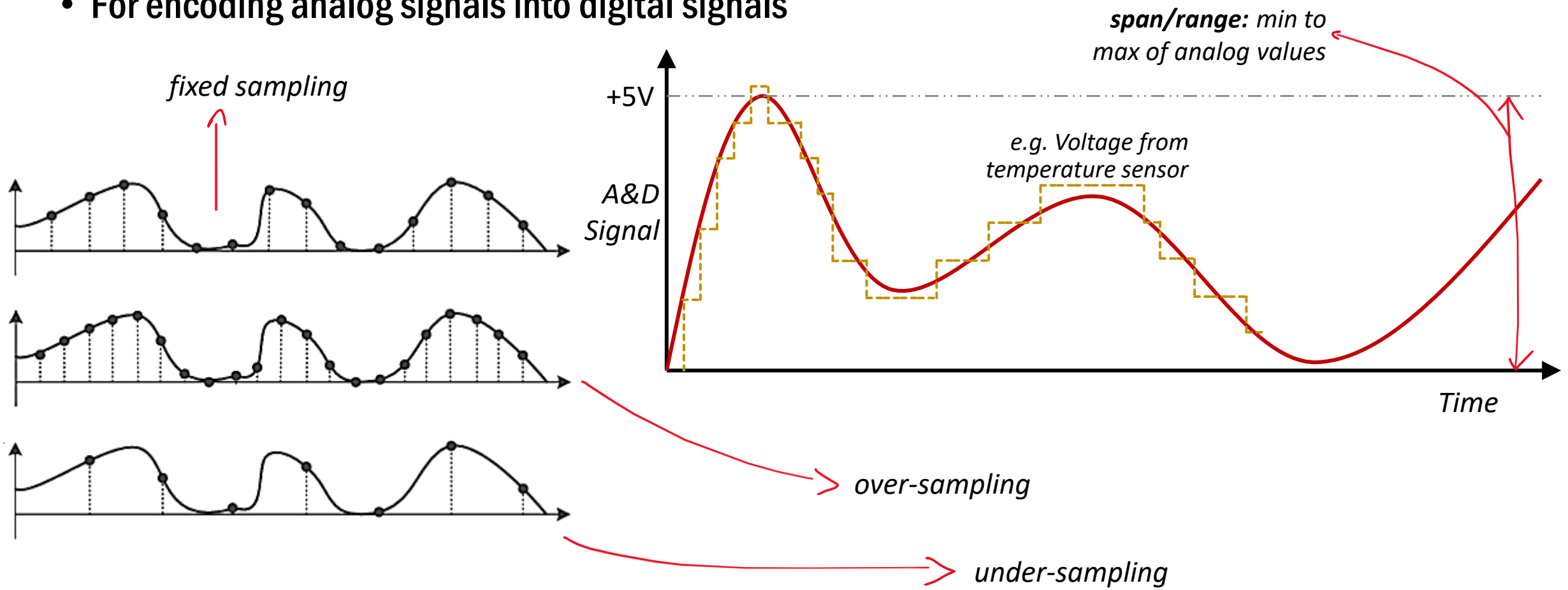
quantization

0h3 0h2 0h3 0h6 0h3 0h3 0h3

encoding

Sampling Ratio

- Analog-to-Digital-Converter (ADC)
  - For encoding analog signals into digital signals

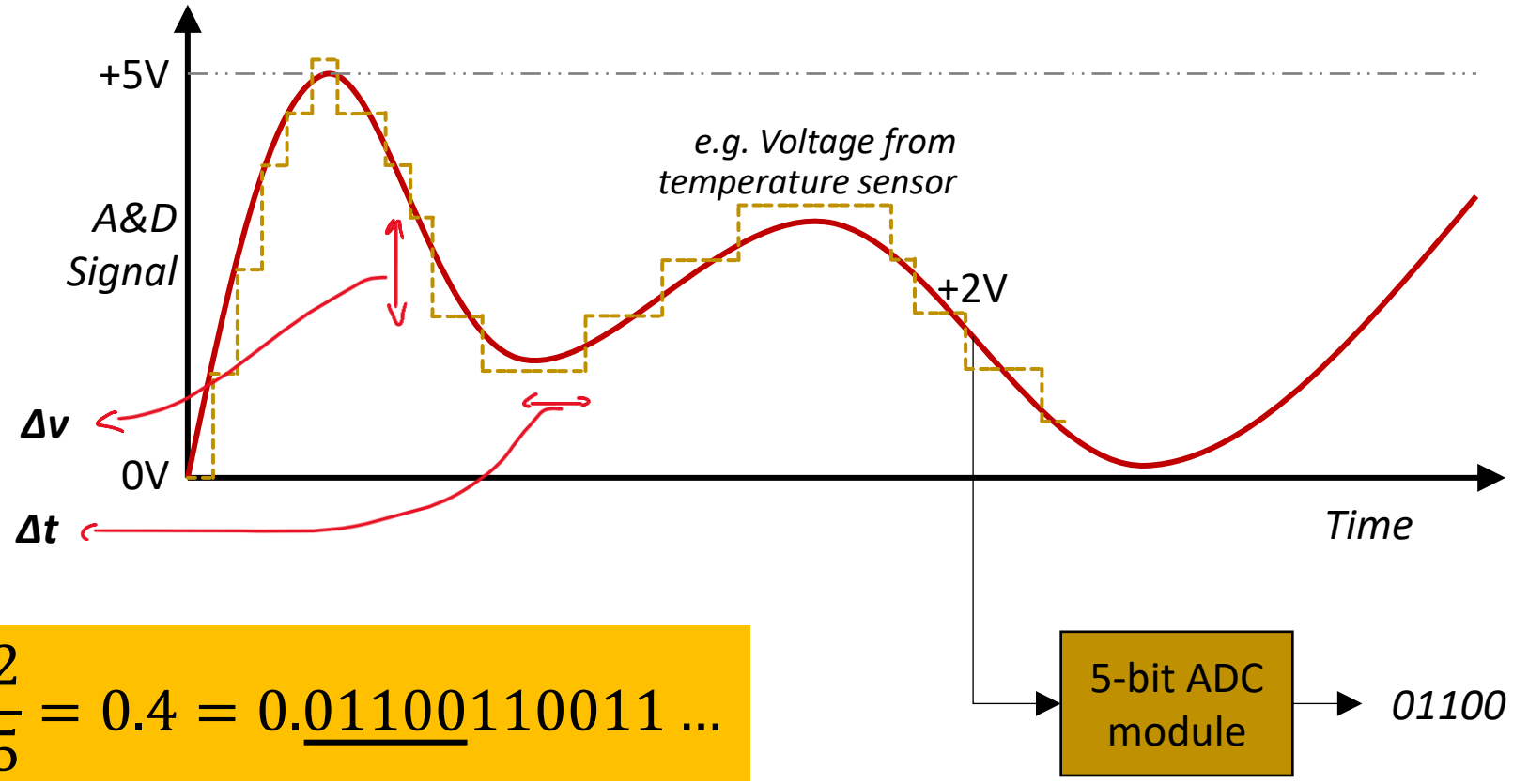


Example of ADC Flow

- Analog-to-Digital-Converter (ADC)
  - For encoding analog signals into digital signals

analog: *continuously valued signal, such as temperature, speed, or voltage with infinite possible values in between*

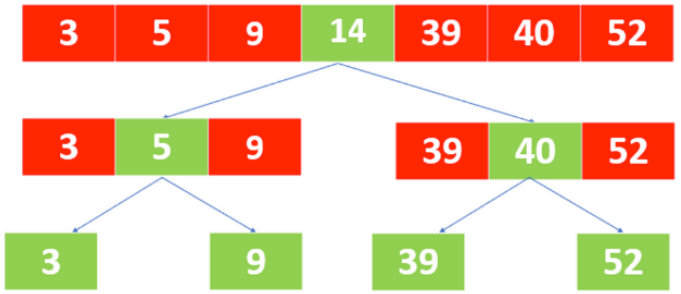
digital: *discretely valued signal, such as integers encoded in binary*



$$= \frac{2}{5} = 0.4 = 0.\underline{01100}110011\ldots$$

ADC Conversion Logic

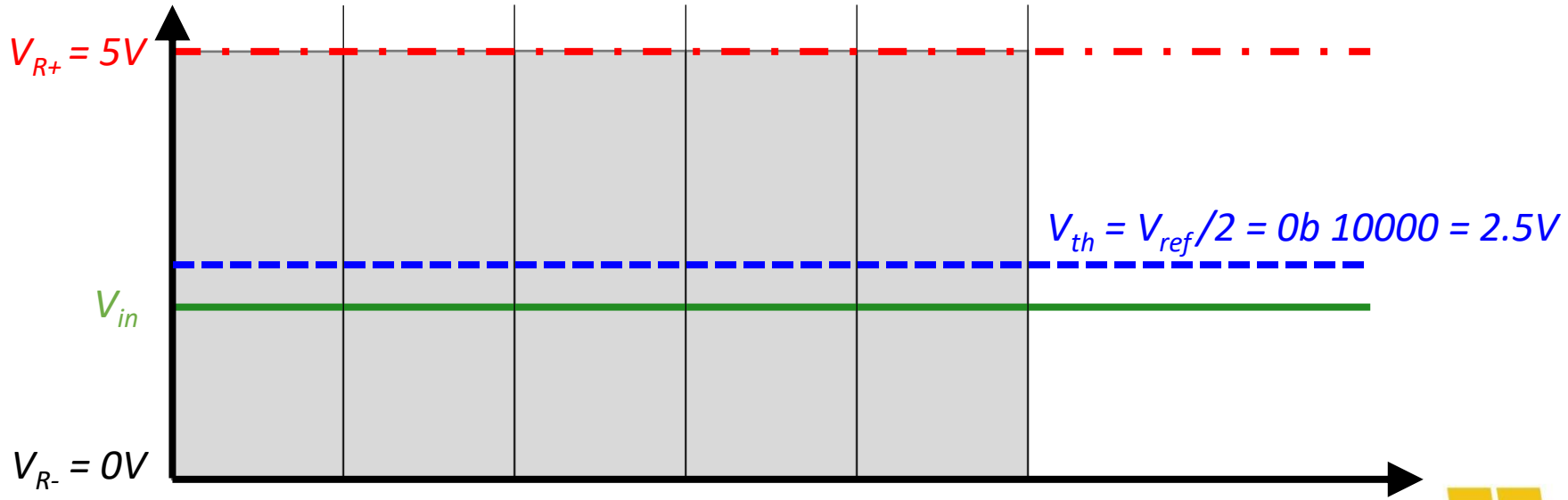
- Moving bit by bit to reach to the closest value
  - Similar to Binary search
- Let's take an example of V_{in} = 2V and V_{ref} = 5V for a 5-bit ADC.



- Now, we set b4
- With b4 = 1
  - V_{in} < threshold
  - We need to clear b4

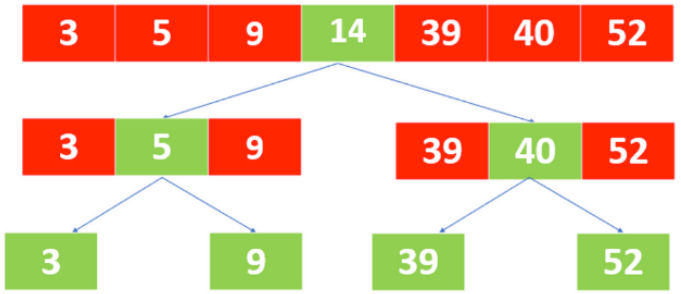
0.5x	0.25x	0.125x	0.0625x	0.03125x
b4 = 1	b3	b2	b1	b0

= 0b 10000



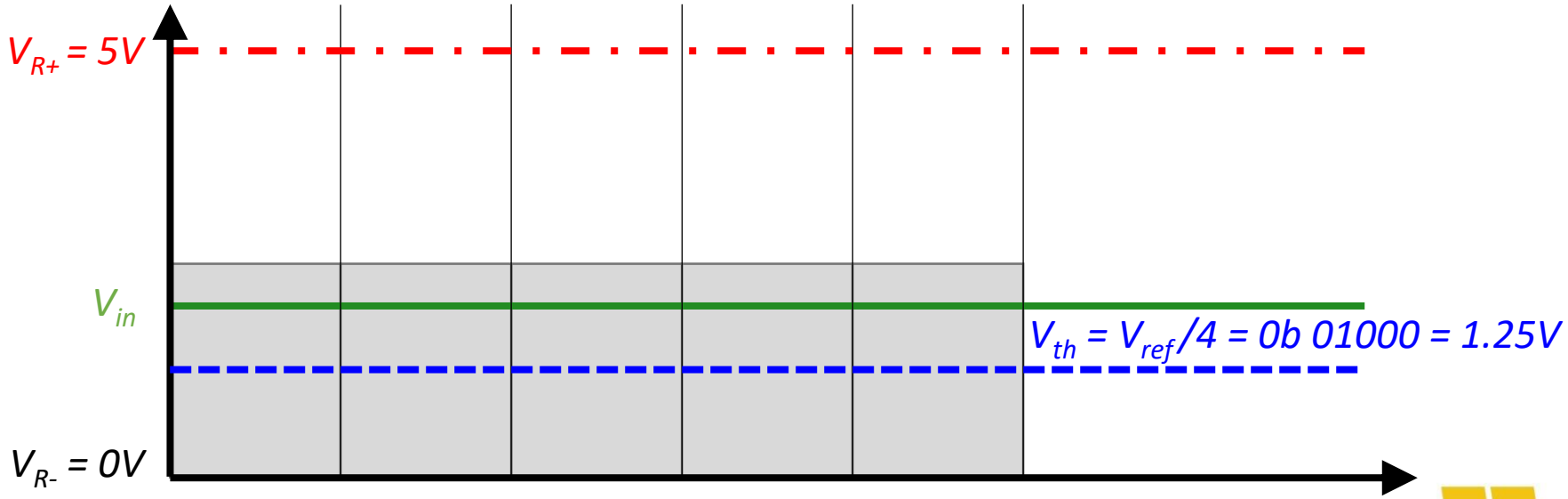
ADC Conversion Logic

- Moving bit by bit to reach to the closest value
  - Similar to Binary search
- Let's take an example of V_{in} = 2V and V_{ref} = 5V for a 5-bit ADC.
  - We clear b4
  - Now, we set b3
  - With b3=1
    - V_{in} > Threshold
    - We keep b3=1



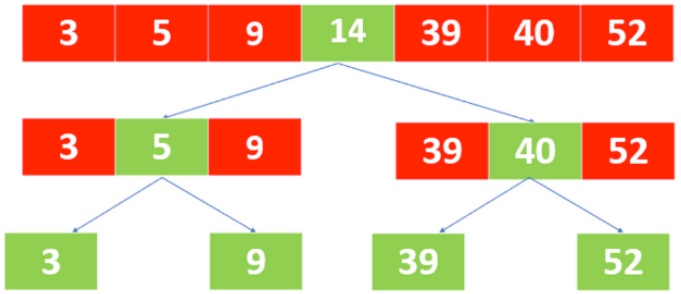
0.5x	0.25x	0.125x	0.0625x	0.03125x
b4 = 0	b3 = 1	b2	b1	b0

= 0b 01000



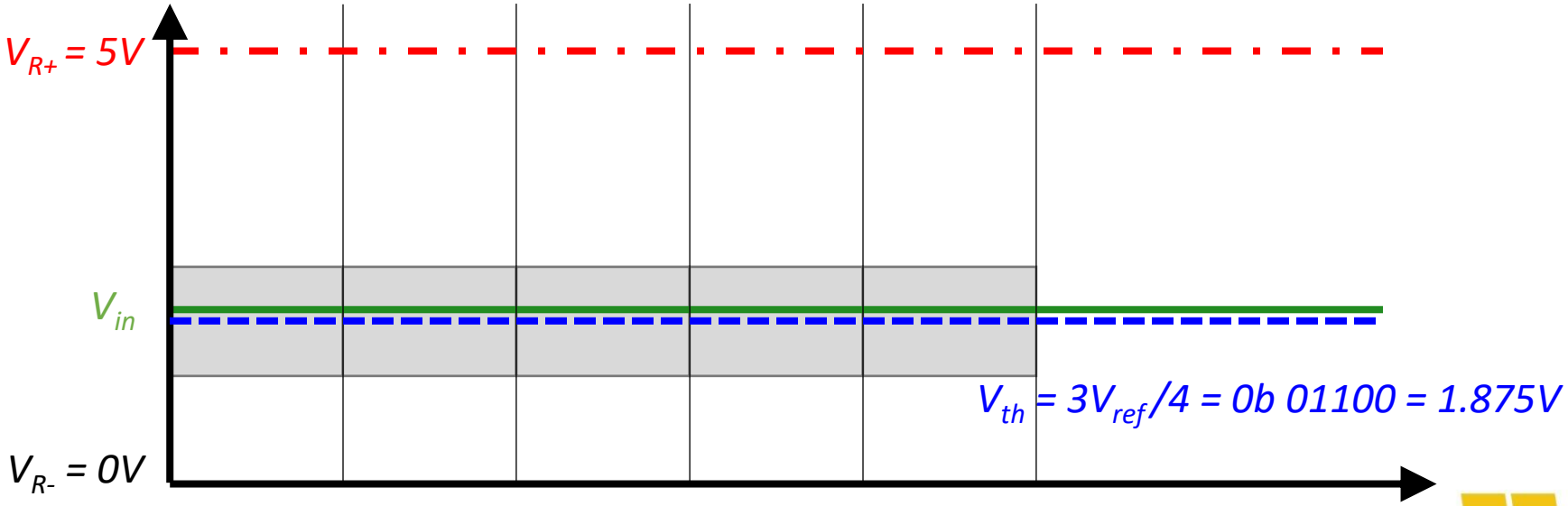
ADC Conversion Logic

- Moving bit by bit to reach to the closest value
  - Similar to Binary search
- Let's take an example of V_{in} = 2V and V_{ref} = 5V for a 5-bit ADC.
  - With b4,b3= 01
  - Now, we set b2
  - With b2=1
    - V_{in} > Threshold
    - We keep b2=1



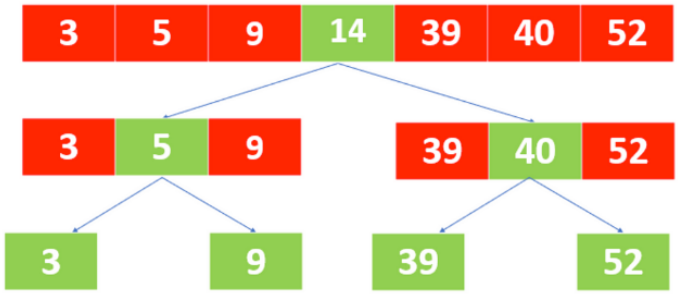
0.5x	0.25x	0.125x	0.0625x	0.03125x
b4 = 0	b3 = 1	b2 = 1	b1	b0

= 0b 01100



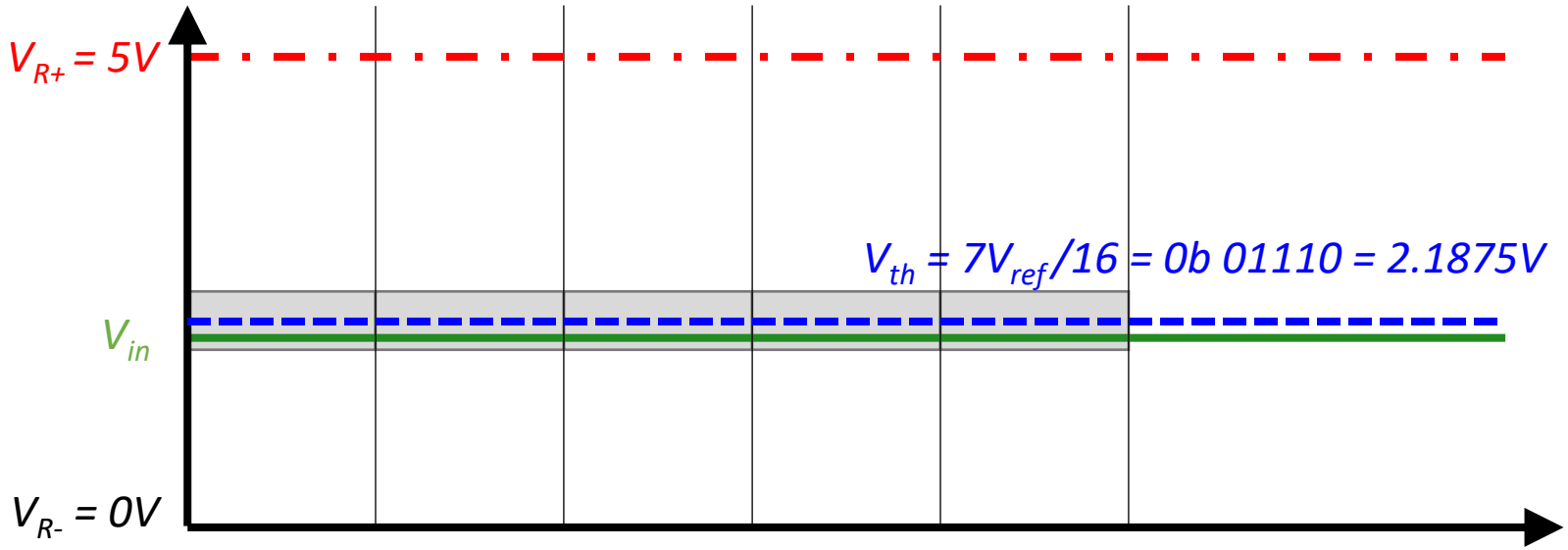
ADC Conversion Logic

- Moving bit by bit to reach to the closest value
  - Similar to Binary search
- Let's take an example of V_{in} = 2V and V_{ref} = 5V for a 5-bit ADC.
  - With b4,b3,b2= 011
  - Now, we set b1
  - With b1=1
    - V_{in} < Threshold
    - We need to clear b1



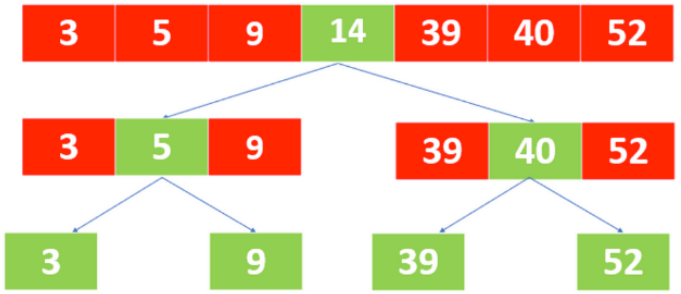
0.5x	0.25x	0.125x	0.0625x	0.03125x
b4 = 0	b3 = 1	b2 = 1	b1 = 1	b0

= 0b 01110



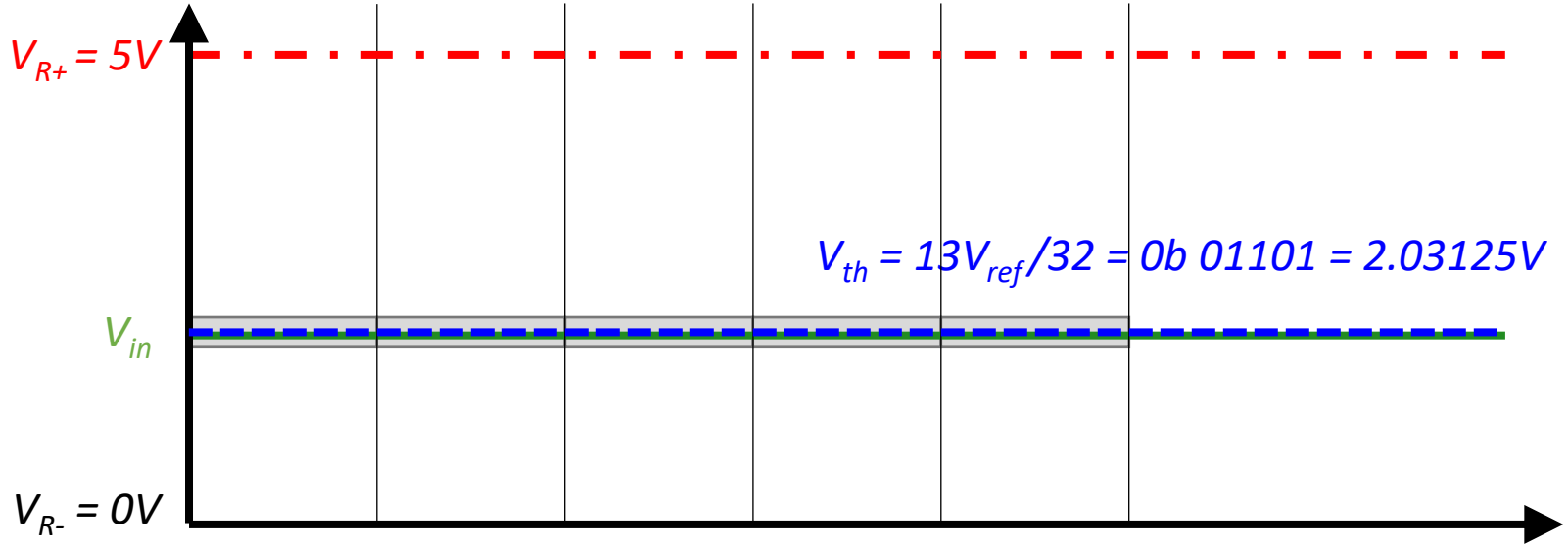
ADC Conversion Logic

- Moving bit by bit to reach to the closest value
  - Similar to Binary search
- Let's take an example of V_{in} = 2V and V_{ref} = 5V for a 5-bit ADC.
  - With b4,b3,b2,b1= 0110
  - Now, we set b0
  - With b0=1
    - V_{in} < Threshold
    - We need to clear b0



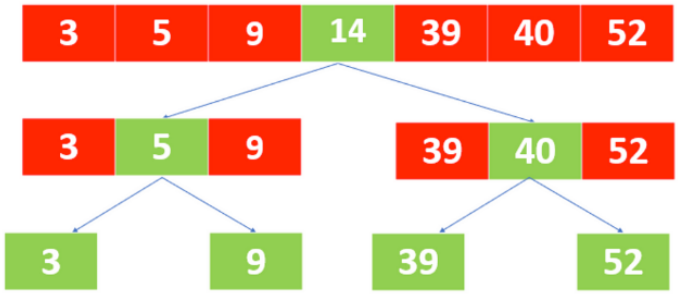
0.5x	0.25x	0.125x	0.0625x	0.03125x
b4 = 0	b3 = 1	b2 = 1	b1 = 0	b0 =1

= 0b 01100



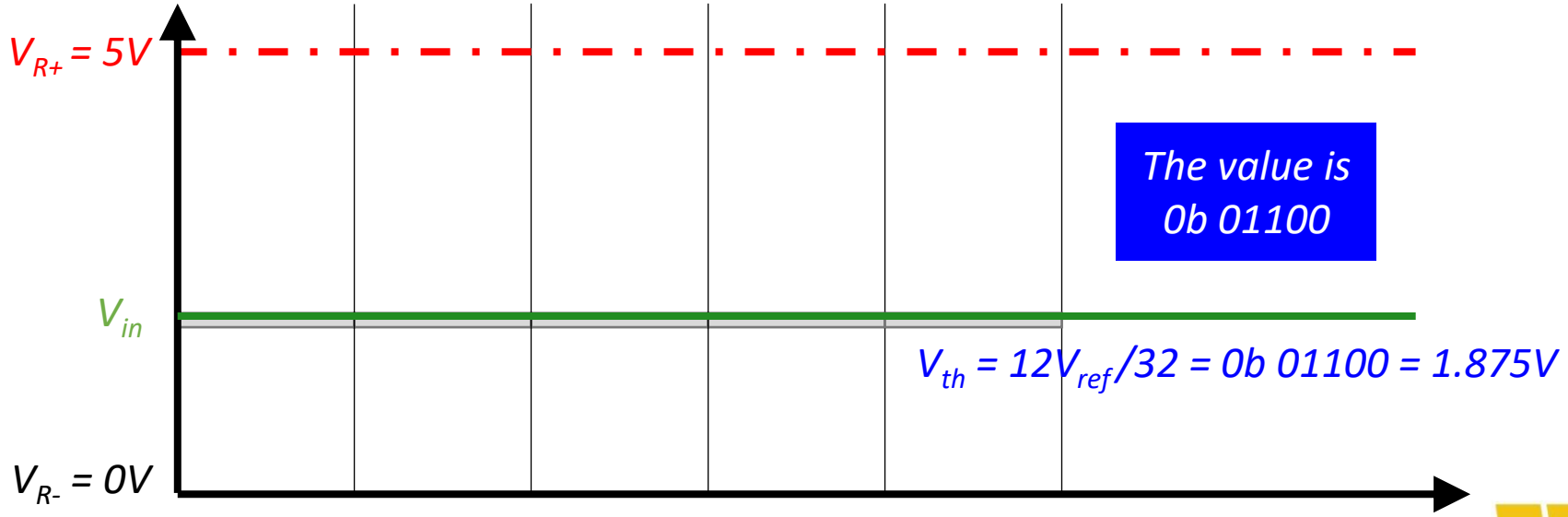
ADC Conversion Logic

- Moving bit by bit to reach to the closest value
  - Similar to Binary search
- Let's take an example of V_{in} = 2V and V_{ref} = 5V for a 5-bit ADC.
  - With b4,b3,b2,b1= 0110
  - Now, we set b0
  - With b0=1
    - V_{in} < Threshold
    - We need to clear b0



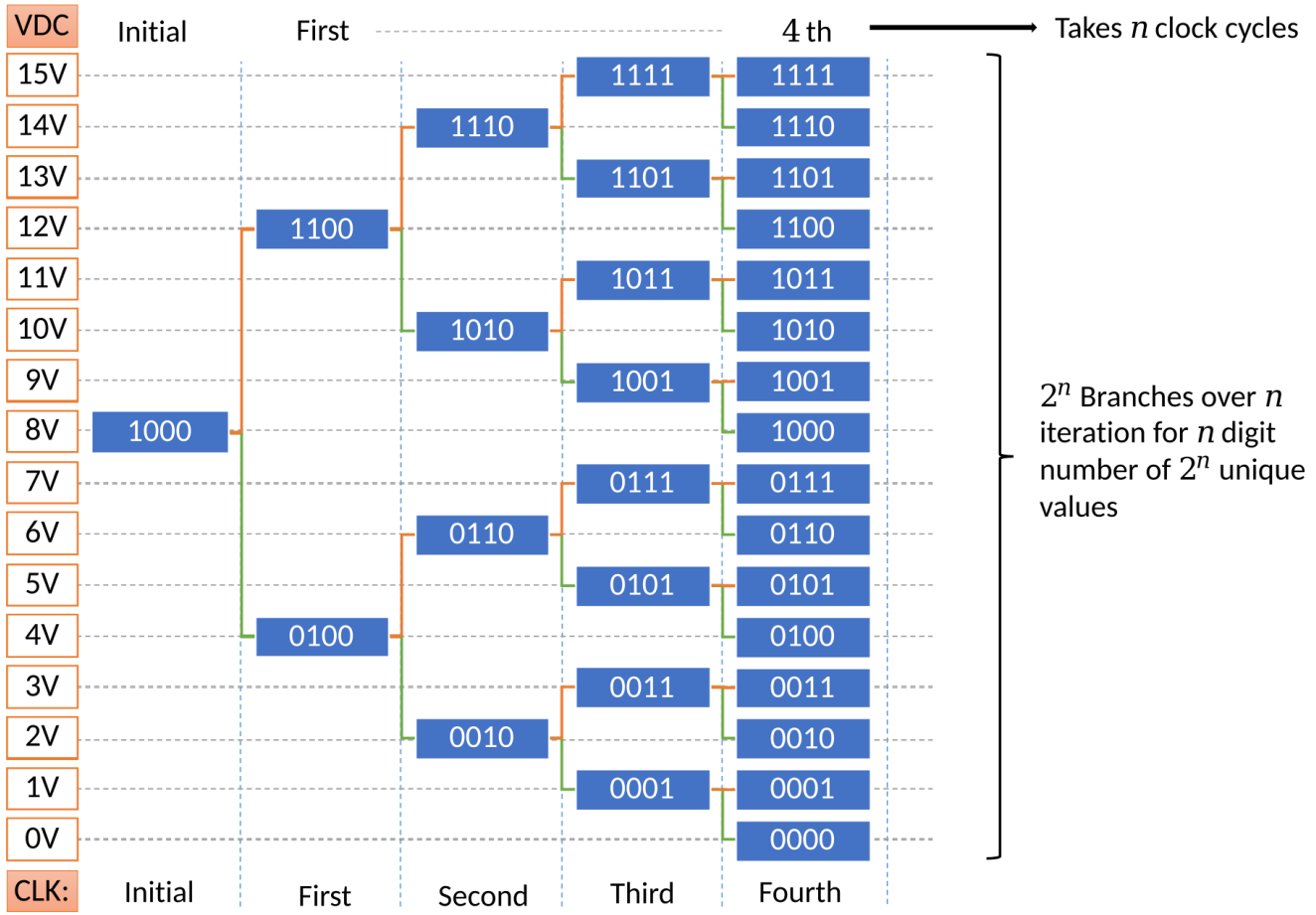
0.5x	0.25x	0.125x	0.0625x	0.03125x
b4 = 0	b3 = 1	b2 = 1	b1 = 0	b0 =0

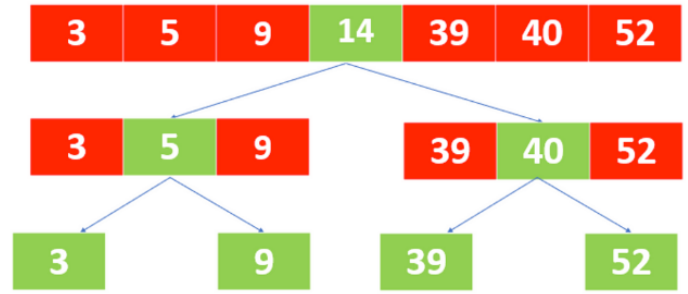
= 0b 01100



ADC Conversion Logic

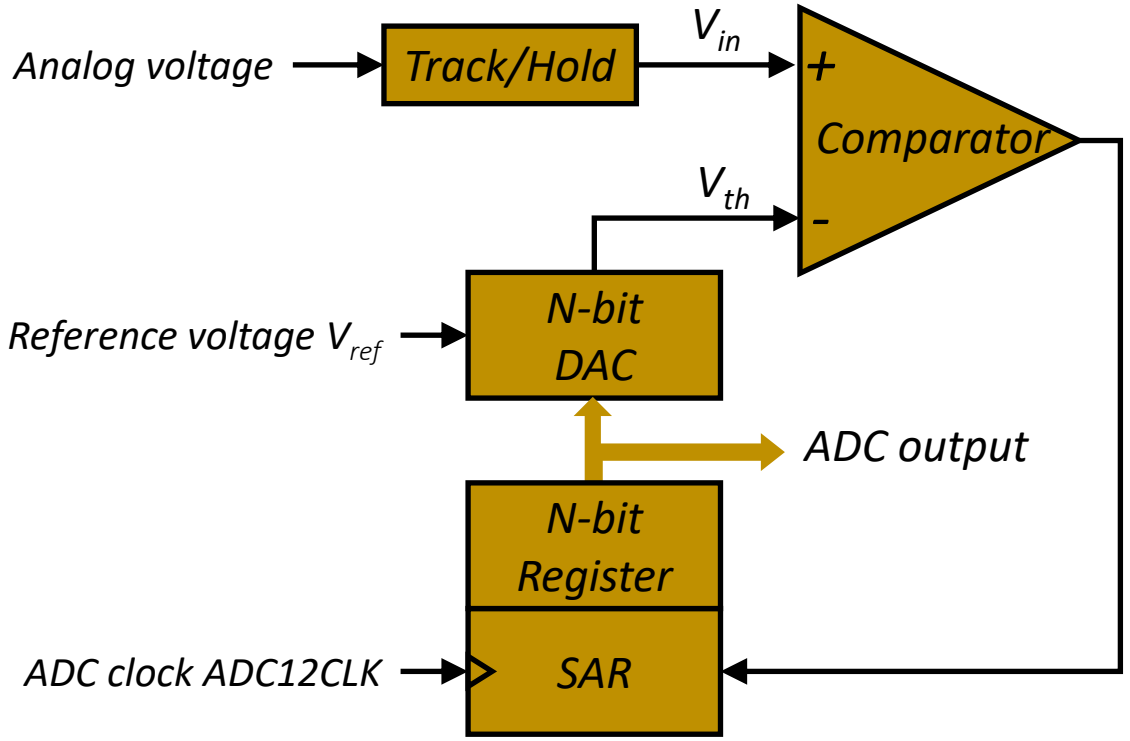
- Moving bit by bit to reach to the closest value
  - Similar to Binary search





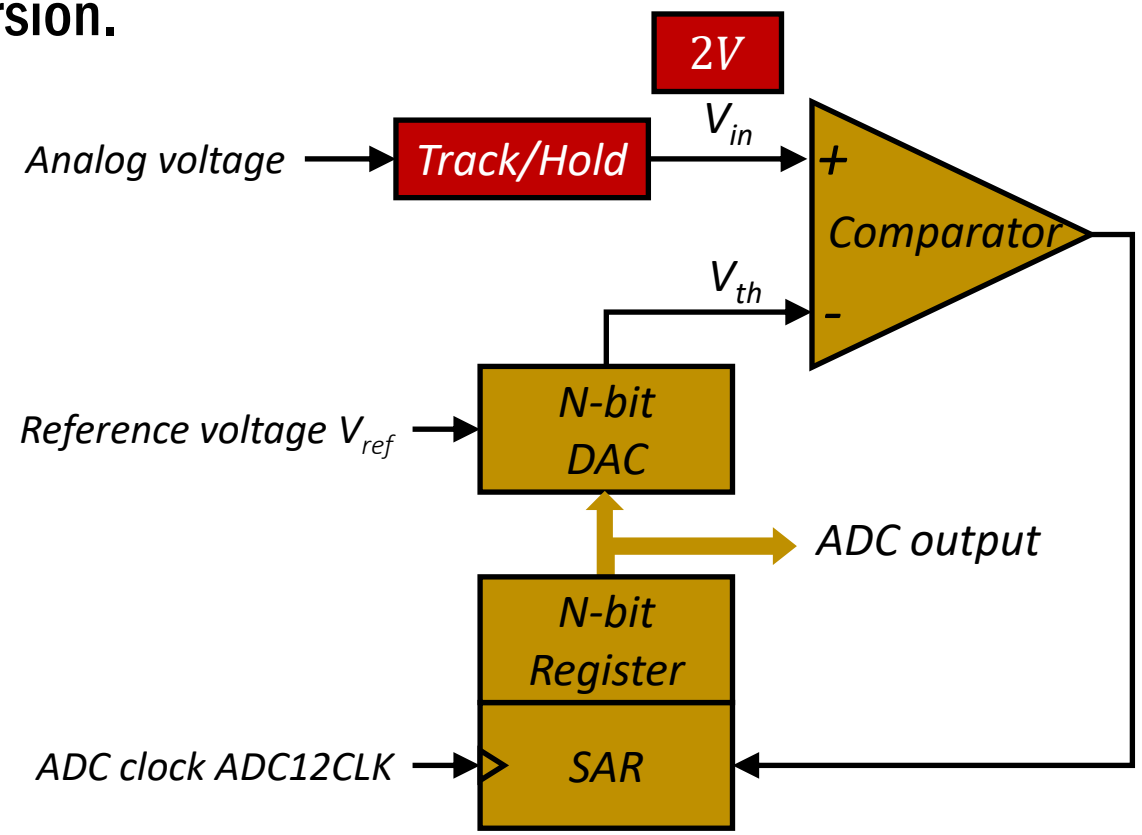
Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - The analog voltage is sampled and stored for conversion.
  - V_{in} is the input voltage to the ADC module.



Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - The analog voltage is sampled and stored for conversion.
  - V_{in} is the input voltage to the ADC module.
- Let's take an example of V_{in} = 2V and V_{ref} = 5V for a 5-bit ADC.

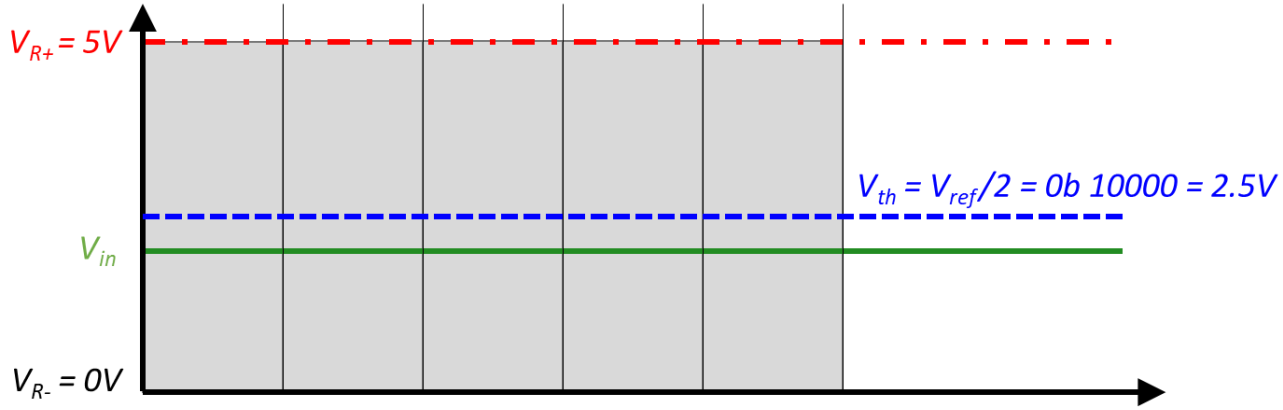


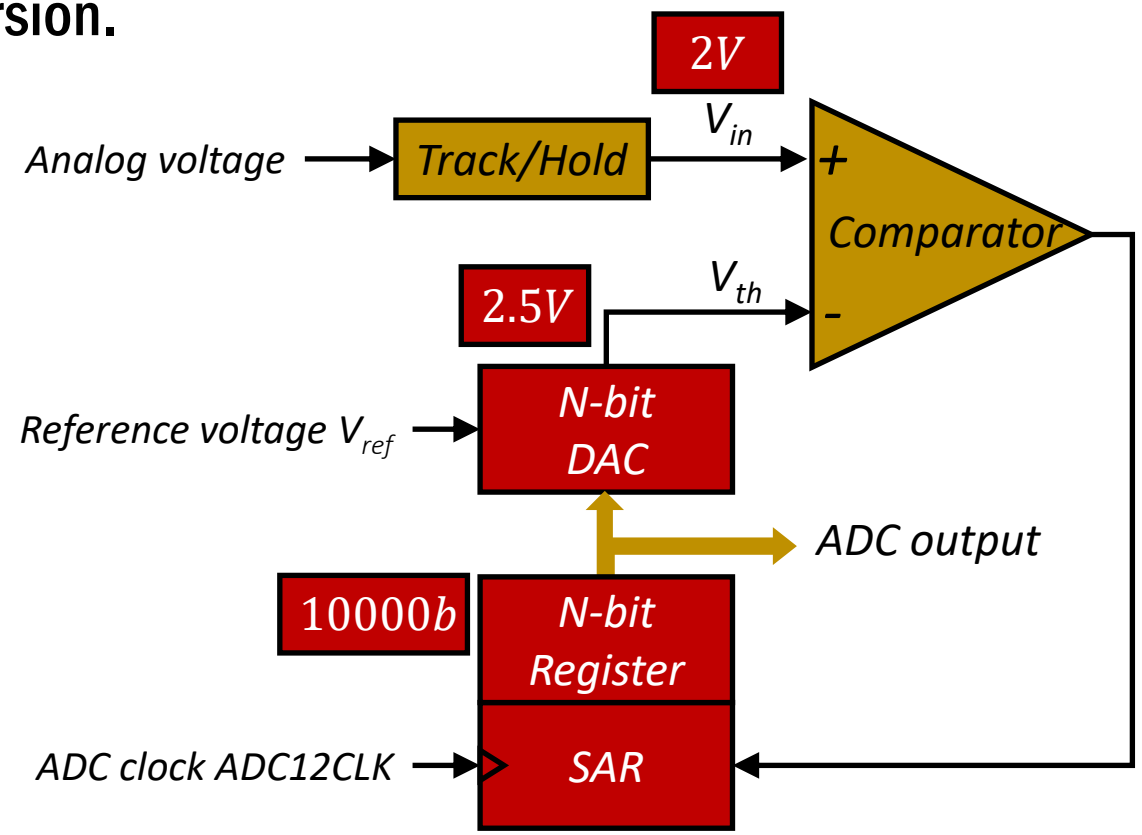
Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - The analog voltage is sampled and stored for conversion.
  - V_{in} is the input voltage to the ADC module.

The SAR is initialized to the first estimate 10000b (b4 = 1)
The corresponding voltage V_{th} is output by the DAC.

0.5x	0.25x	0.125x	0.0625x	0.03125x	
b4 = 1	b3	b2	b1	b0	= 0b 10000



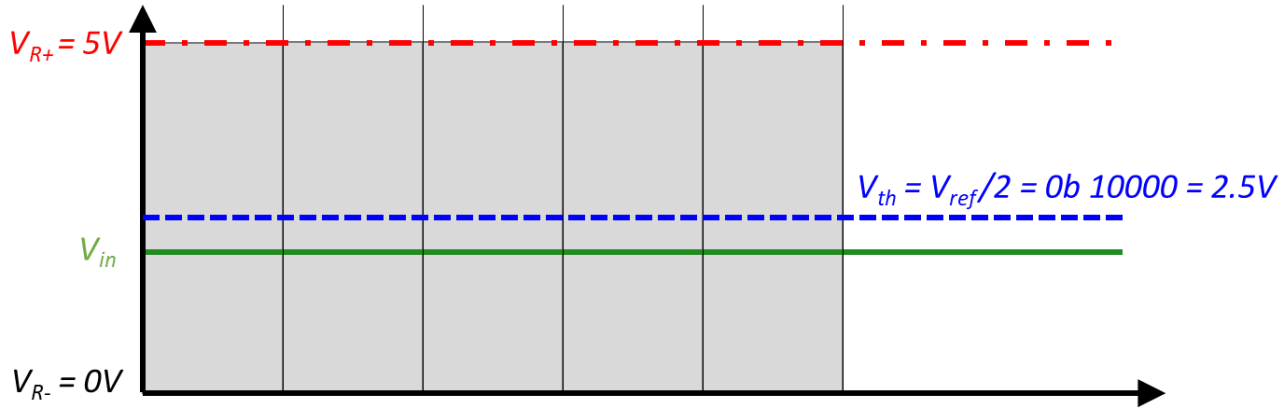


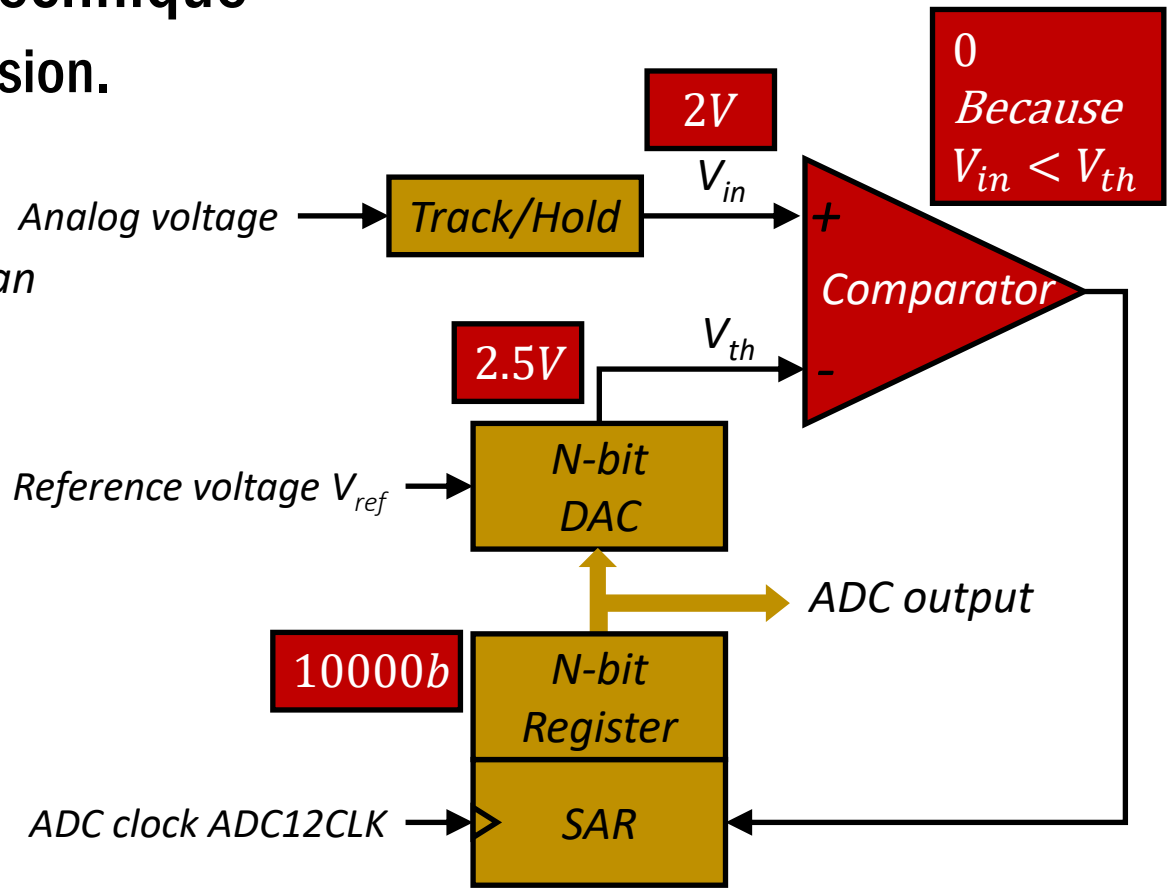
Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - The analog voltage is sampled and stored for conversion.
  - V_{in} is the input voltage to the ADC module.

The comparator provides a high output if the V_{in} is greater than V_{out}, else it provides a low output.

0.5x	0.25x	0.125x	0.0625x	0.03125x	
b4 = 1	b3	b2	b1	b0	= 0b 10000



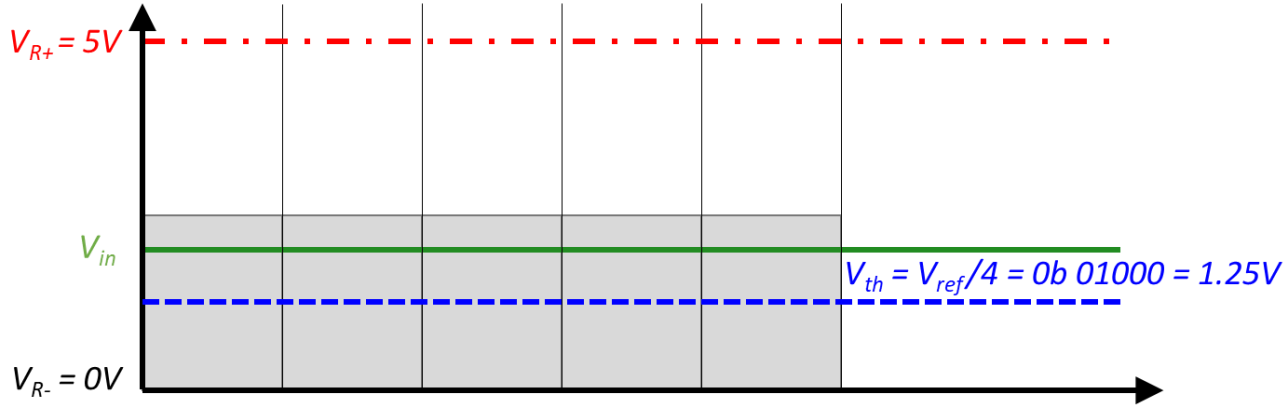


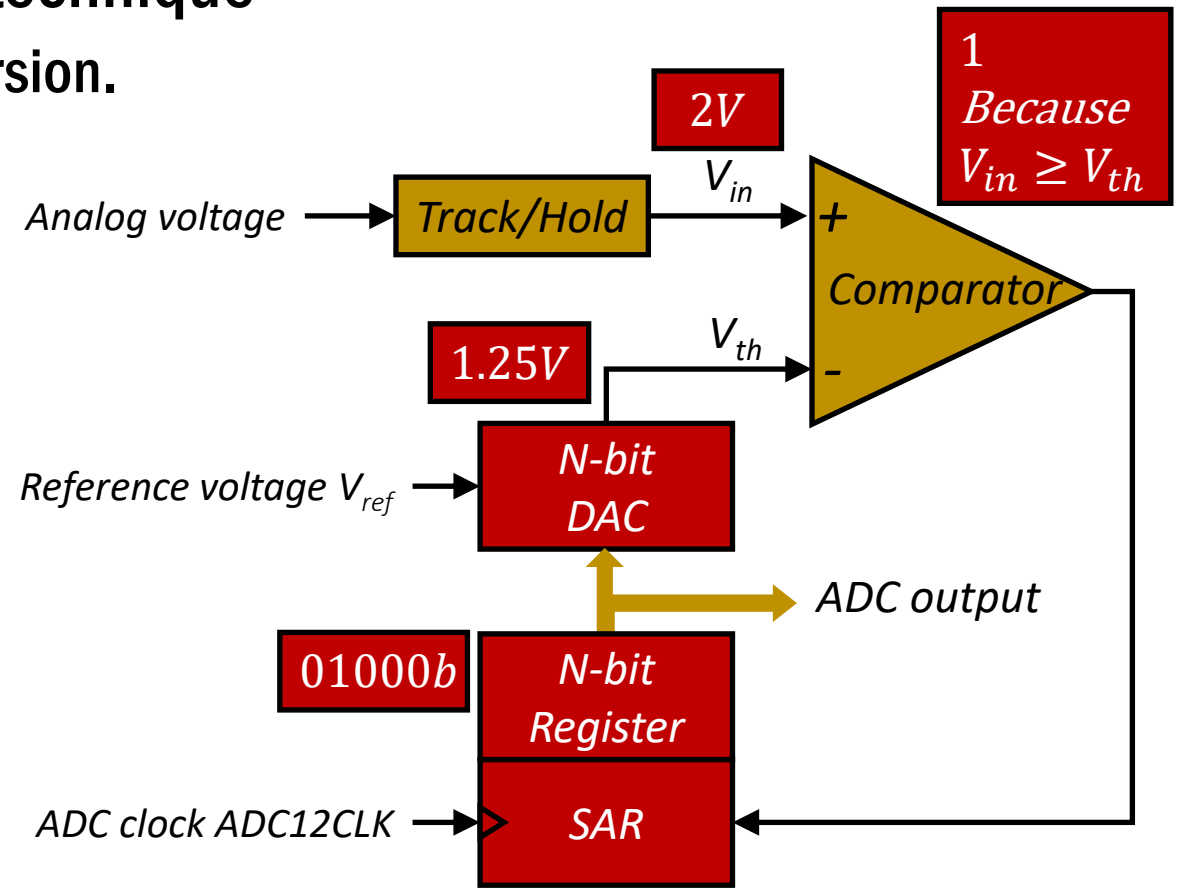
Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - The analog voltage is sampled and stored for conversion.
  - V_{in} is the input voltage to the ADC module.

The SAR is updated based on the comparator output and the voltage V_{th} is updated.

0.5x	0.25x	0.125x	0.0625x	0.03125x	
b4 = 0	b3 = 1	b2	b1	b0	= 0b 01000



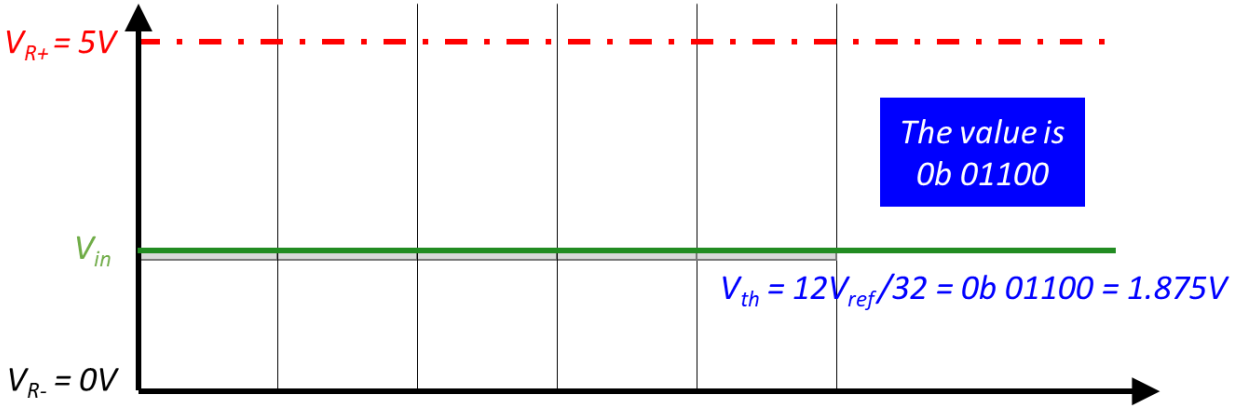


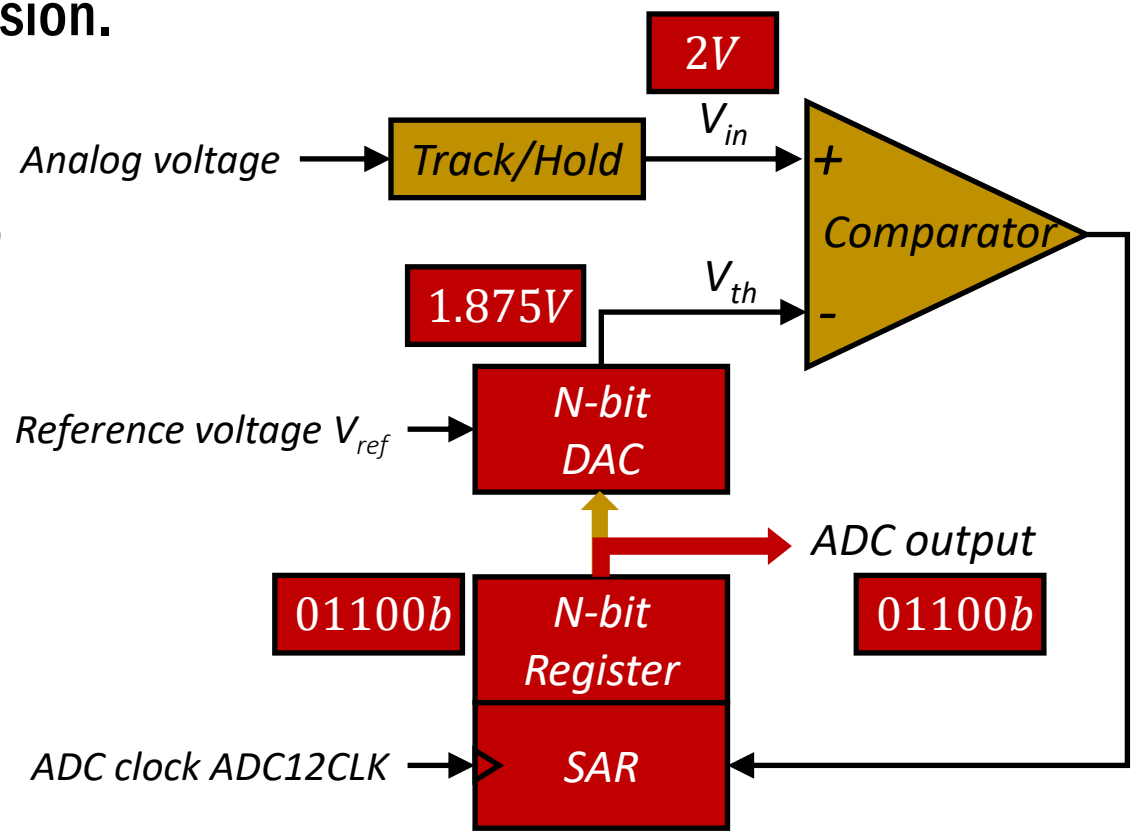
Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - The analog voltage is sampled and stored for conversion.
  - V_{in} is the input voltage to the ADC module.

The SAR block performs the ADC conversion logic n-times for n-bits. Then, the n-bit digital value corresponding to the input voltage V_{in} is output by the ADC.

0.5x	0.25x	0.125x	0.0625x	0.03125x	
b4 = 0	b3 = 1	b2 = 1	b1 = 0	b0 =0	= 0b 01100

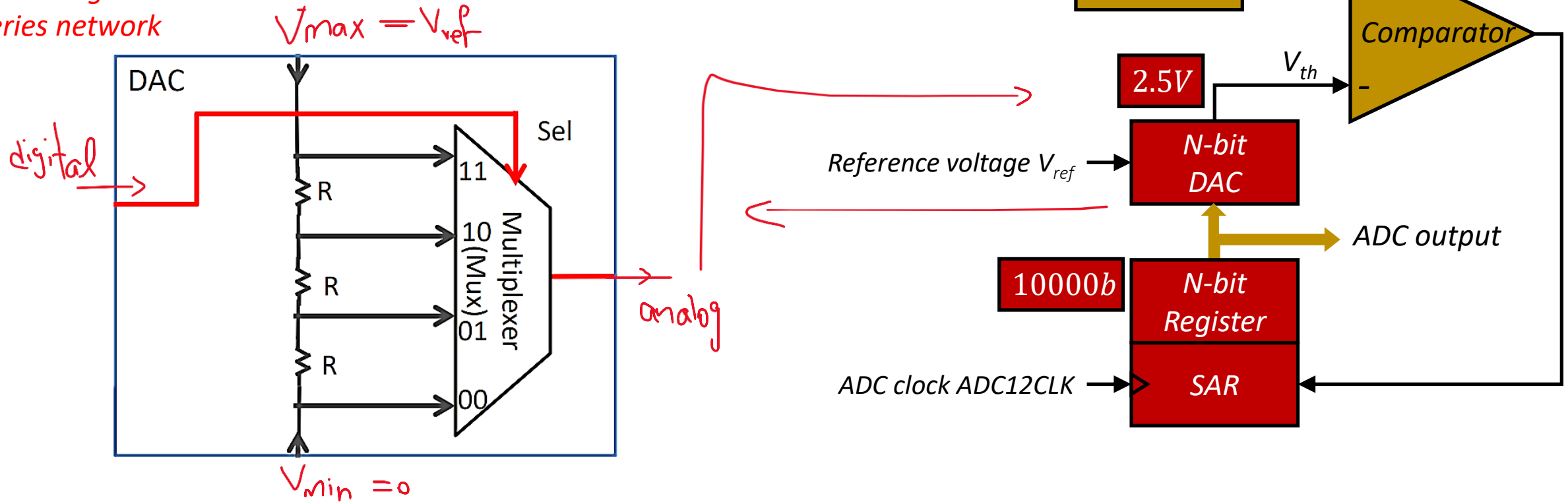




Signal Conversion in ADC

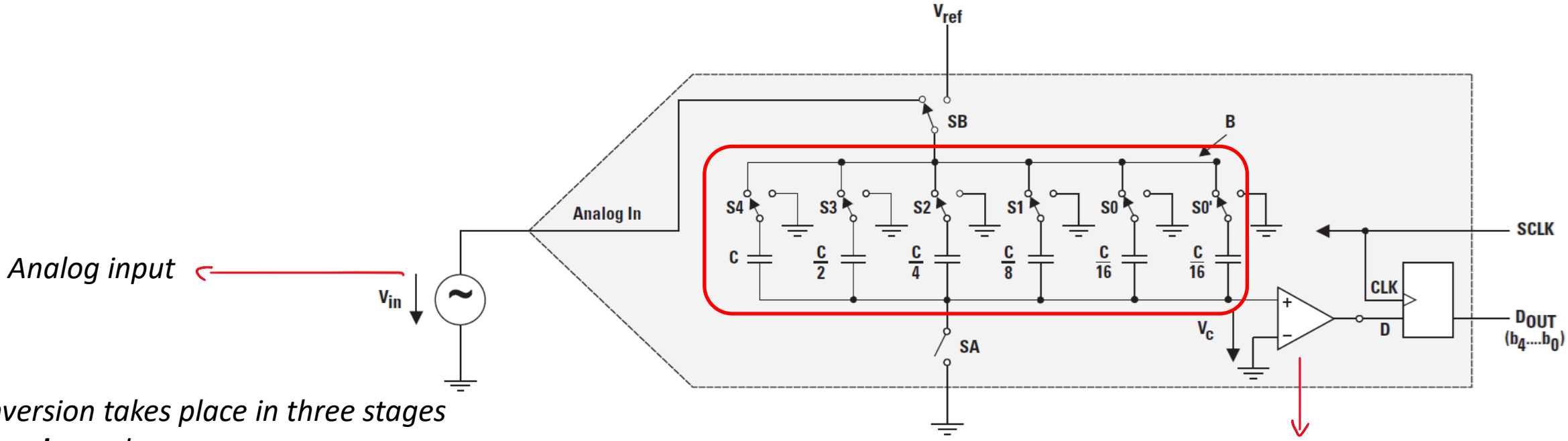
- Using Successive approximation register (SAR) technique
  - The analog voltage is sampled and stored for conversion.
  - DAC for converting digital to V_{ref} is required

DAC using R series network



Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - This is a charge-redistribution SAR



The conversion takes place in three stages

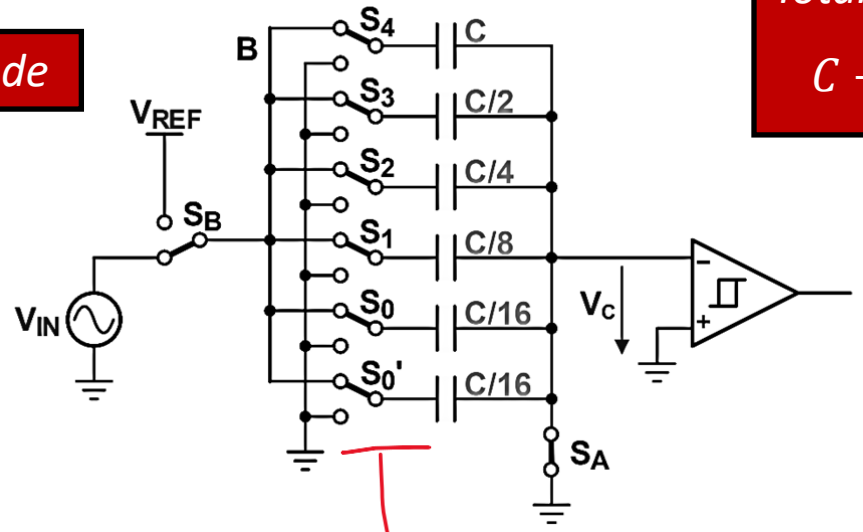
1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode*

Op-amp configured as comparator

Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - This is a charge-redistribution SAR

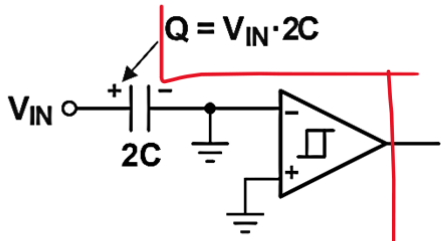
sample mode



Total C of capacitors in parallel =

$$C + \frac{C}{2} + \frac{C}{4} + \frac{C}{8} + \frac{C}{16} + \frac{C}{16} = 2C$$

≙



Switches are all connected to V_{IN}.

The total charge would be V_{IN}.2C

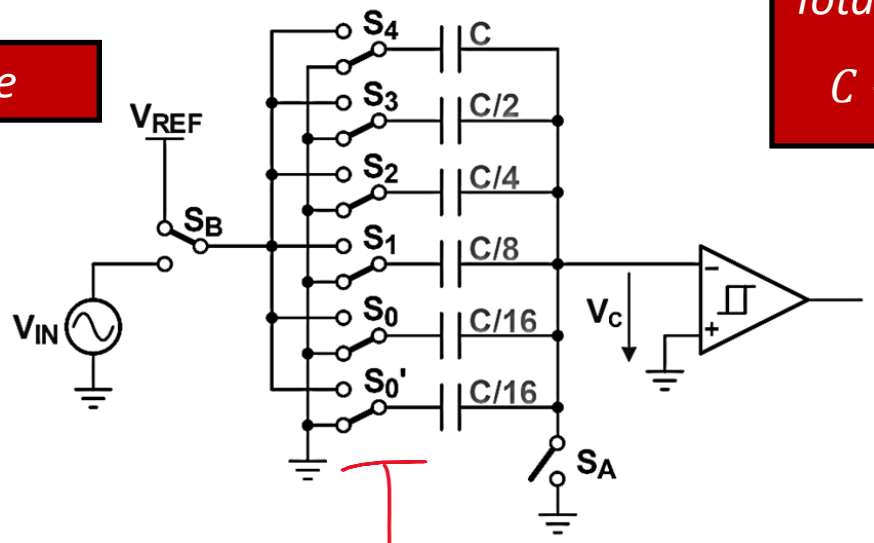
The conversion takes place in three stages

1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode*

Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - This is a charge-redistribution SAR

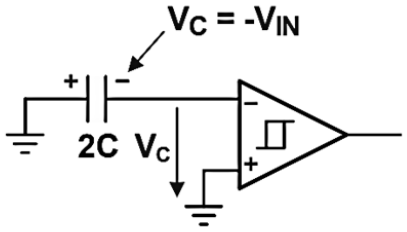
Hold mode



Total C of capacitors in parallel =

$$C + \frac{C}{2} + \frac{C}{4} + \frac{C}{8} + \frac{C}{16} + \frac{C}{16} = 2C$$

≙



The conversion takes place in three stages

1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode*

Switches are all connected to GND.
Holding V_{IN} (at the inverting comparator input)
$VC = 0 - V_{IN} = -V_{IN}$

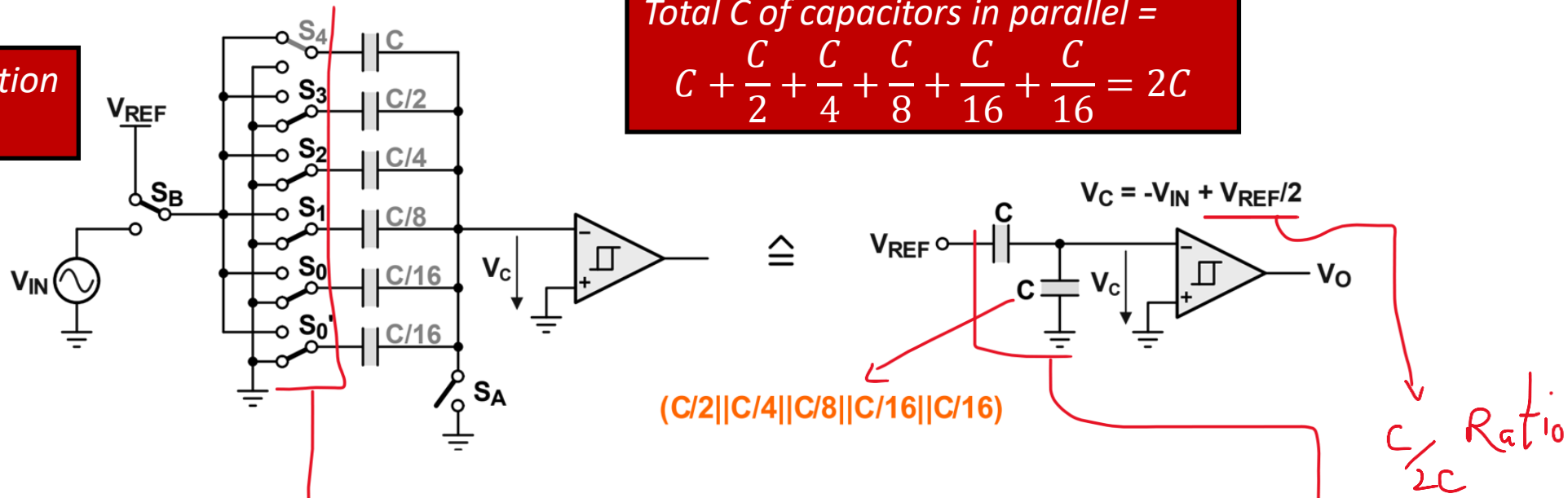
Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - This is a charge-redistribution SAR

Redistribution mode

Total C of capacitors in parallel =

$$C + \frac{C}{2} + \frac{C}{4} + \frac{C}{8} + \frac{C}{16} + \frac{C}{16} = 2C$$



The conversion takes place in three stages

1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode*

One-by-one connecting the switches (here it is for S_4).

Capacitor C forms a 1:1 capacitance divider with the remaining capacitors connected to ground.

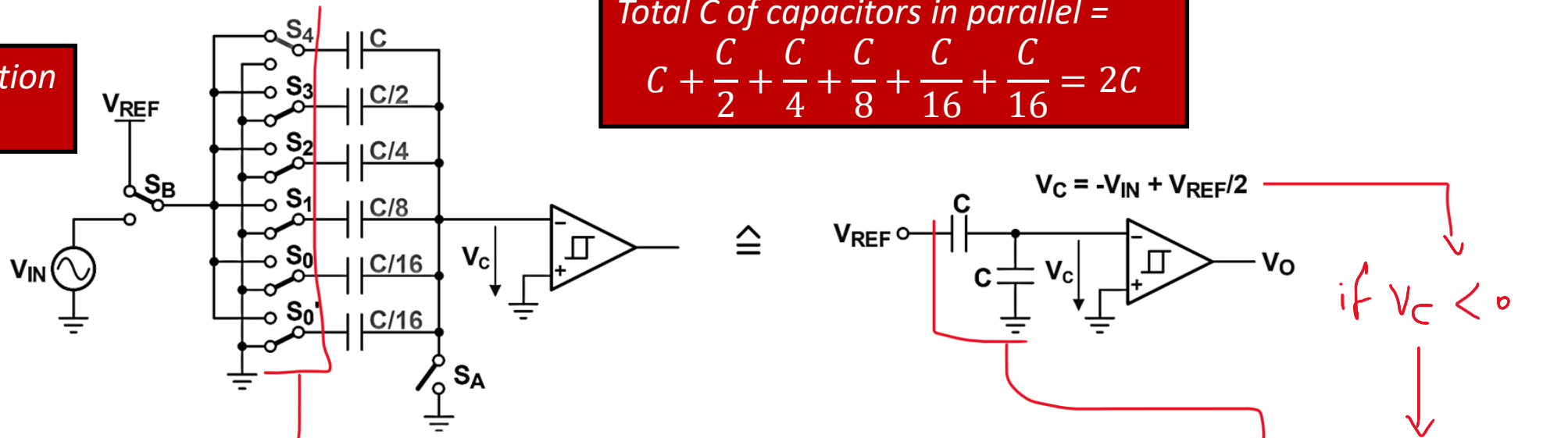
Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - This is a charge-redistribution SAR

Redistribution mode

Total C of capacitors in parallel =

$$C + \frac{C}{2} + \frac{C}{4} + \frac{C}{8} + \frac{C}{16} + \frac{C}{16} = 2C$$



The conversion takes place in three stages

1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode*

One-by-one connecting the switches (here it is for S_4).

Capacitor C forms a 1:1 capacitance divider with the remaining capacitors connected to ground.

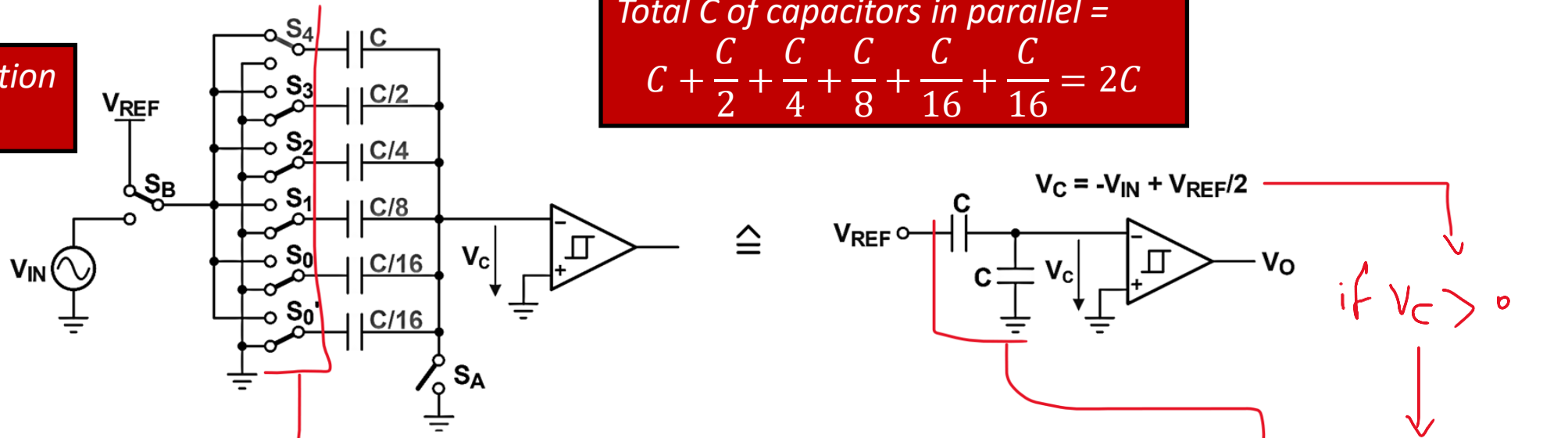
Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - This is a charge-redistribution SAR

Redistribution mode

Total C of capacitors in parallel =

$$C + \frac{C}{2} + \frac{C}{4} + \frac{C}{8} + \frac{C}{16} + \frac{C}{16} = 2C$$



The conversion takes place in three stages

1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode*

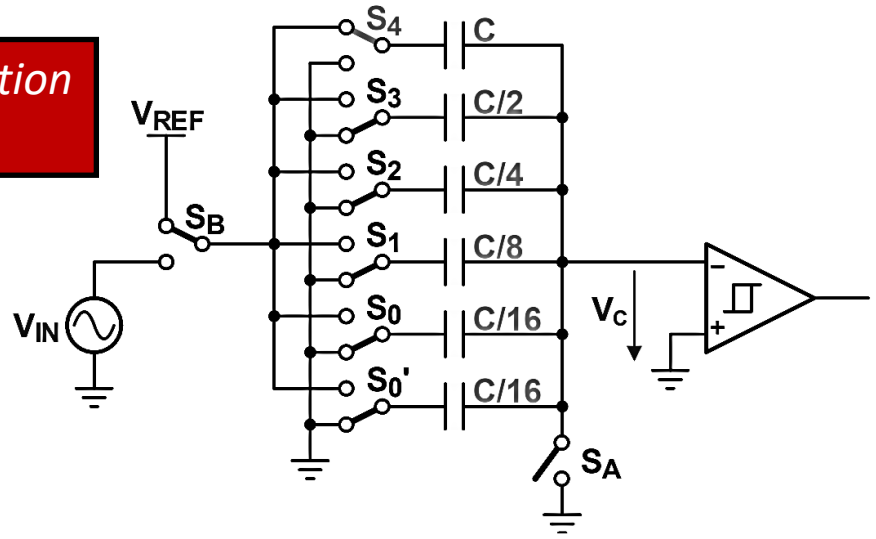
One-by-one connecting the switches (here it is for S_4).

Capacitor C forms a 1:1 capacitance divider with the remaining capacitors connected to ground.

Signal Conversion in ADC

- Using Successive approximation register (SAR) technique
  - This is a charge-redistribution SAR

Redistribution mode



Procedure
$SB = V_{ref};$
$SA = open;$
For capacitor in $\{S_4, S_3, S_2, S_1, S_0\}$
Connect capacitor to V_{ref}.
If Comparator output is high
Connect capacitor to gnd.
Else
Connect capacitor to V_{ref}.

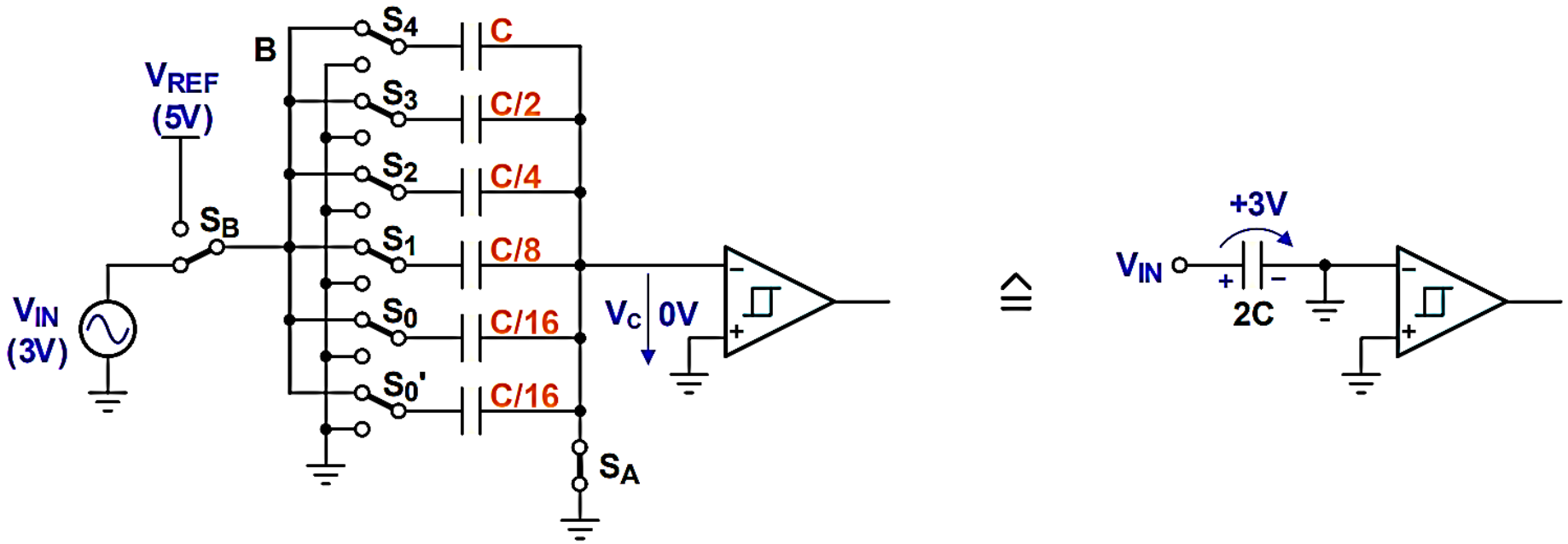
The conversion takes place in three stages

1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode*

Comparator output	Capacitor status
High ($V_{IN} < V_{ref}$)	Retain (connect to V_{ref})
Low ($V_{IN} > V_{ref}$)	Discharge (connect to ground)

Signal Conversion in ADC - Example

- With the same 5-bit SAR-ADC
  - Assuming that V_{IN} = 3V and V_{ref} = 5V

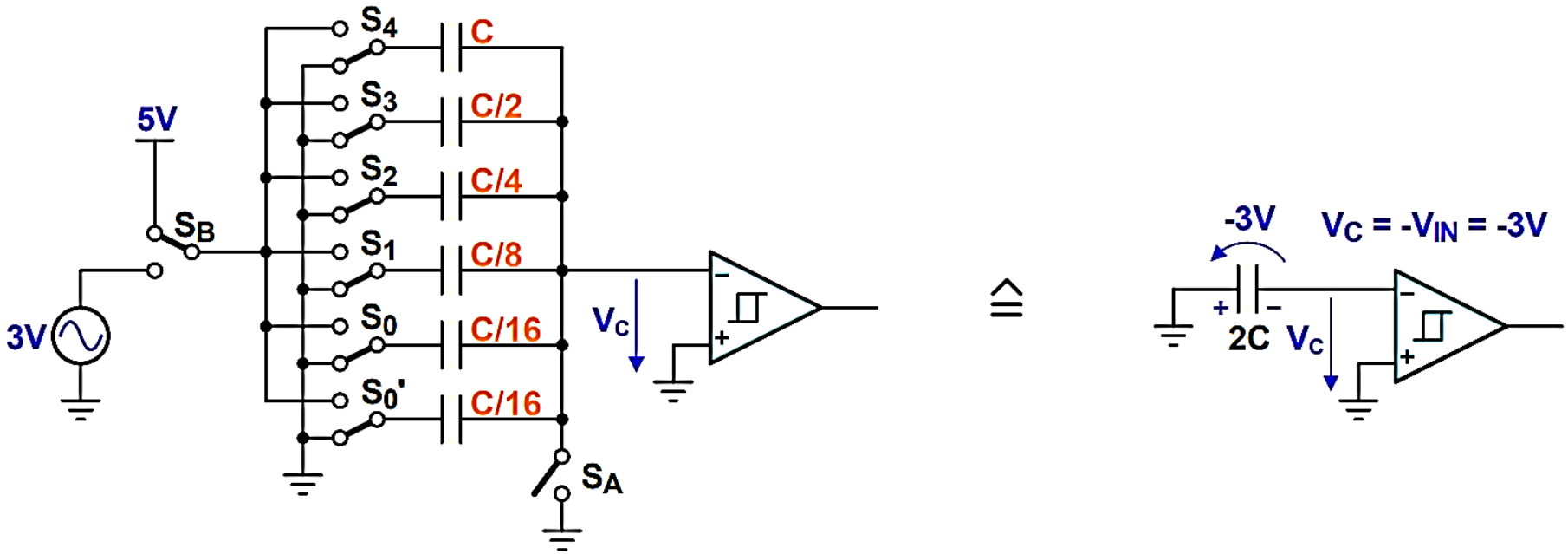


The conversion takes place in three stages

1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode*

Signal Conversion in ADC - Example

- With the same 5-bit SAR-ADC
  - Assuming that V_{IN} = 3V and V_{ref} = 5V

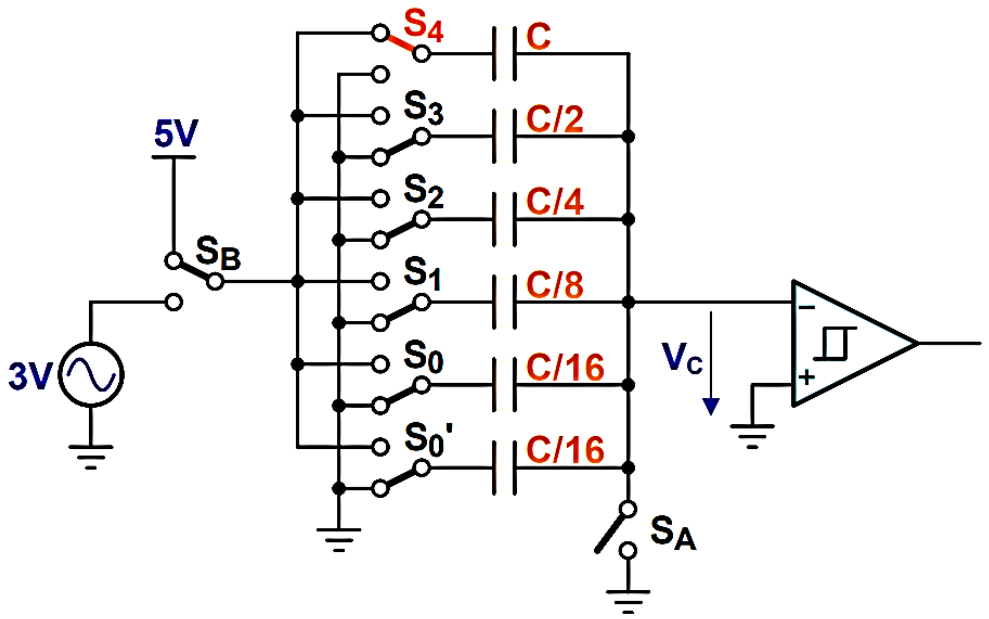


The conversion takes place in three stages

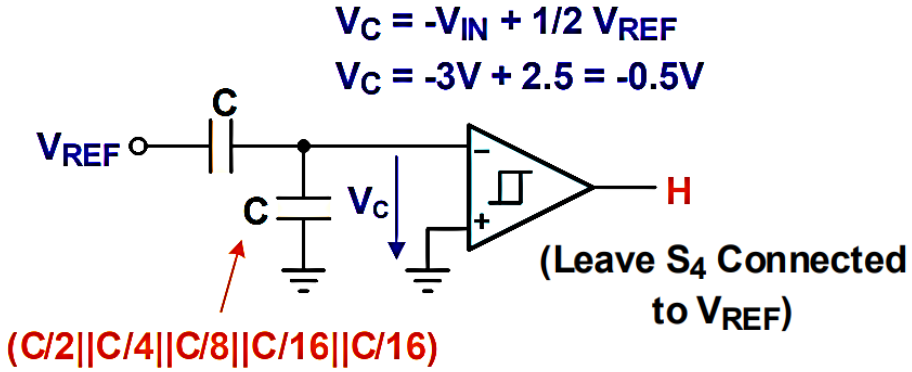
1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode*

Signal Conversion in ADC - Example

- With the same 5-bit SAR-ADC
  - Assuming that V_{IN} = 3V and V_{ref} = 5V



≙

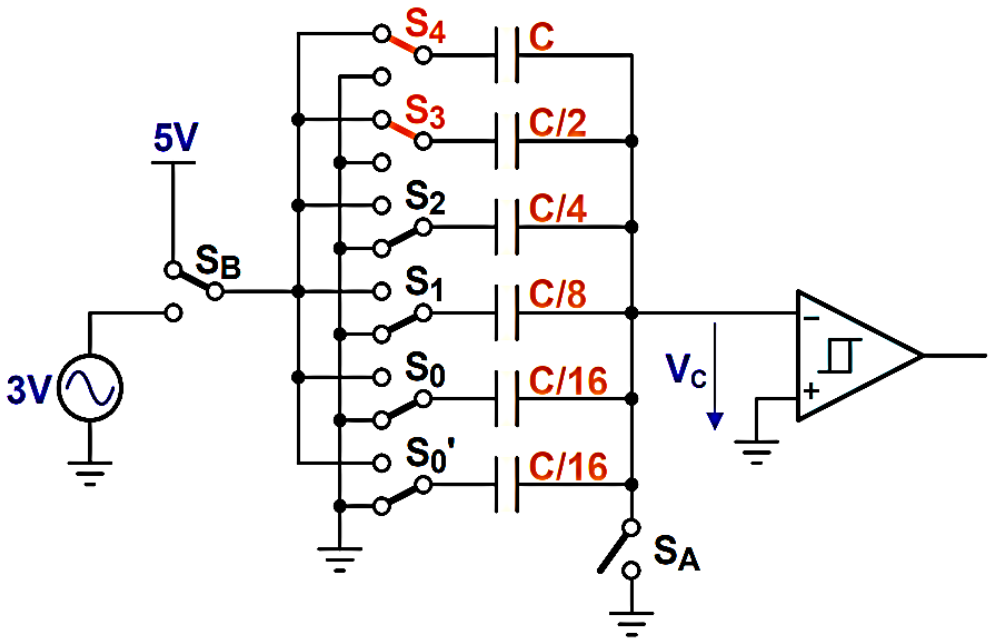


The conversion takes place in three stages

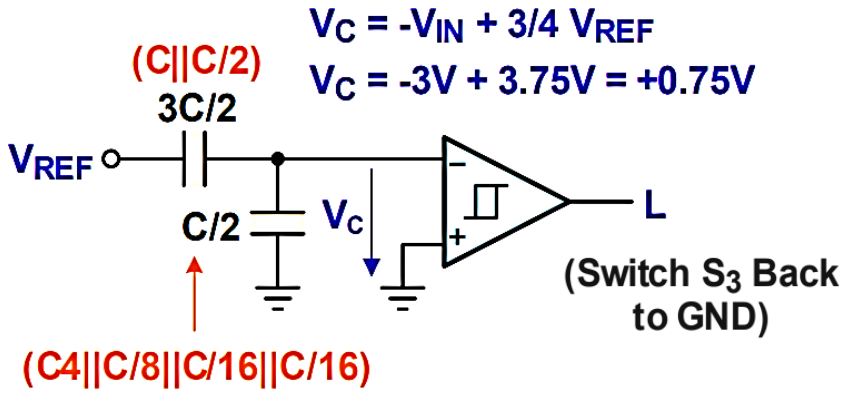
1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode (bit 4)*

Signal Conversion in ADC - Example

- With the same 5-bit SAR-ADC
  - Assuming that V_{IN} = 3V and V_{ref} = 5V



≙

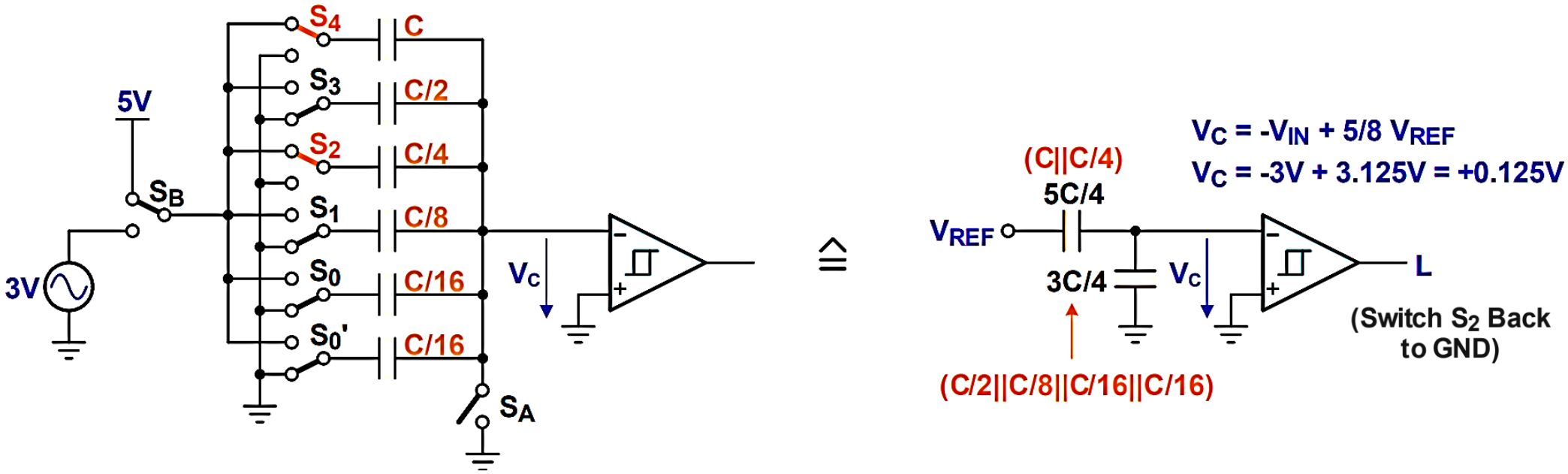


The conversion takes place in three stages

1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode (bit 3)*

Signal Conversion in ADC - Example

- With the same 5-bit SAR-ADC
  - Assuming that V_{IN} = 3V and V_{ref} = 5V

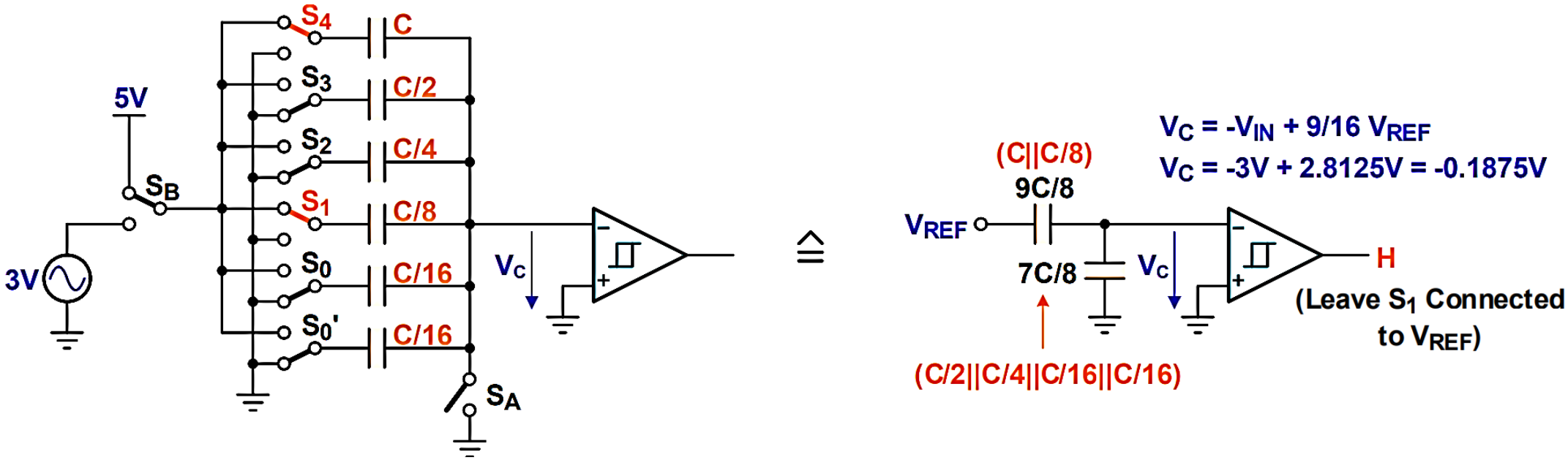


The conversion takes place in three stages

1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode (bit 2)*

Signal Conversion in ADC - Example

- With the same 5-bit SAR-ADC
  - Assuming that V_{IN} = 3V and V_{ref} = 5V

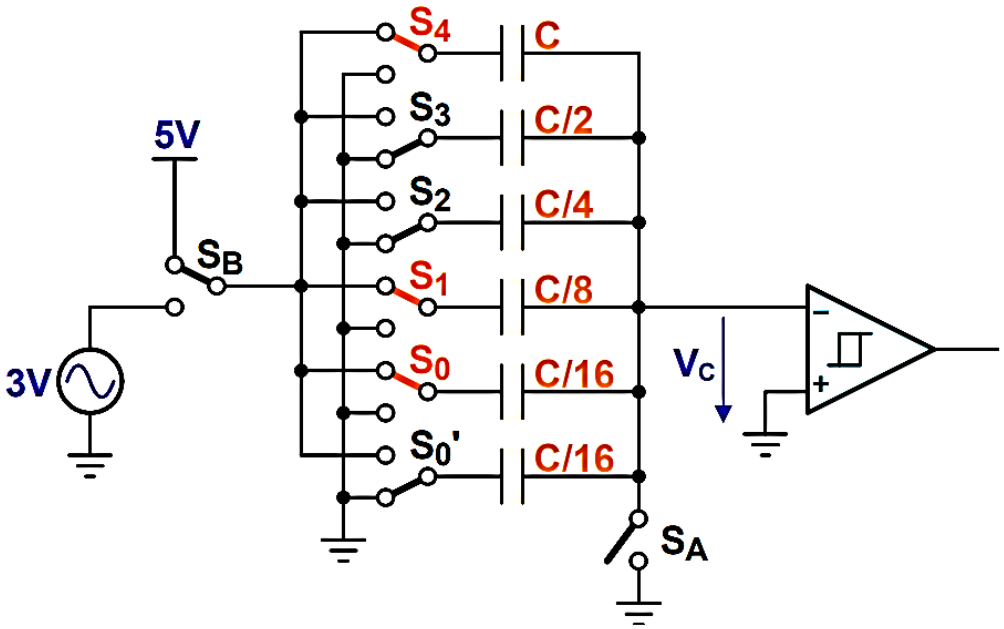


The conversion takes place in three stages

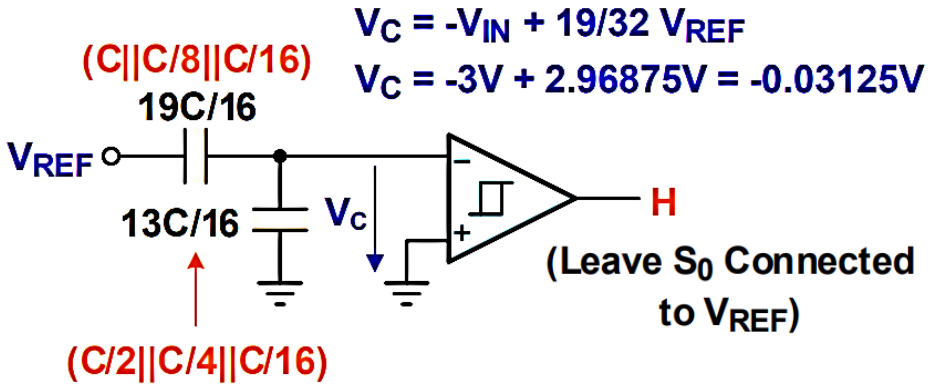
1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode (bit 1)*

Signal Conversion in ADC - Example

- With the same 5-bit SAR-ADC
  - Assuming that V_{IN} = 3V and V_{ref} = 5V



≙

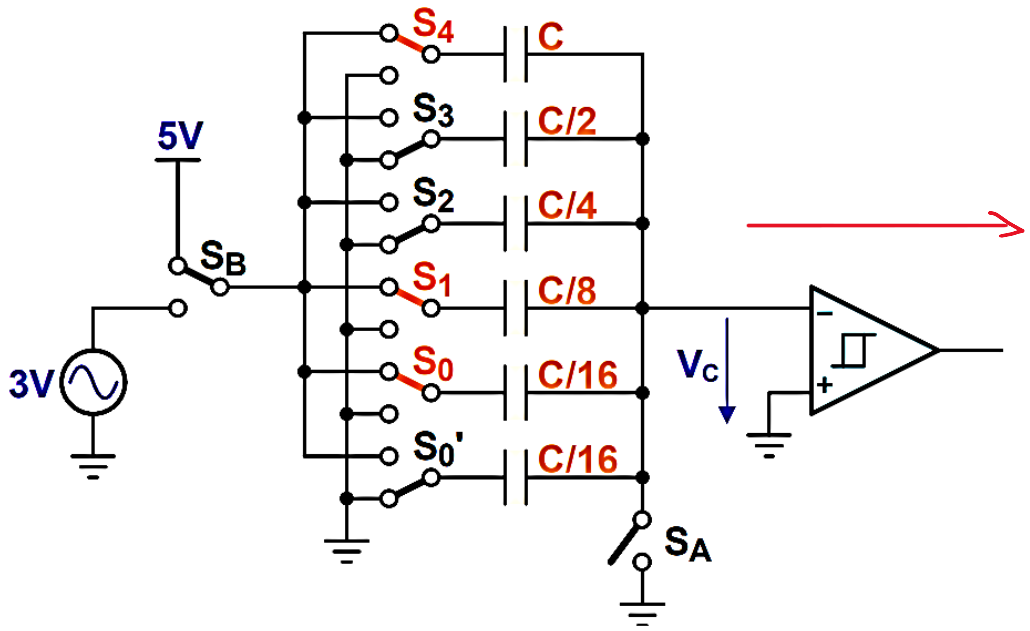


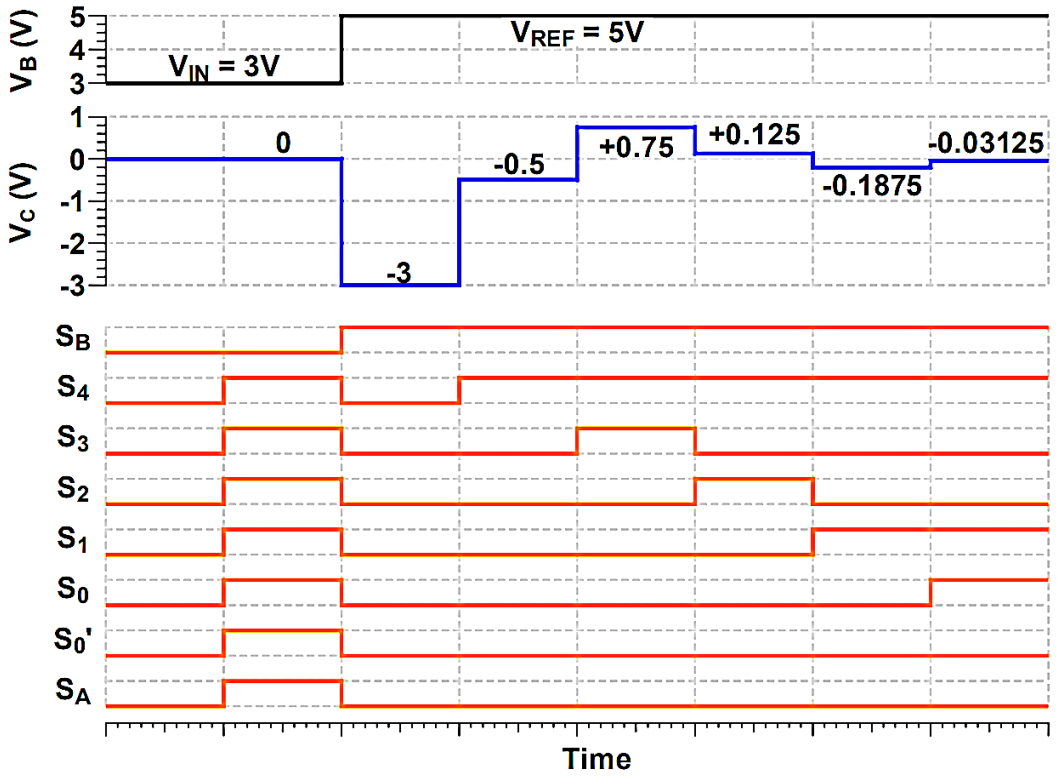
The conversion takes place in three stages

1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode (bit 0)*

Signal Conversion in ADC - Example

- With the same 5-bit SAR-ADC
  - Assuming that V_{IN} = 3V and V_{ref} = 5V



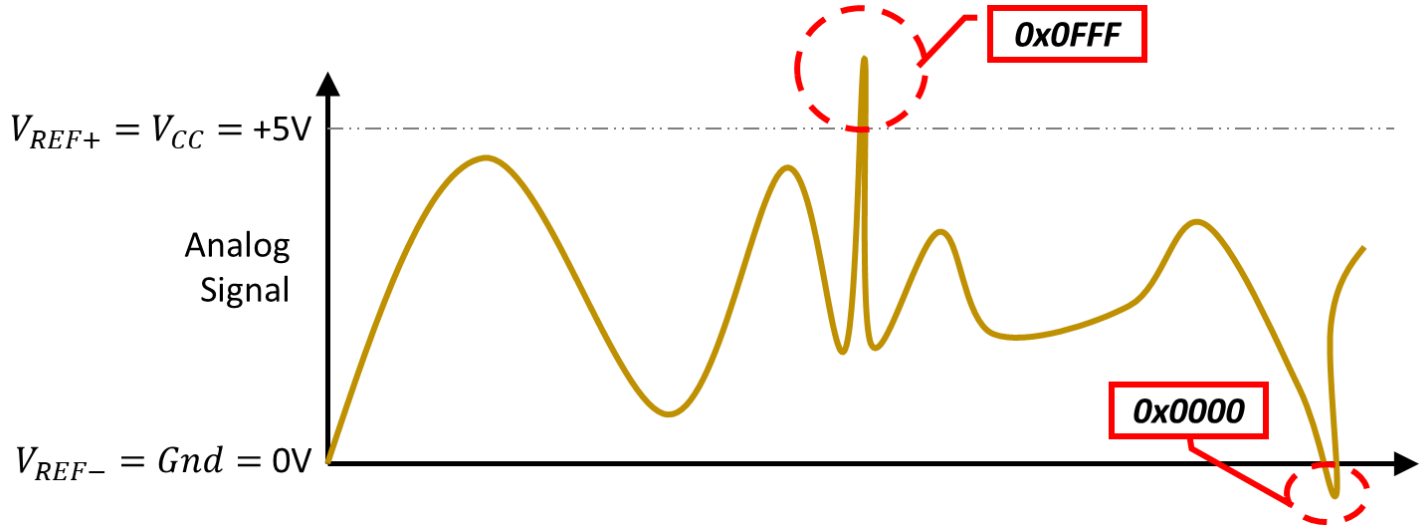


The conversion takes place in three stages

1. ***Sample*** *mode*
2. ***Hold*** *mode*
3. ***Redistribution*** *mode*

ADC – Reference Voltage

- Reference Voltage (limit)
  - Reference voltages define the voltage range where ADC performs the conversion.
  - When the voltage exceeds this range → max/min will be considered.



Similarly,

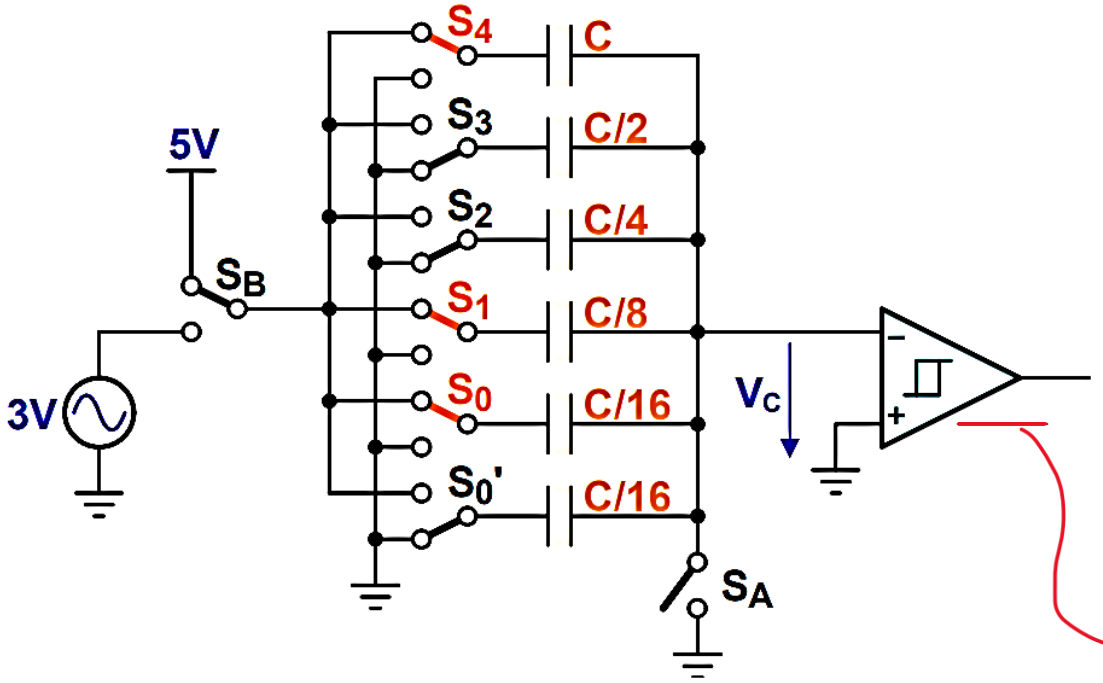
If $V_{IN} \geq V_{REF+}$ *then ADC output = max. value i.e. 0x0FFF*

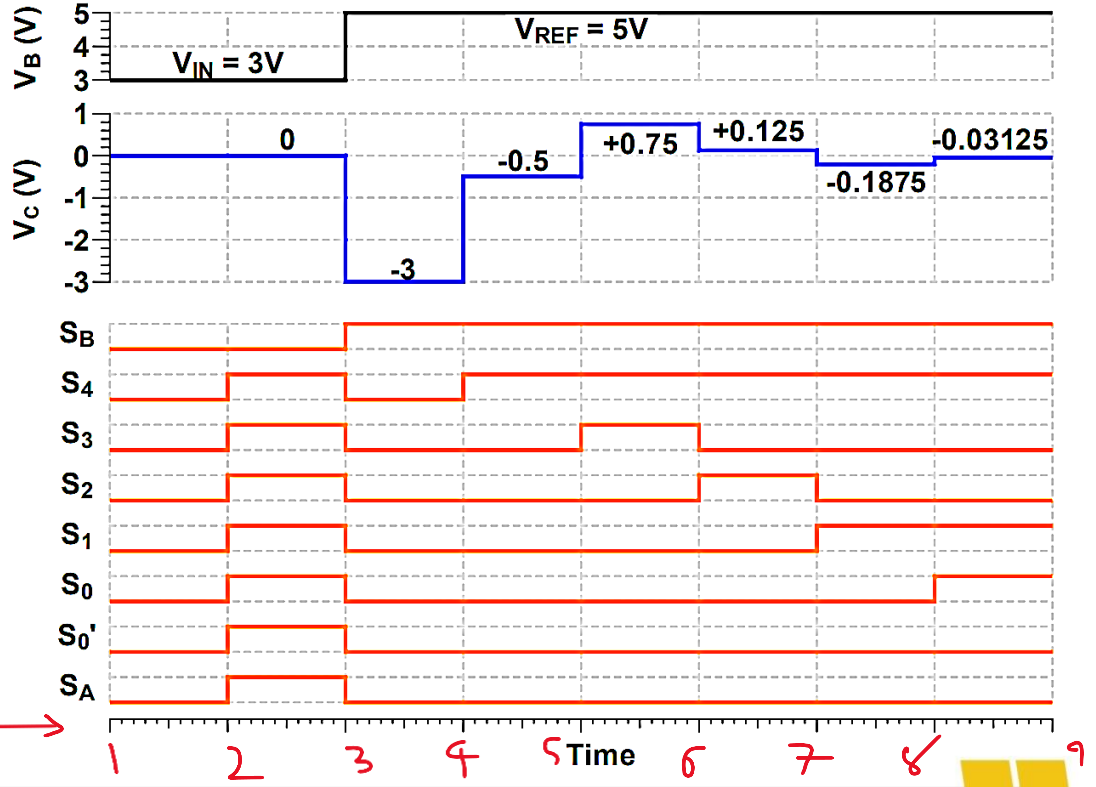
If $V_{IN} \leq V_{REF-}$ *then ADC output = min. value i.e. 0x0000*

Signal Conversion in ADC

- Sampling time in SAR
  - The RC circuit in the SAR takes time to charge and discharge. The total amount of time taken is

$$\text{Sampling time } t_{sample} \geq (R_I + R_s).(C_I + C_S).\ln(2^{n+2})$$





Signal Conversion in ADC

- **Sampling time in SAR**
  - **The RC circuit in the SAR takes time to charge and discharge. The total amount of time taken is**

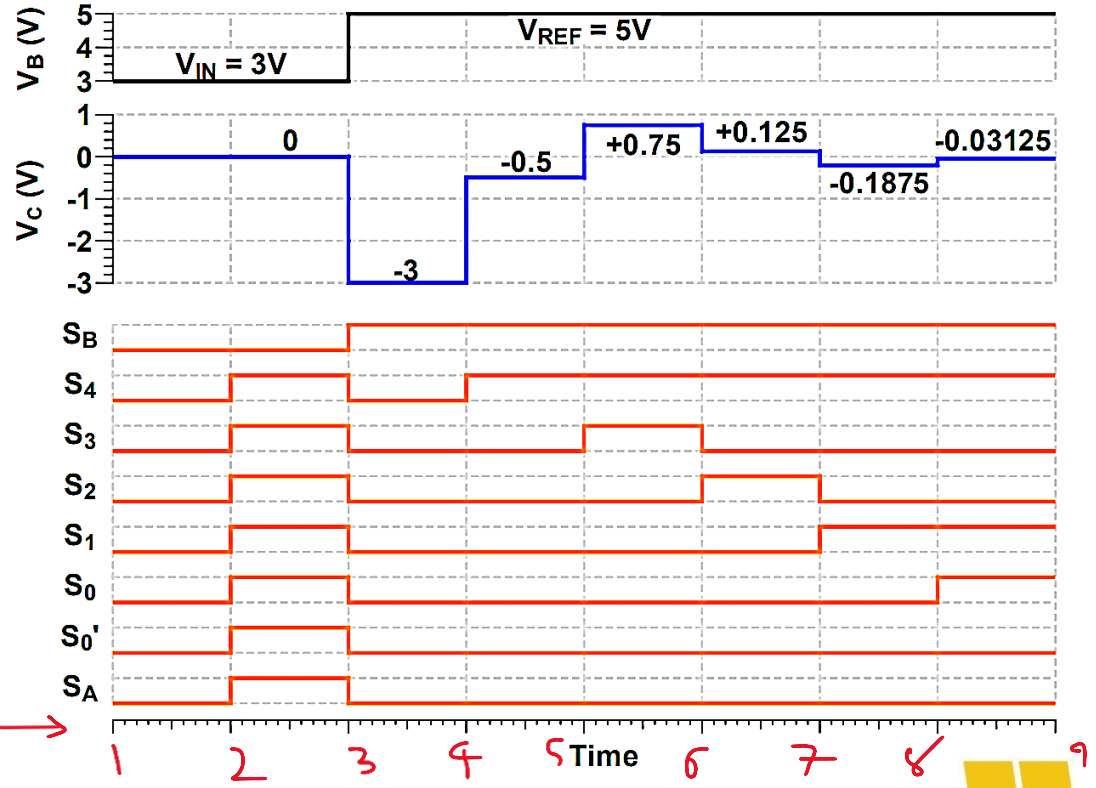
$$\text{Sampling time } t_{sample} \geq (R_I + R_s).(C_I + C_S).\ln(2^{n+2})$$

where,

R_s is the external source resistance
R_I is the internal input resistance
C_s is the external parasitic capacitance
C_I is the input capacitance
n: # of bits

The sampling time is typically in the range of µs.

Refer datasheet for device specific values.
(Table 5-25 in device datasheet)



Signal Conversion in ADC

- **Sampling time in SAR (SHT)**
  - **The RC circuit in the SAR takes time to charge and discharge. The total amount of time taken is**

$$\text{Sampling time } t_{sample} \geq (R_I + R_s).(C_I + C_S).\ln(2^{n+2})$$

where,

R_s is the external source resistance
R_I is the internal input resistance
C_s is the external parasitic capacitance
C_I is the input capacitance
n: # of bits

The sampling time is typically in the range of µs.

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
$f_{ADC12CLK}$	Frequency for specified performance	For specified performance of ADC12 linearity parameters with ADC12PWRMD = 0. If ADC12PWRMD = 1, the maximum is 1/4 of the value shown here.		0.45		5.4	MHz
$f_{ADC12CLK}$	Frequency for reduced	Linearity parameters have reduced performance			32.768		kHz
t_{Sample}	Sampling time	R_S = 400 Ω, R_I = 4 kΩ, C_I = 15 pF, C_{pext}= 8 pF (4)	(ADC12SHP = 0) with buffered reference (ADC12VRSEL = 0x1, 0x3, 0x5, 0x7, 0x9, 0xB, 0xD, 0xF)	1			µs
			Extended sample mode (ADC12SHP = 0) with unbuffered reference (ADC12VRSEL= 0x0, 0x2, 0x4, 0x6, 0xC, 0xE)	See (5)			µs

ADC in MSP430

- ADC12_B module performs the ADC operation
  - Has a single 12-bit ADC core ⟶ *supports 8-bit, 10-bit, and 12-bit resolution modes*
  - 32 different analog signals can be sampled at a time
  - Multiple reference voltages (V_{ref})
    - 1.2 V, 2.0 V, 5.0 V

- Configuration requirements
  - Clock source
  - Sampling time
  - Reference voltages
  - ADC channels

ADC in MSP430

- ADC12_B module performs the ADC operation
  - Has a single 12-bit ADC core
  - 32 different analog signals can be sampled at a time
  - Multiple reference voltages (V_{ref})
    - 1.2 V, 2.0 V, 5.0 V

- Configuration requirements
  - Clock source
  - Sampling time
  - Reference voltages
  - ADC channels

- *ACLK*
- *SMCLK*
- *MCLK*
- ***MODCLK***

ADC in MSP430

- ADC12_B module performs the ADC operation
  - Has a single 12-bit ADC core
  - 32 different analog signals can be sampled at a time
  - Multiple reference voltages (V_{ref})
    - 1.2 V, 2.0 V, 5.0 V

- Configuration requirements
  - Clock source
  - Sampling time
  - Reference voltages
  - ADC channels

- *ACLK*
- *SMCLK*
- *MCLK*
- ***MODCLK***

Modulator oscillator (MODOSC) generates MODCLK.
The frequency varies between 3.7 MHz and 6.3 MHz with a typical value of 4.8 MHz.

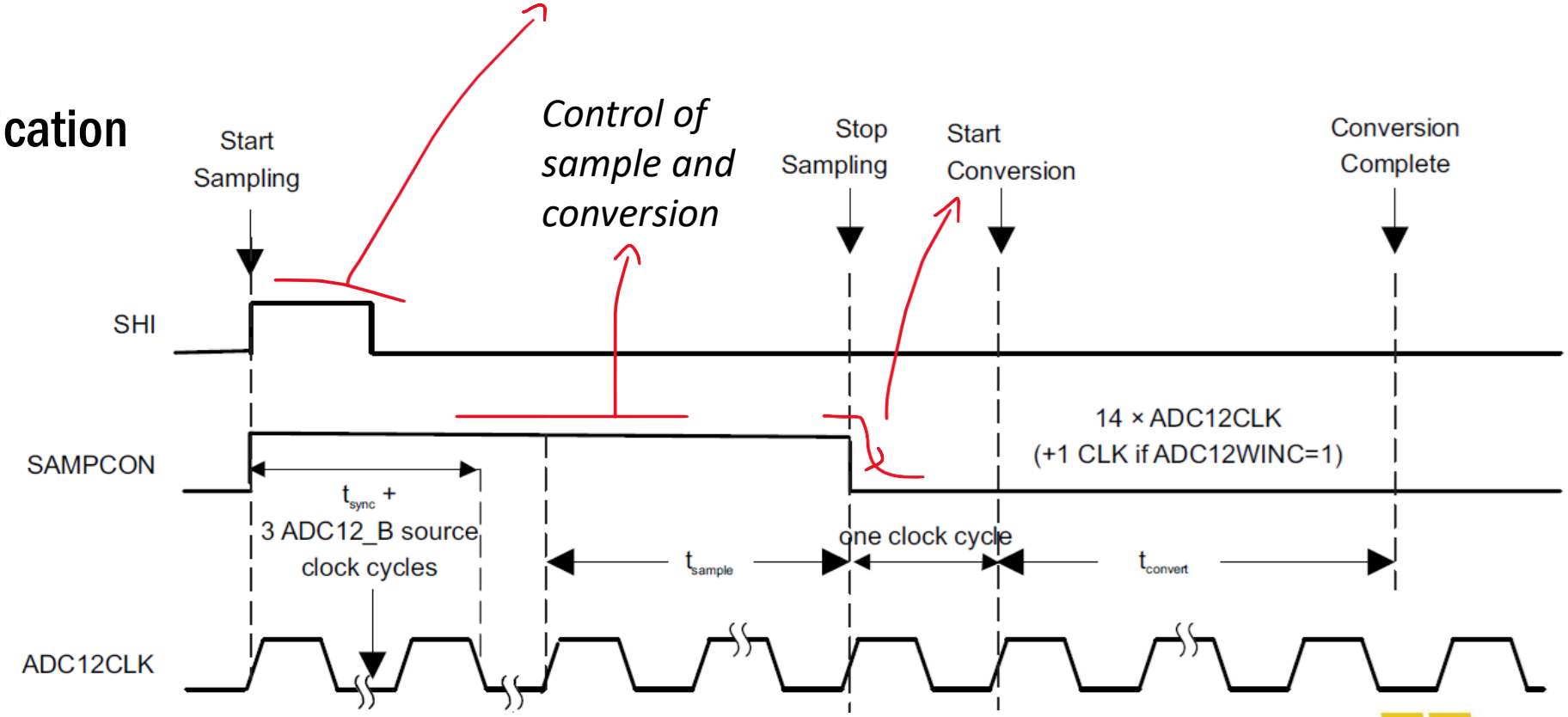
The frequency can be further divided by 1,2,3,4,5,6,7 or 8.

MODCLK is a cheap clock made of RC circuit.
Since timing precision in not very important, we will be using the MODOSC

ADC in MSP430

- ADC timing (required time)
  - Includes sampling time and conversion time
- Sampling time
  - Depending on the application (size of data)
- Conversion time
  - 10 cycles (for 8-bit)
  - 12 cycles (for 10-bit)
  - 14 cycles (for 12-bit)

This rising edge of signal shows that the ADC is started!



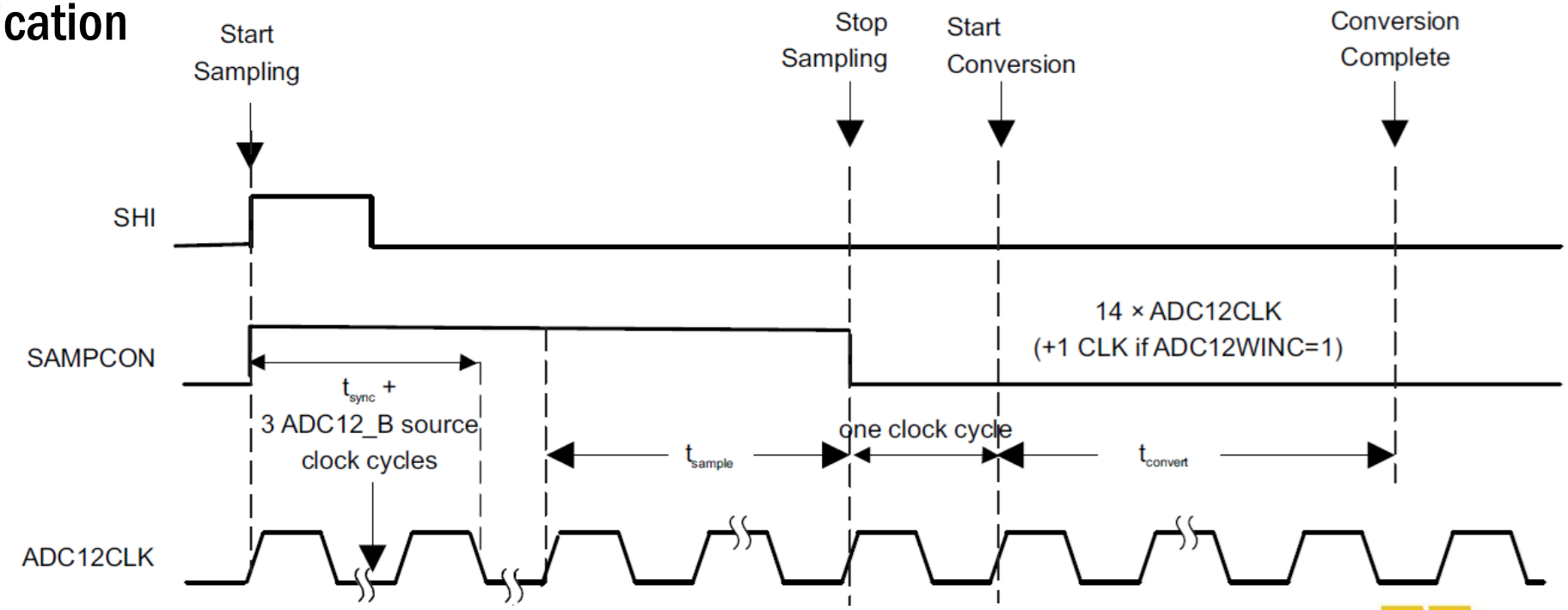
ADC in MSP430

- ADC timing (required time)
  - Includes sampling time and conversion time
- Sampling time
  - Depending on the application (size of data)
- Conversion time
  - 14 clock cycles

depends on the sensor

$$\text{Sampling time } t_{sample} \geq (R_I + R_s).(C_I + C_S).\ln(2^{n+2})$$

The sampling time is typically in the range of µs.



ADC in MSP430

- ADC timing (required time)
  - Includes sampling time and conversion time
- Sampling time
  - Depending on the application (size of data)
- Conversion time
  - 14 clock cycles

depends on the sensor

Sampling time $t_{sample} \geq (R_I + R_s).(C_I + C_S).\ln(2^{n+2})$

The sampling time is typically in the range of µs.

Minimum values for MSP430FR6989

R_s is the external source resistance
R_I is the internal input resistance
C_s is the external parasitic capacitance
C_I is the input capacitance
n: # of bits

$R_I = 4\ K\Omega;\ C_I = 15\ pF$
$R_S = 400\ \Omega;\ C_S = 8\ pF$

→ *minimum* $t_{sample} \geq 1\ \mu s$

For the inbuilt temperature sensor in MSP430, $\boldsymbol{R_S = 250\ K\Omega}$

ADC in MSP430

- Sampling time → in clock cycles
  - The ADC module requires the sample time to be provided in number of clock cycles.
  - Different applications need different amount of sampling time
    - e.g., the temperature sensor needs a minimum sample time of 30μs

- Clock source → MODOSC → frequency varies between 3.7 MHz and 6.3 MHz

Minimum sampling time ⟷ *Maximum possible frequency*

- Example: to achieve $t_{sampling}$=30µs, (at 6.3 MHz)

1 s = 6.3 x 10^6 cycles → 30 µs = 30 x 6.3 cycles = 189 cycles

- Example: to achieve $t_{sampling}$=30µs, (at 6.3/8 MHz) – with /8 divider

1 s = 6.3/8 x 10^6 cycles → 30 µs = 30 x 6.3/8 cycles ≈ 24 cycles

ADC in MSP430

- Sampling time → in clock cycles

The number of clock cycles can be configured to
4 cycles,
8 cycles,
16 cycles,
***32 cycles**,*
64 cycles,
128 cycles,
***192 cycles**,*
256 cycles,
384 cycles
or 512 cycles.

- Example: to achieve $t_{sampling}$=30µs, (at 6.3 MHz)

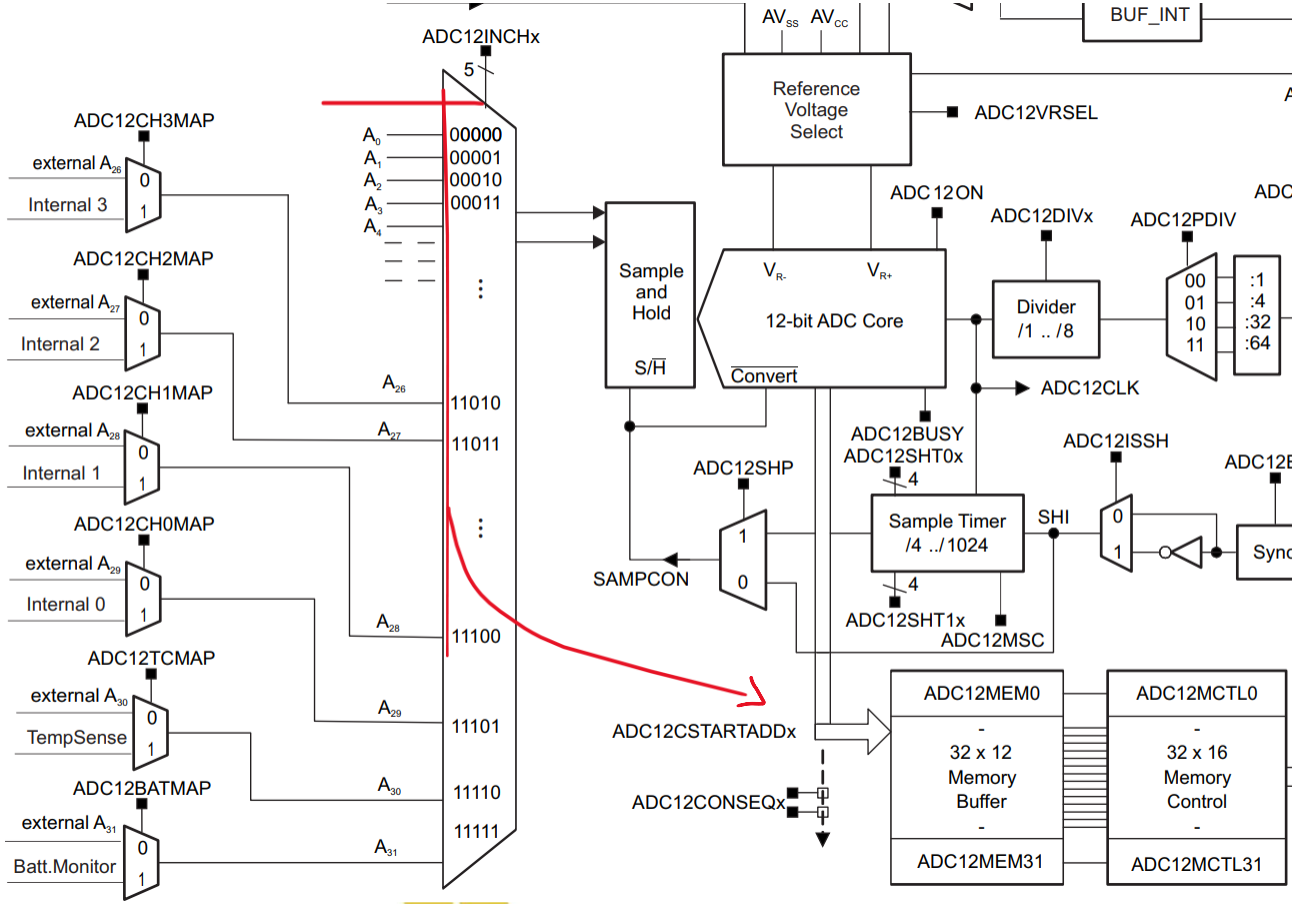
*1 s = 6.3 x 10^6 cycles → 30 µs = 30 x 6.3 cycles = 189 cycles <= **192 clock cycles***

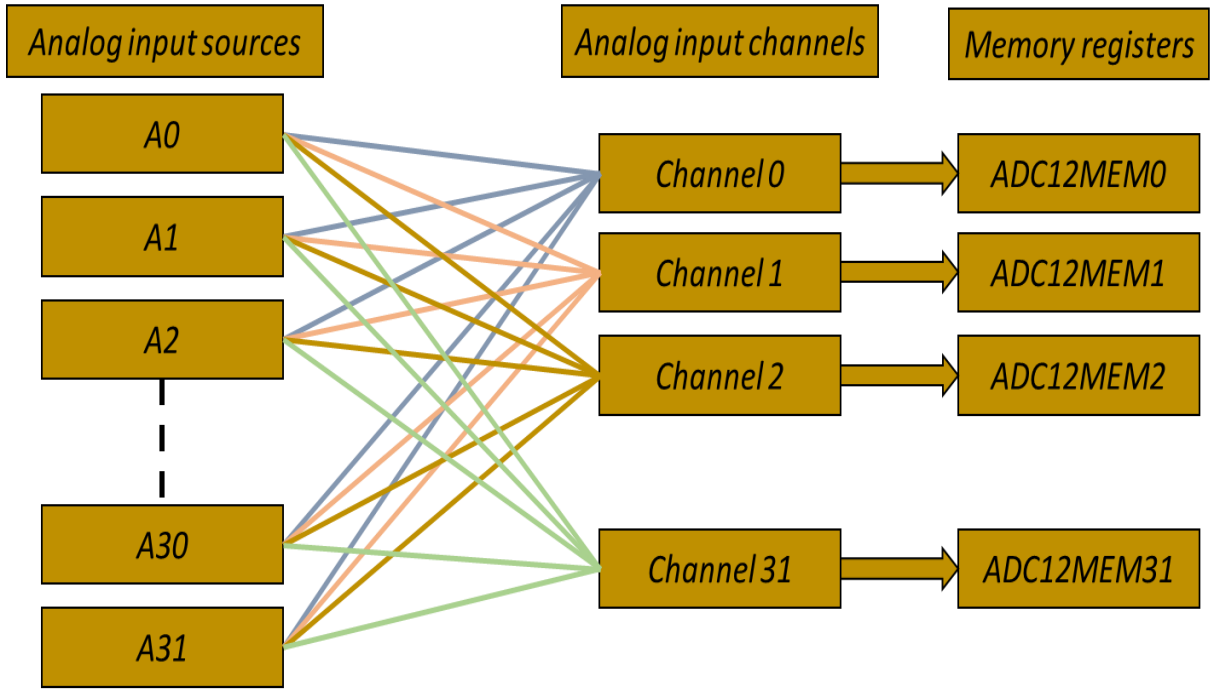
- Example: to achieve $t_{sampling}$=30µs, (at 6.3/8 MHz) – with /8 divider

*1 s = 6.3/8 x 10^6 cycles → 30 µs = 30 x 6.3/8 cycles ≈ 24 cycles <= **32 clock cycles***

ADC in MSP430 – Channels

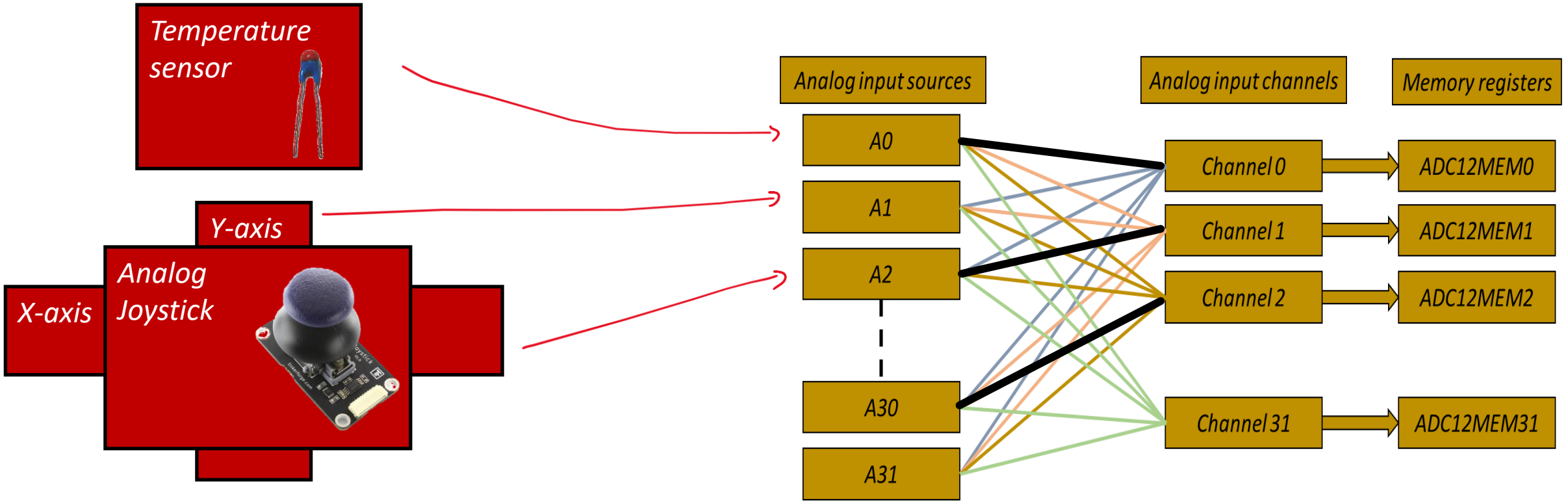
- 32 different channels in ADC12_B
  - Each channel with an input and a corresponded fixed memory register to store the output!





ADC in MSP430 – Channels

- 32 different channels in ADC12_B
  - Each channel with an input and a corresponded fixed memory register to store the output!



ADC in MSP430 - Registers

- **ADC12 Control Register 0**

Two set of channels can have two different sampling periods

ADC12CTL0

ADC12SHT1x	4 bits	Configures the number of clock cycles for channels 8 to 23
ADC12SHT0x	4 bits	Configures the number of clock cycles for channels 0 to 7 and 24 to 31.
ADC12ON	1 bit	Turns on the ADC_12 module
ADC12MSC	1 bit	Enables converting multiple channels at a time
ADC12ENC	1 bit	Enables conversion
ADC12SC	1 bit	Setting it, starts the ADC conversion. Reset by hardware automatically, after conversion.

All ADC12 module configuration must be done only when ADC12ENC is 0.

ADC in MSP430 - Registers

- **ADC12 Control Register 0**

ADC12CTL0

15	14	13	12	11	10	9	8
ADC12SHT1x				ADC12SHT0x			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12MSC	Reserved		ADC12ON	Reserved		ADC12ENC	ADC12SC
rw-(0)	r-0	r-0	rw-(0)	r-0	r-0	rw-(0)	rw-(0)

Can be modified only when ADC12ENC = 0.

channels 8-23

0000b = 4 ADC12CLK cycles
0001b = 8 ADC12CLK cycles
0010b = 16 ADC12CLK cycles
0011b = 32 ADC12CLK cycles
0100b = 64 ADC12CLK cycles
0101b = 96 ADC12CLK cycles
0110b = 128 ADC12CLK cycles
0111b = 192 ADC12CLK cycles
1000b = 256 ADC12CLK cycles
1001b = 384 ADC12CLK cycles
1010b = 512 ADC12CLK cycles
1011b = Reserved
1100b = Reserved
1101b = Reserved
1110b = Reserved
1111b = Reserved

channels 0-7 & 24-31

0000b = 4 ADC12CLK cycles
0001b = 8 ADC12CLK cycles
0010b = 16 ADC12CLK cycles
0011b = 32 ADC12CLK cycles
0100b = 64 ADC12CLK cycles
0101b = 96 ADC12CLK cycles
0110b = 128 ADC12CLK cycles
0111b = 192 ADC12CLK cycles
1000b = 256 ADC12CLK cycles
1001b = 384 ADC12CLK cycles
1010b = 512 ADC12CLK cycles
1011b = Reserved
1100b = Reserved
1101b = Reserved
1110b = Reserved
1111b = Reserved

Field	Type	Reset	Description
ADC12MSC	RW	0	ADC12_B multiple sample and conversion. Valid only for sequence or repeated modes. Can be modified only when ADC12ENC = 0. 0b = The sampling timer requires a rising edge of the SHI signal to trigger each sample-and-convert. 1b = The incidence of the first rising edge of the SHI signal triggers the sampling timer, but further sample-and-conversions are performed automatically as soon as the prior conversion is completed.
Reserved	R	0	Reserved. Always reads as 0.
ADC12ON	RW	0	ADC12_B on. Can be modified only when ADC12ENC = 0. 0b = ADC12_B off 1b = ADC12_B on
Reserved	R	0	Reserved. Always reads as 0.
ADC12ENC	RW	0	ADC12_B enable conversion. 0b = ADC12_B disabled 1b = ADC12_B enabled
ADC12SC	RW	0	ADC12_B start conversion. Software-controlled sample-and-conversion start. ADC12SC and ADC12ENC may be set together with one instruction. ADC12SC is reset automatically. 0b = No sample-and-conversion-start 1b = Start sample-and-conversion

ADC in MSP430 - Registers

- **ADC12 Control Registers 1 & 2**

ADC12CTL1

ADC12SSEL	*2 bits*	*Selects the clock source*
ADC12DIV	*3 bits*	*Clock divider*
ADC12BUSY	*1 bit*	*Indicates whether the ADC module is busy with conversions.*
ADC12SHS	*3 bits*	*Trigger for sample and hold operation*
ADC12SHP	*1 bit*	*Sampling signal source select*
ADC12CONSEQ	*2 bits*	*Conversion mode*

ADC12CTL2

ADC12RES	*2 bits*	*Selects the resolution of the ADC (8 bit, 10 bit or 12 bit)*
ADC12DF	*1 bit*	*Data format*
ADC12CSTARTADD	*5 bits*	*Starting channel for conversion*

ADC in MSP430 - Registers

- **ADC12 Control Registers 1**

ADC12CTL1

15	14	13	12	11	10	9	8
Reserved	ADC12PDIV		ADC12SHSx			ADC12SHP	ADC12ISSH
r-0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12DIVx			ADC12SSELx		ADC12CONSEQx		ADC12BUSY
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)

Can be modified only when ADC12ENC = 0.

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved. Always reads as 0.
14-13	ADC12PDIV	RW	0h	ADC12_B predivider. This bit predivides the selected ADC12_B clock source. 00b = Predivide by 1 01b = Predivide by 4 10b = Predivide by 32 11b = Predivide by 64
8	ADC12ISSH	RW	0h	ADC12_B invert signal sample-and-hold. 0b = The sample-input signal is not inverted. 1b = The sample-input signal is inverted.

Bit	Field	Type	Reset	Description
7-5	ADC12DIVx	RW	0h	ADC12_B clock divider 000b = /1 001b = /2 010b = /3 011b = /4 100b = /5 101b = /6 110b = /7 111b = /8
4-3	ADC12SSELx	RW	0h	ADC12_B clock source select 00b = ADC12OSC (MODOSC) 01b = ACLK 10b = MCLK 11b = SMCLK

ADC in MSP430 - Registers

- **ADC12 Control Registers 2**

ADC12CTL2

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved		ADC12RES		ADC12DF	Reserved		ADC12PWRMD
r0	r0	rw-(1)	rw-(0)	rw-(0)	r0	r0	rw-(0)

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved. Always reads as 0.
5-4	ADC12RES	RW	2h	ADC12_B resolution. This bit defines the conversion result resolution. This bit should only be modified when ADC12ENC=0. 00b = 8 bit (10 clock cycle conversion time) 01b = 10 bit (12 clock cycle conversion time) 10b = 12 bit (14 clock cycle conversion time) 11b = Reserved
3	ADC12DF	RW	0h	ADC12_B data read-back format. Data is always stored in the binary unsigned format. 0b = Binary unsigned. 1b = Signed binary (2s complement), left aligned.
0	ADC12PWRMD	RW	0h	Enables ADC low-power mode for ADC12CLK with 1/4 the specified maximum for ADC12PWRMD = 0. This bit should only be modified when ADC12ENC = 0. 0b = Regular power mode where sample rate is not restricted 1b = Low power mode enable, ADC12CLK can not be greater than 1/4 the device-specific data sheet specified maximum for ADC12PWRMD = 0

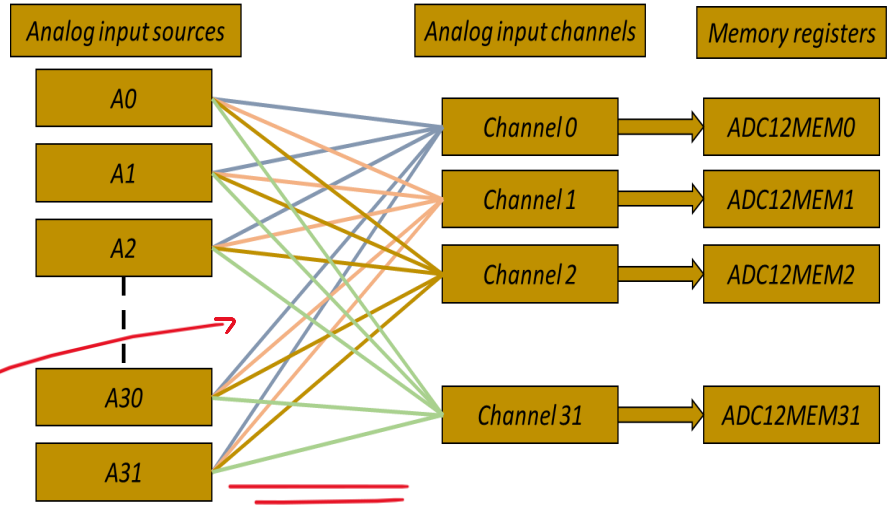
ADC in MSP430 - Registers

- **ADC12 Conversion memory control register – channel x**

ADC12MCTLx

ADC12VRSEL	4 bits	Selects the reference voltages
ADC12INCH	5 bits	Selects one source from 32 available analog voltage input sources

Bit	Field	Type	Reset	Description
11-8	ADC12VRSEL	RW	0h	Selects combinations of VR+ and VR- sources as well as the buffer selection. Note: there is only one buffer so it can be used for either VR+ or VR-, but not both. Can be modified only when ADC12ENC = 0. 0000b = VR+ = AVCC, VR- = AVSS 0001b = VR+ = VREF buffered, VR- = AVSS 0010b = VR+ = VeREF-, VR- = AVSS 0011b = VR+ = VeREF+ buffered, VR- = AVSS
4-0	ADC12INCHx	RW	0h	Input channel select. If even channels are set as differential, then odd channel configuration is ignored. Can be modified only when ADC12ENC = 0. 00000b = If ADC12DIF = 0: A0; If ADC12DIF = 1: Ain+ = A0, Ain- = A1 00001b = If ADC12DIF = 0: A1; If ADC12DIF = 1: Ain+ = A0, Ain- = A1 00010b = If ADC12DIF = 0: A2; If ADC12DIF = 1: Ain+ = A2, Ain- = A3 00011b = If ADC12DIF = 0: A3; If ADC12DIF = 1: Ain+ = A2, Ain- = A3 00100b = If ADC12DIF = 0: A4; If ADC12DIF = 1: Ain+ = A4, Ain- = A5



ADC in MSP430 - Registers

- Interrupt flag (Optional)
  - ADC12IERx – ADC12 Interrupt enable registers.
  - ADC12IFGRx – ADC12 Interrupt flag registers.

ADC12IER0

15	14	13	12	11	10	9	8
ADC12IE15	ADC12IE14	ADC12IE13	ADC12IE12	ADC12IE11	ADC12IE10	ADC12IE9	ADC12IE8
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12IE7	ADC12IE6	ADC12IE5	ADC12IE4	ADC12IE3	ADC12IE2	ADC12IE1	ADC12IE0
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

0b: interrupt disabled
1b: interrupt enabled

ADC12IER1

15	14	13	12	11	10	9	8
ADC12IE31	ADC12IE30	ADC12IE29	ADC12IE28	ADC12IE27	ADC12IE26	ADC12IE25	ADC12IE24
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12IE23	ADC12IE22	ADC12IE21	ADC12IE20	ADC12IE19	ADC12IE18	ADC12IE17	ADC12IE16
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

0b: interrupt disabled
1b: interrupt enabled

ADC in MSP430 - Registers

- **Interrupt flag (Optional)**
  - **ADC12IERx – ADC12 Interrupt enable registers.**
  - **ADC12IFGRx – ADC12 Interrupt flag registers.**

This bit is set when ADC12MEMx is loaded with a conversion result.
The ADC12IFGx bit is reset if ADC12MEM31 is accessed, or it can be reset with software

ADC12IFGR0

15	14	13	12	11	10	9	8
ADC12IFG15	ADC12IFG14	ADC12IFG13	ADC12IFG12	ADC12IFG11	ADC12IFG10	ADC12IFG9	ADC12IFG8
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12IFG7	ADC12IFG6	ADC12IFG5	ADC12IFG4	ADC12IFG3	ADC12IFG2	ADC12IFG1	ADC12IFG0
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

0b: NO interrupt pending
1b: interrupt pending

ADC12IFGR1

15	14	13	12	11	10	9	8
ADC12IFG31	ADC12IFG30	ADC12IFG29	ADC12IFG28	ADC12IFG27	ADC12IFG26	ADC12IFG25	ADC12IFG24
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12IFG23	ADC12IFG22	ADC12IFG21	ADC12IFG20	ADC12IFG19	ADC12IFG18	ADC12IFG17	ADC12IFG16
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

0b: NO interrupt pending
1b: interrupt pending

ADC in MSP430 - Programming

- **Determining the register values for SHT and conversion time**

$$\text{Sampling time } t_{sample} \geq (R_I + R_s).(C_I + C_S).\ln(2^{n+2})$$

The number of clock cycles can be configured to

R_I = 4 KΩ
R_E = 10 KΩ
C_I = 15 pF
C_E = 1 pF
n (in ADC12_B) = 12

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
$f_{ADC12CLK}$	Frequency for specified performance	For specified performance of ADC12 linearity parameters with ADC12PWRMD = 0. If ADC12PWRMD = 1, the maximum is 1/4 of the value shown here.		0.45		5.4	MHz
$f_{ADC12CLK}$	Frequency for reduced	Linearity parameters have reduced performance			32.768		kHz
t_{Sample}	Sampling time	R_S = 400 Ω, R_I = 4 kΩ, C_I = 15 pF, C_{pext}= 8 pF (4)	(ADC12SHP = 0) with buffered reference (ADC12VRSEL = 0x1, 0x3, 0x5, 0x7, 0x9, 0xB, 0xD, 0xF)	1			µs
			Extended sample mode (ADC12SHP = 0) with unbuffered reference (ADC12VRSEL= 0x0, 0x2, 0x4, 0x6, 0xC, 0xE)	See (5)			µs

4 cycles,
8 cycles,
16 cycles,
***32 cycles**,*
64 cycles,
128 cycles,
192 cycles,
256 cycles,
384 cycles
or 512 cycles.

$(4KΩ + 10KΩ).(15pF+ 1pF).\ln(2^{12+2})$
≈ 2.2 µs

*Using **MODCLK** at the frequency*
- *varies between 3.7 MHz and 6.3 MHz*
- *with a typical value of 4.8 MHz*

*1 s = 6.3 x 10^6 cycles → 3 µs = 3 x 6.3 cycles = 18.9 cycles <= **32 cycles***

Divider is required in case it is more than 512!

ADC in MSP430 - Programming

- Determining the register values for SHT and conversion time
  - SHT in clock cycles
    - Based on the clock source, divider, and RC model of the ADC

R_I = 4 KΩ
R_E = 10 KΩ
C_I = 15 pF
C_E = 1 pF
n (in ADC12_B) = 12

→ $(4KΩ + 10KΩ).(15pF + 1pF).ln(2^{12+2})$
≈ 2.2 µs

1 s = 6.3 x 10^6 cycles → 3 µs = 3 x 6.3 cycles
= 18.9 cycles <= **32 cycles**

channels 8-23	*channels 0-7 & 24-31*
0000b = 4 ADC12CLK cycles	0000b = 4 ADC12CLK cycles
0001b = 8 ADC12CLK cycles	0001b = 8 ADC12CLK cycles
0010b = 16 ADC12CLK cycles	0010b = 16 ADC12CLK cycles
0011b = 32 ADC12CLK cycles	0011b = 32 ADC12CLK cycles
0100b = 64 ADC12CLK cycles	0100b = 64 ADC12CLK cycles
0101b = 96 ADC12CLK cycles	0101b = 96 ADC12CLK cycles
0110b = 128 ADC12CLK cycles	0110b = 128 ADC12CLK cycles
0111b = 192 ADC12CLK cycles	0111b = 192 ADC12CLK cycles
1000b = 256 ADC12CLK cycles	1000b = 256 ADC12CLK cycles
1001b = 384 ADC12CLK cycles	1001b = 384 ADC12CLK cycles
1010b = 512 ADC12CLK cycles	1010b = 512 ADC12CLK cycles
1011b = Reserved	1011b = Reserved
1100b = Reserved	1100b = Reserved
1101b = Reserved	1101b = Reserved
1110b = Reserved	1110b = Reserved
1111b = Reserved	1111b = Reserved

15	14	13	12	11	10	9	8
ADC12SHT1x *channels 8-23*				ADC12SHT0x *channels 0-7 & 24-31*			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12MSC	Reserved		ADC12ON	Reserved		ADC12ENC	ADC12SC
rw-(0)	r-0	r-0	rw-(0)	r-0	r-0	rw-(0)	rw-(0)
	Can be modified only when ADC12ENC = 0.						

ADC in MSP430 - Programming

- **Implementation – for joystick X-axis and Y-axis**

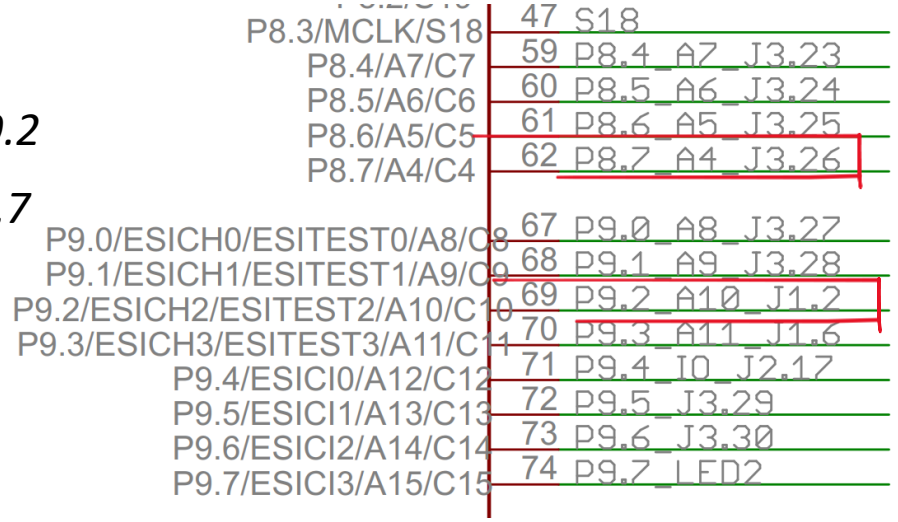
From the Boosterpack for joystick pinout

BoosterPack Plug-in Module Header Connection	Pin Function
J1.2	Horizontal X-axis
J1.5	Select button
J3.26	Vertical Y-axis

HorAxis → connected to p9.2

VerAxis → connected to p8.7

From the Launchpad (Full schematic)



- **Default is GPIO**
  - **configure this pin to the A10/A4 function**

P9.2/ESICH2/ESITEST2/A10/C10

x-axis

2	P9.2 (I/O) ***default***	I: 0; O: 1	0	0
	N/A	0	0	1
	Internally tied to DVSS	1		
	ESITEST2 (2)	X	1	0
	ESICH2/A10/C10 (2)(3)(4)	X	1	1

P8.7/A4/C4

y-axis

7	P8.7 (I/O) ***default***	I: 0; O: 1	0	0
	N/A	0	0	1
	Internally tied to DVSS	1		
	N/A	0	1	0
	Internally tied to DVSS	1		
	A4/C4 (2) (3)	X	1	1

```
// X-axis: A10/P9.2, for A10 (P9DIR=x, P9SEL1=1, P9SEL0=1)
P9SEL1 |= BIT2;          // bit 2 of P9
P9SEL0 |= BIT2;          // bit 2 of P9

// Y-axis: A4/P8.7, for A4 (P8DIR=x, P8SEL1=1, P8SEL0=1)
P9SEL1 |= BIT7;          // bit 7 of P8
P9SEL0 |= BIT7;          // bit 7 of P8
```

ADC in MSP430 - Programming

- **Implementation – for joystick X-axis only**

Setting the pins for joystick (x-axis through P9.2)

Turning on the ADC module

Turning off the (Enable Conversion) bit while modifying the configuration

select the number of cycles based on the sampling time (here it is 10 → 512)

select the sample-and-hold source and determine the pulse-mode select, clock source, and clock divider

Determine the resolution (12-bit) and data format (unsigned binary)

```
void Initialize_ADC(void){
  // X-Axis of the Joystick is connected to pin P9.2
  // X-Axis: A10/P9.2, for A10 (P9DIR=x, P9SEL1=1, P9SEL0=1)
  P9SEL0 |= BIT2;
  P9SEL1 |= BIT2;

  // Turn on the ADC module
  ADC12CTL0 |= ADC12ON;

  // Turn OFF ENC
  ADC12CTL0 &= ~ADC12ENC;

  // Configure ADC12CTLx registers
  ADC12CTL0 |= ADC12SHT1_10;
  ADC12CTL1 = ADC12SHS_0 | ADC12SHP | ADC12DIV_7 | ADC12SSEL_0;
  ADC12CTL2 = ADC12RES_2;

  ADC12MCTL0 |= ADC12INCH_10 | ADC12VRSEL_0;

  // Turn ON ENC
  ADC12CTL0 |= ADC12ENC;

  return;
}
```

Determine the voltage range and channel selection

Turning on the EC bit after all configurations

ADC in MSP430 - Programming

- **Implementation – for joystick X-axis only**

```
void main(void){

  WDTCTL = WDTPW | WDTHOLD;
  PM5CTL0 &= ~LOCKLPM5;
  Initialize_ADC();
  Initialize_UART();

  while (1) {
    ADC12CTL0 |= ADC12SC;           // start the conversion

    // wait until busy flag is set
    while((ADC12CTL1 & ADC12BUSY) != 0);
    uart_write_uint16(ADC12MEM0)
  }
}
```

Write data to UART (from ADC)

```
void Initialize_ADC(void){
  // X-Axis of the Joystick is connected to pin P9.2
  // X-Axis: A10/P9.2, for A10 (P9DIR=x, P9SEL1=1, P9SEL0=1)
  P9SEL0 |= BIT2;
  P9SEL1 |= BIT2;

  // Turn on the ADC module
  ADC12CTL0 |= ADC12ON;

  // Turn OFF ENC
  ADC12CTL0 &= ~ADC12ENC;

  // Configure ADC12CTLx registers
  ADC12CTL0 |= ADC12SHT1_10;
  ADC12CTL1 = ADC12SHS_0 | ADC12SHP | ADC12DIV_7 | ADC12SSEL_0;
  ADC12CTL2 = ADC12RES_2;

  ADC12MCTL0 |= ADC12INCH_10 | ADC12VRSEL_0;

  // Turn ON ENC
  ADC12CTL0 |= ADC12ENC;

  return;
}
```

Turning on the EC bit after all configurations

Thank You!

Questions?

Email: kamali@ucf.edu

UCF HEC 435 **(407) 823 – 0764**

https://www.ece.ucf.edu/~kamali/

HAVEN Research Group

https://haven.ece.ucf.edu/

UNIVERSITY OF CENTRAL FLORIDA