Lecture 06

Lecture contents:

1-DAC
- AD7523
- Coding
- DAC0830/ DAC0832

2-ADC
- Digital ramp ADC
- SAR ADC
- Flash ADC
- General algorithm and interfacing
- ADC0808/ ADC 0809

"Slide_set_2 14:42"

مبئدياً المحاضرة طويلة معلش بس الدنيا لذيذة ومش هتحتاج تفتح السلايدز معاها عشان اغلبها موجود هنا

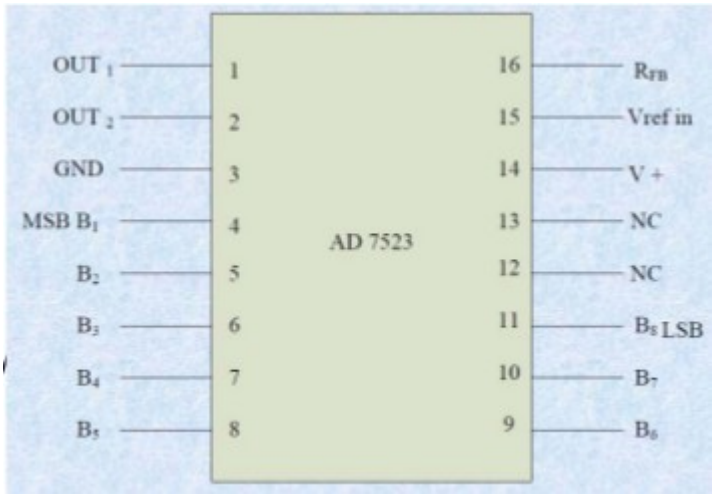
احنا طول الترم كنا بنتكلم عن ال interconnections اللي بتبقى جوة ال computer system و كنا بدأنا نتكلم عن ال interfacing اللي بيبقى بين ال analog world وال computer وبدأنا نتكلم عن ال DAC وقلنا ان فيه weighted sum وفيه R-2R ladder وقلنا برضه ان ال R-2R احسن وهو اللي الناس بتستخدمه

المحاضرة دي بقى هندخل اكتر في ال DAC ومنه هنروح لل ADC، فهنبدأ ب DAC اسمه *AD7523* :

دة بيعمل conversion ل 8 bits وفيه 16 input/output pins :

- **AD 7523 8-bit Multiplying DAC**
  - 16 pin DIP, multiplying DAC, containing R-2R ladder for D-A conversion along with single pole double thrown NMOS switches to connect the digital inputs to the ladder.

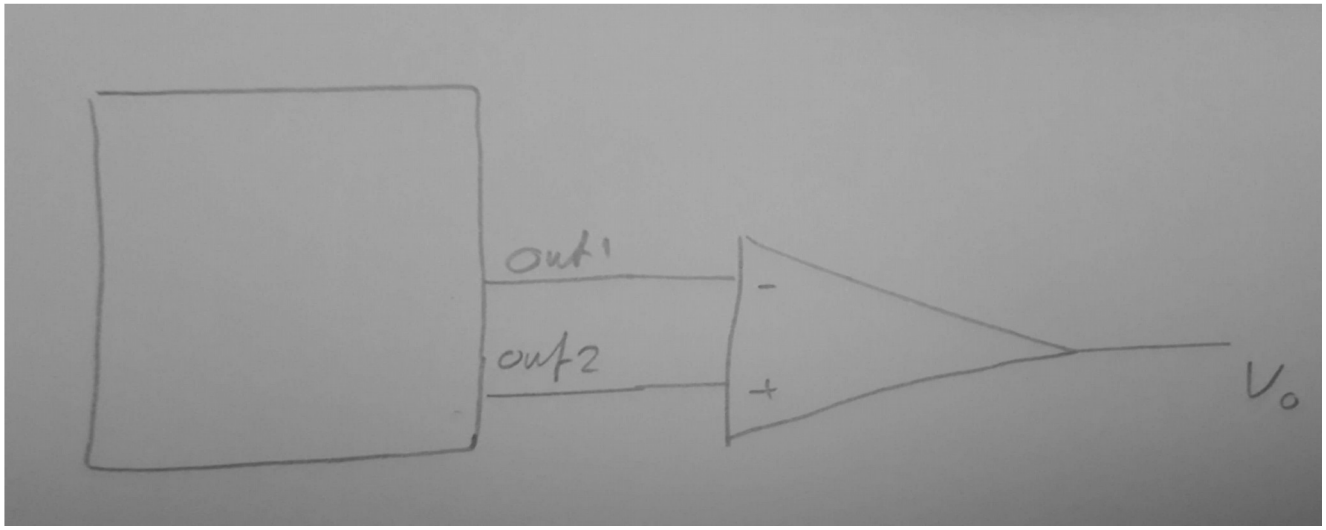
ودة شكل ال block بتاعه:



V^+ دي هي ال supply voltage وال range بتاعه بيبقى من 5+ ل 15+ فولت

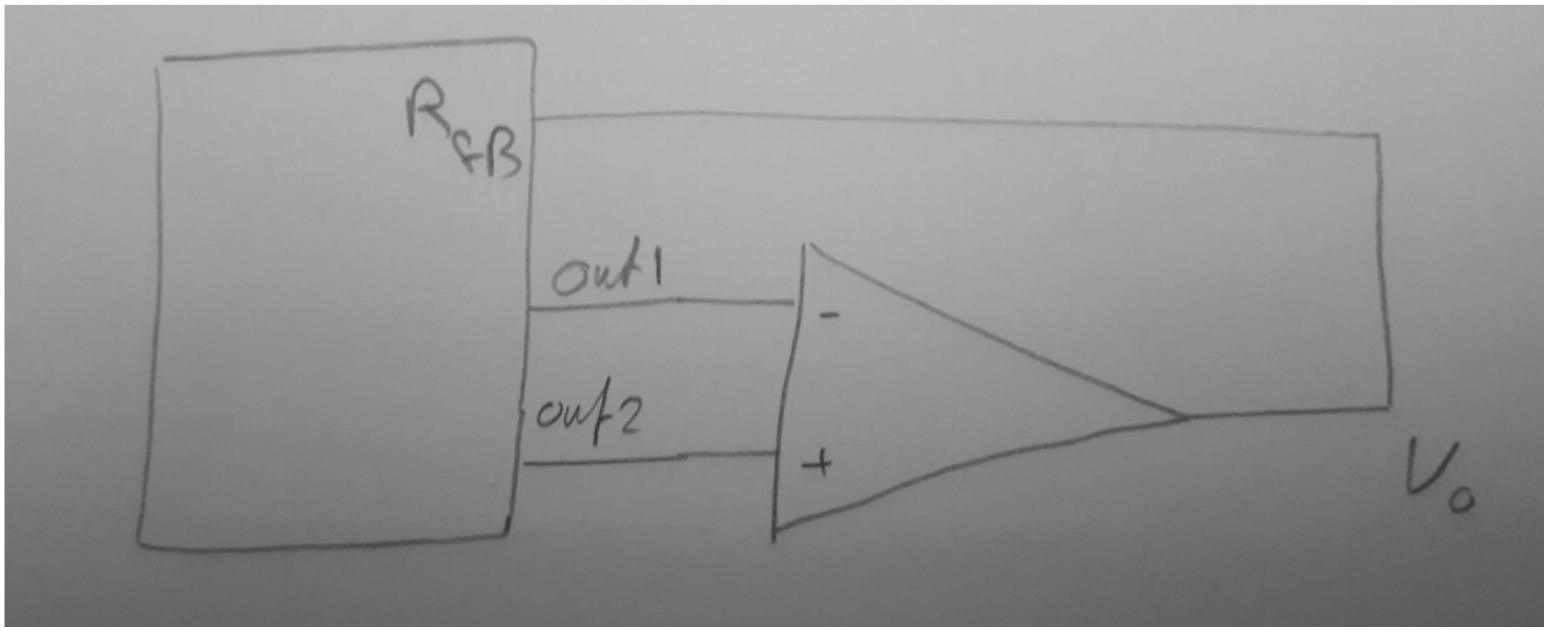
بس V^{ref} دي اللي بنحدد بيها ال maximum voltage اللي عايزين نطلعه من ال DAC بعد ال conversion وبالتالي بنحدد بيها ال step size كمان .. ودي ال range بتاعها من -10 ل 10 فولت
B^1 : B^8 دول هم ال input data وخلي بالك ان ال MSB هنا هي B^1 وال LSB هي B^8

كل اللي فات دة حلو مافيهوش مشكلة بس فيه حاجة غريبة هتبدأ تحصل هنا:
ال IC دي بتخرج differential output واحنا متعودين في العادي اننا بناخد ال output as a single ended .. وكمان ال output اللي خارج دة current واحنا عايزينه يبقى voltage .. ممكن نحط resistance وممكن نعمل حاجة احسن بإننا نحط opamp وبالمرة هيشيل ال differential o/p وكمان

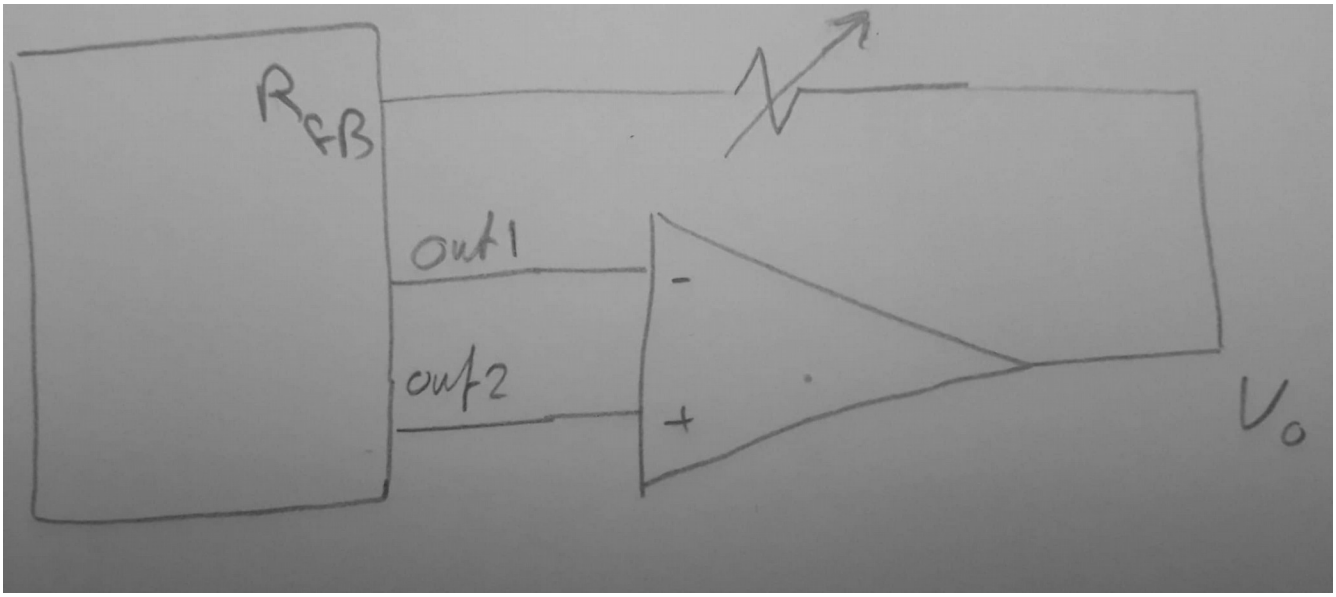


ممكن نقدر نتحكم به في ال gain :

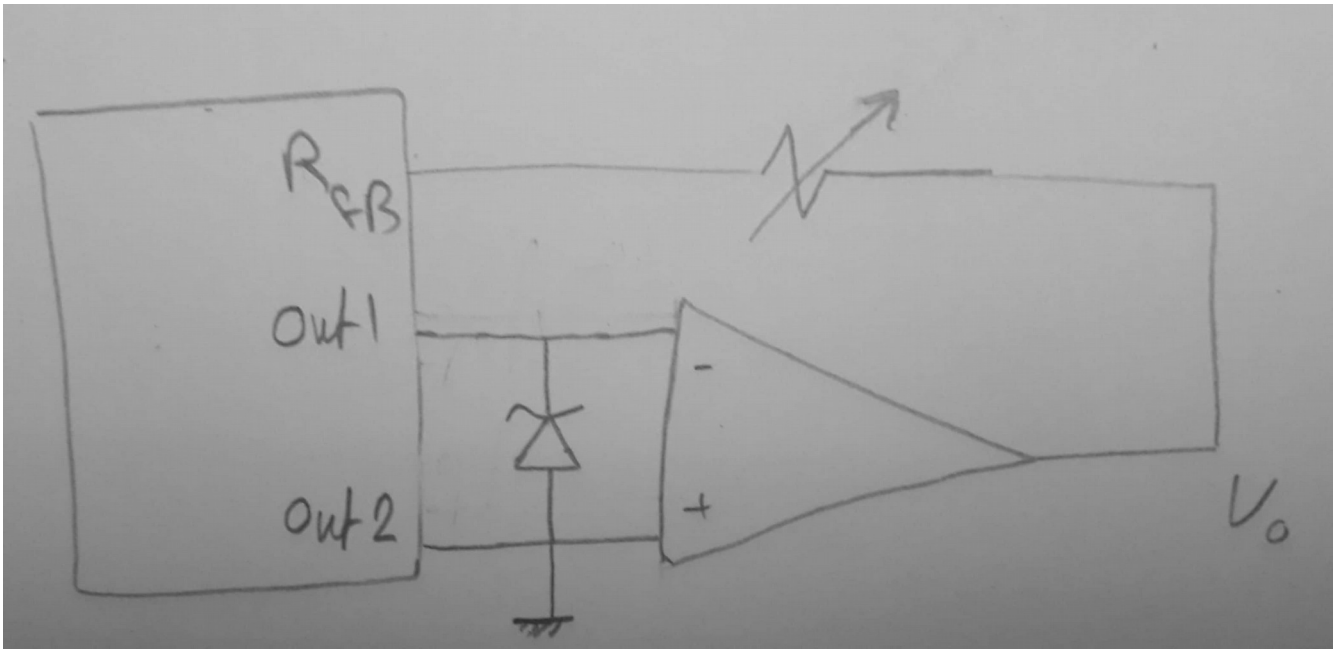
بس كدة ال opamp بقى comparator فمحتاجين نقفل ال loop ب negative feedback .. عشان كدة فيه pin في ال DAC اسمها R_{FB} .. دي متوصلة internally بحيث اننا لما نوصلها ب V_{out} تعمل negative feedback :



بس بما ان دة opamp فاحنا نقدر نتحكم في ال gain عن طريق ال feedback resistance .. واحنا مانقدرش نلعب في R_{FB} عشان دي حاجة جوة ال IC، فممكن نزود احنا potentiometer كمان من عندنا قبل مانوصل ال R_{FB} وبكدة هنقدر نتحكم في ال gain فعلاً :



كدة تمام وصّلنا ال DAC فهناخد V_{out} بقى ونوصلها على motor مثلا او أياً كان ال application ... بس افرض حصل حاجة في ال transient state خلت فيه spike تطلع على ال negative port .. ساعتها ال opamp هيطلع V_o كبيرة اوي وممكن الموتور يتحرق .. فمحتاجين نحط حاجة تحمي ال circuit لو حصل negative transient، فهنحط zener بحيث ان في ال steady state ماعندناش مشكلة فال zener هيبقى off والدنيا تمشي عادي جدا
بس لو حصل spike ساعتها ال zener هيـ breakdown وهيتوصل بال ground وال opamp مش هيشتغل لأن $V^{minus} = zero = V^{+}$ وبكدة هنبقى عملنا protection against negative transient :



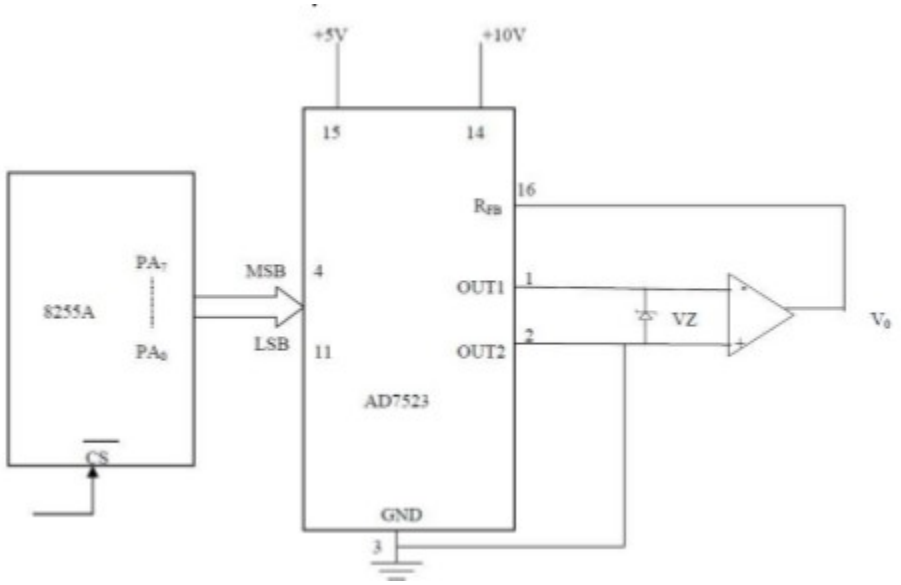
ماتشغلش بالك بال ground اللي طلع تحت دة عشان الدكتور ماتكلمش عنه وازاي نفصل بينه وبين V_{out2} .. بس يعني لو طلعت فوق شوية هتلاقي مكتوب ان ال DAC دة فيه NMOS switches فممكن يكون فيه internal switch هو اللي بيختار ground ولا V_{out2} .. المهم مالناش دعوة يعني

دلوقتي بقى عايزين نحل ال example دة :

- Interface DAC AD7523 with an 8086 CPU running at 8MHZ and write an assembly language program to generate a sawtooth waveform of period 1ms with Vmax 5V

لو جينا نفكر في ال algorithm اللي هنستخدمه في الكود هنلاقي اننا عايزين نعمل counter يبدأ من 0 ويفضل يعد لحد ما يوصل لقيمة معينة اللي هي 5v وبعد كدة ال counter يرجع تاني ل 0 ويبدأ يعد من الأول وهكذا .. سهلة اشطة

بس الحقيقة المشكلة مش في الكود .. المشكلة ان 8086 مافيهوش output latches ناخد منها ال output اللي خارج منه، فمحتاجين نحط external latches بين ال 8086 وال DAC .. اللي هي هتبقى IC اسمها 8255A ومنها هنروح لل DAC بقى:



نيجي للكود بقى:
اول حاجة هنعمل ال configurations بتاعت ال processor .. ودي هتبقى given في الامتحان :

```
ASSUME    CS:CODE
CODE SEGMENT
START:    MOV AL,80h        ;make all ports output
          OUT CW, AL
```

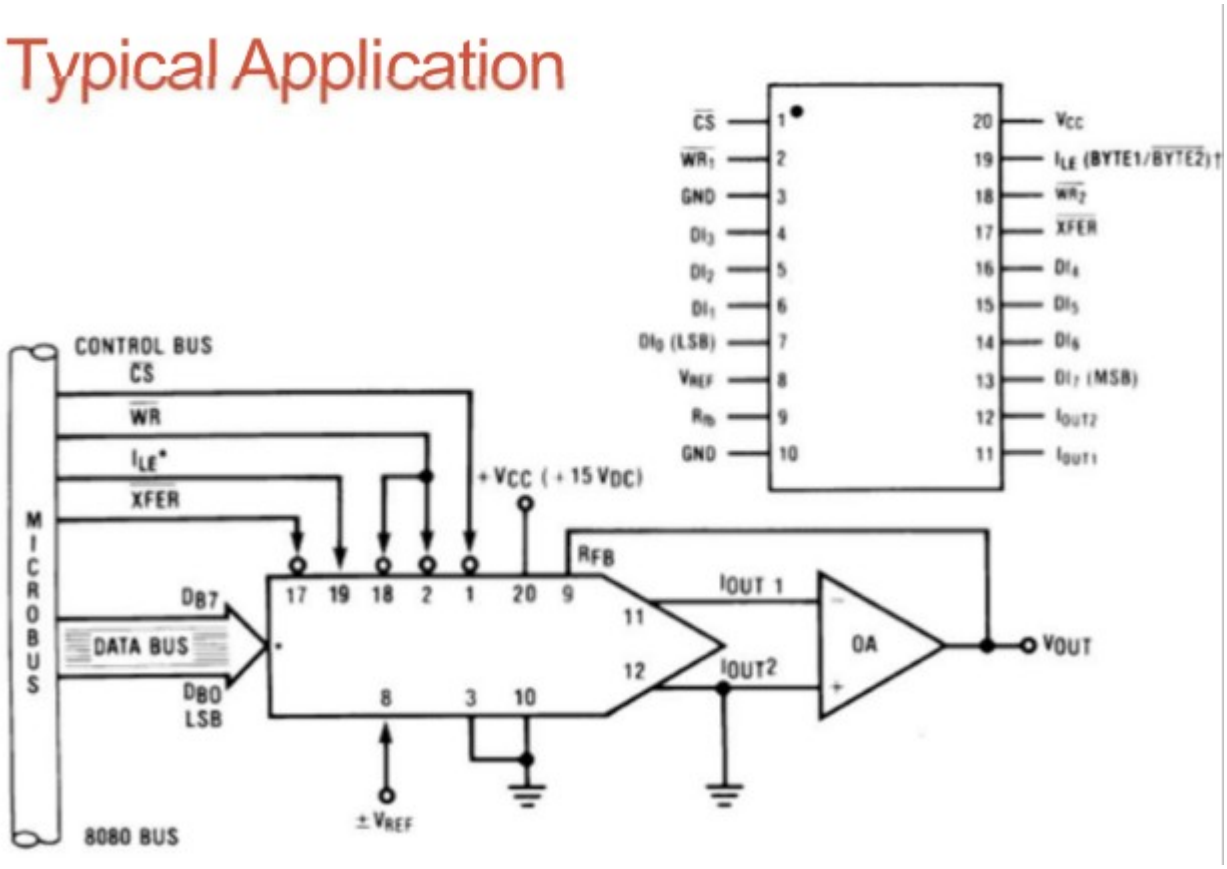
نبدأ بقى في الكود .. اول حاجة عايزين نبدأ ال counter من 0 .. وهندي الstatement دي label اسمه AGAIN عشان هنفضل نعملها على طول (infinite loop)
بعد كدة هنـ out القيمة بتاعت AL على port A اللي في 8255 وهنديله label اسمه back عشان هنفضل نرجعلها كتير بعد كدة
بعد كدة ن increment ونقارن القيمة الجديدة دي بالقيمة اللي احنا عايزين نوصل لها (اللي هي هنا هتبقى 0F2) .. لو وصلنا للقيمة دي فعلاً يبقى هنروح على AGAIN ونبدأ من ال 0 تاني .. ولو لسة ماوصلناش يبقى هنروح على BACK ونـ out القيمة دي ونعمل increment ونقارن تاني وهكذا :

```
AGAIN:    MOV AL,00h      ;start voltage for ramp
BACK:     OUT PA, AL
          INC AL
          CMP AL, 0F2h
          JB BACK
          JMP AGAIN
CODE ENDS
END       START
```

ال example اللي فات دة لازم هييجي في الامتحان .. بس هيبقى فيه لعب طبعاً، الدكتور قال ان اللعب هيبقى في ال coding مش في ال interfacing .. يعني مثلا ممكن يقول اعمل sawtooth بس ب slope اكبر .. ساعتها بدل ما هنـ increment ب 1 هنزود اكتر فهنـ increment ب 2 مثلا او 3
وممكن يبقى عايز ال ramp تطلع براحتها فساعتها هنحط delay ... وممكن يقول عايز حاجة تانية زي sin wave مثلا وساعتها هيبقى فيه lookup table بناخد منه القيم اللي هنطلعها

طيب ال 0F2 دي بقى جت منين؟؟ .. المفروض في السؤال هيدينا كل instruction بتاخد كام clk cycle .. واحنا معانا ال frequency فهنعرف نجيب ال t_{clk} .. هنضرب ال t_{clk} في ال total number of cycles وهنبقى عرفنا ال loop دي بتتنفذ في اد ايه بالظبط ..
احنا عايزين ال sawtooth تتعمل في 1 msec فهنقسم ال 1 msec دي على اللي هيطلع من نتيجة الضرب اللي فاتت ونحول اللي هيطلع دة ل hex

في سلايد 20 بيقول ان دلوقتي فيه DAC's فيها internal latches فممكن نوصلها مع ال microprocessors على طول حتى لو كان حاجة زي 8086 :



كدة خلصنا ال DAC .. ندخل في ال ADC :

ال ADC هندرس منه 3 انواع : Digital ramp – successive approximation – flash ADC
أسرعم هو ال flash ADC زي ما هنشوف بعد شوية وأبطأهم هو ال digital ramp وأنسبهم في الاستخدام هو ال SAR

فكرة ال ADC عامةً أياً كان نوعه هي انه بيستخدم DAC و comparator و counter عشان يطلع ال most approximate value ..بس مش ال exact value
ولازم يكون فيه start of conversion & end of conversion pulses عشان زي ماهو واضح ال ADC بيحتاج وقت عشان يعرف يعمل conversion صح، على عكس ال DAC

Digital ramp:

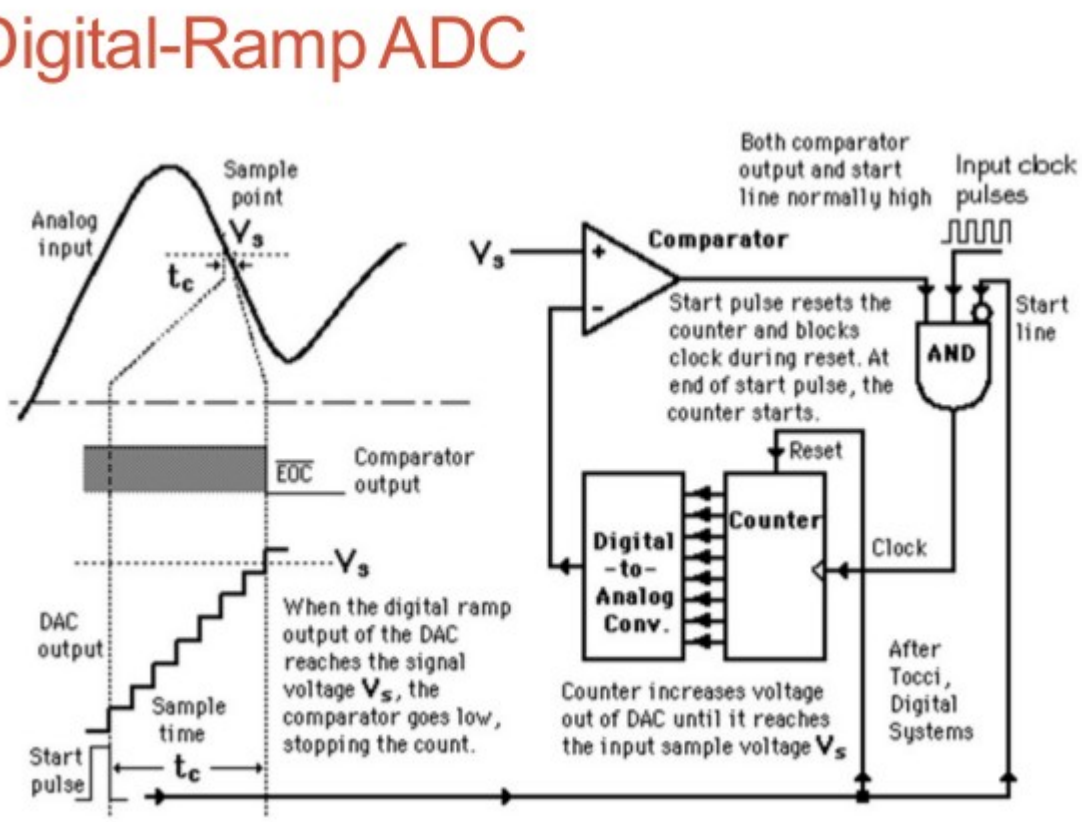
لو انت فاهم بيشتغل ازاي اعمل skip للكلام اللي جي دة

دلوقتي احنا عملنا sampling وطلعنا volt قيمته V_s وخلاص نقدر نبدأ ال conversion .. فال start pulse بتتـ activate وهي اصلا متوصلة مع ال reset بتاع ال counter .. فأول ما تبقى ب 1 ال counter هيتعمل له reset فهيبدأ من الصفر

هيخرج ال output بتاعه على DAC وال DAC هيحول ال output دة ل analog value .. بعد كدة ال value دي هتدخل على comparator يقارنها بال V_s .. لو كان ال V_s اكبر يبقى هيخرج V_{cc} .. فعايزين ندخل ال output بتاع ال comparator دة على ال clock بتاعت ال counter عشان نقوله ان احنا لسة ماوصلناش للقيمة بتاعتنا فيزود 1 ... بس فيه حاجة، ال ADC دة synchronous اصلا فالمفروض ان فيه clock بتاعت السيستم داخلة عليه .. فالمفروض ان ال counter عشان يشتغل صح زي ماحنا عايزين يبقى لازم ال system clock دي تكون positive edge وكمان ال output بتاع ال comparator يبقى 1 .. عشان كدة بندخلهم الاول على and gate

المهم هنفضل نعمل كدة لحد لما في الاخر ال DAC يطلع قيمة اكبر من V_s وال comparator يطلع 0 فال and gate تطلع 0 فقيمة ال counter ماتتغيرش وساعتها يبقى ال conversion خلص

و فيه signal تالتة داخلة على and gate اللي هي ال start pulse بس complemented ودي محطوطة كزيادة تأكيد ان ال counter دايماً بيبدأ من 0



واضح طبعاً ان البتاع اللي فات دة بطئ جدا ودايماً بيبدأ من الصفر ويفضل يزود في نفسه لحد ما يوصل للقيمة اللي عايزينها .. فليه مايبقاش فيه algorithm اسرع مايخلينا نفضل نعد كدة وخلاص؟؟

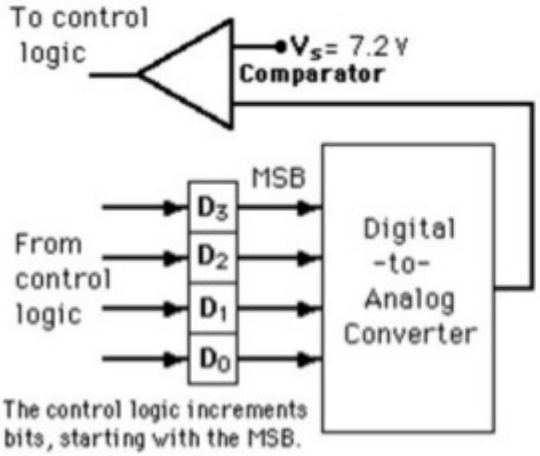
SAR ADC:

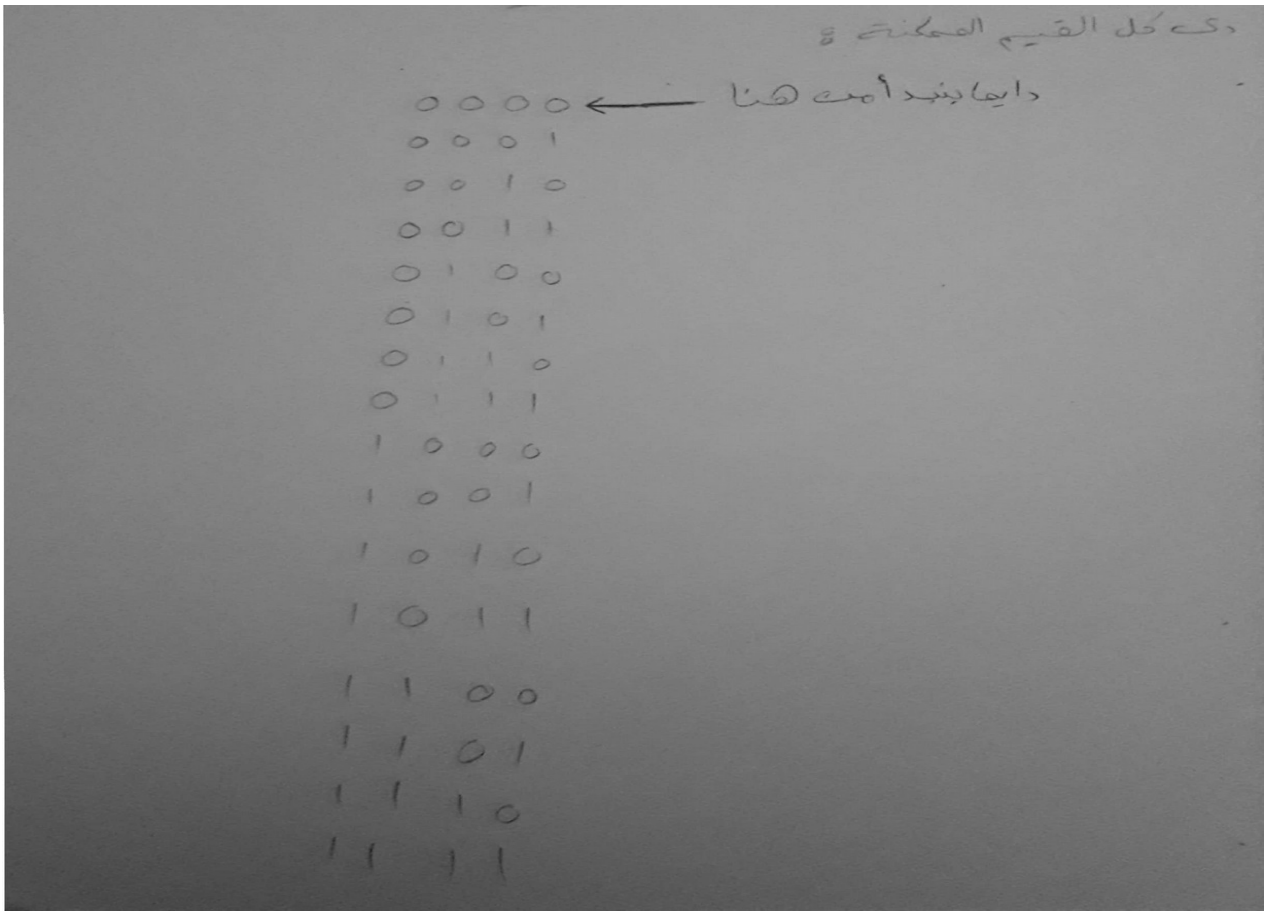
فكرته انه دايماً بيبدأ من الصفر وفيه comparator برضه وفيه counter بس ال counter دة جوة control logic فمش بيعد وخلاص زي ال digital ramp .. طب بيعمل ايه؟؟

بينط على نص ال value على طول (بيخلي ال MSB ب 1) وبعد كدة يدخلها على DAC عشان يحولها لأنالوج ويقارنها .. لو ال V_s اكبر من القيمة اللي طلعت من ال DAC يبقى المفروض ال digital value دي

تكون اكبر فهيكنسل كل القيم اللي اصغر منها وبرضه هيروح يشوف نص ال value الجديدة (يعني هيخلي ال 2^{nd} MSB ب 1) ولسة ال 1^{st} MSB ب 1 .. وبعد كدة ويقارنها وهكذا
ولو كانت اصغر يبقى كان هيكنسل كل القيم اللي اكبر منها وبرضه يشوف نص ال value الجديدة (اللي هي 2^{nd} MSB ب 1 بس ال 1^{st} MSB بقت ب 0)

لو مش فاهم تعالى نشوف مثال احسن :
لو عندنا four-bit SAR ADC وكانت ال step بتاعته ب 1 فولت وال V_s ب 7.2v :





دى كل القيم الممكنة :

دايما بنبدأ منك هنا ← 0 0 0 0 ⇐ 0 Volt

0 0 0 1

0 0 1 0

0 0 1 1

0 1 0 0

0 1 0 1

0 1 1 0

0 1 1 1

خلينا الـ MSB بـ 1 فوصلنا هنا ← 1 0 0 0 ⇐ 8 Volt "أطبر من 7.2"

يعنى القيمة اللى بندور عليها هتبقى فوق من تحت..

فهنكمل كل اللى تحت

ونخلى الـ MSB بـ 0

والـ 2nd MSB بـ 1

1 0 0 1

1 0 1 0

1 0 1 1

1 1 0 0

1 1 0 1

1 1 1 0

1 1 1 1

دي كل القيم الممكنة 8

دايما بنبدأ من هنا ← 0 0 0 0 ⇐ 0 Volt

0 0 0 1

0 0 1 0

0 0 1 1

0 1 0 0 ← صفرنا الـ 1st MSB وخلينا الـ 2nd MSB بـ 1 فوصلنا هنا ⇐ 4 Volt "أصغر من 7.2"

0 1 0 1

0 1 1 0

0 1 1 1

1 0 0 0 ← خلينا الـ MSB بـ 1 فوصلنا هنا ⇐ 8 Volt "أكبر من 7.2"

يعني القيمة اللي بندور عليها هتبقى فوق من تحت..
فهنكمل كل اللي تحت
ونخلي الـ MSB بـ 0
والـ 2nd MSB بـ 1

1 0 0 1

1 0 1 0

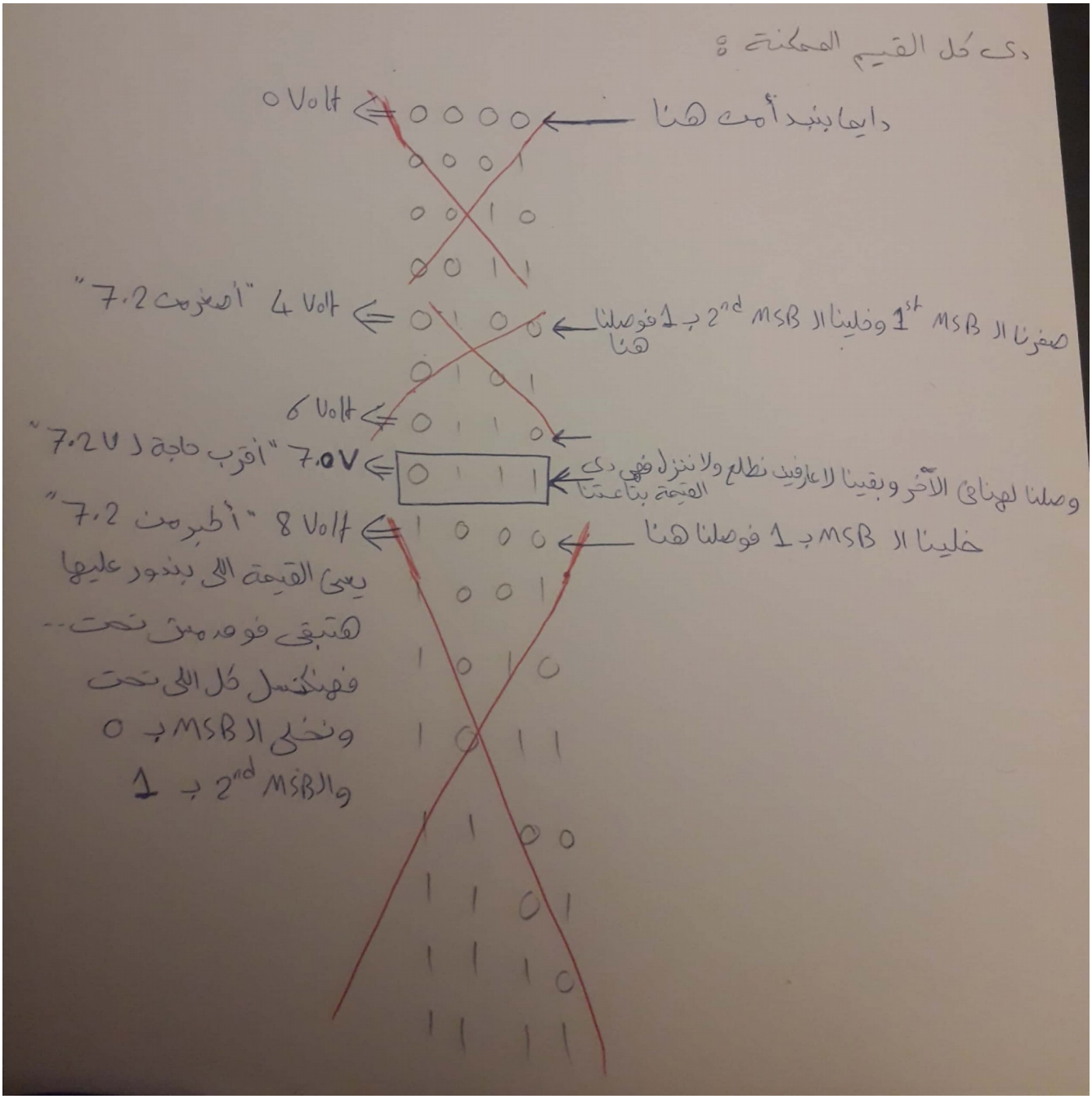
1 0 1 1

1 1 0 0

1 1 0 1

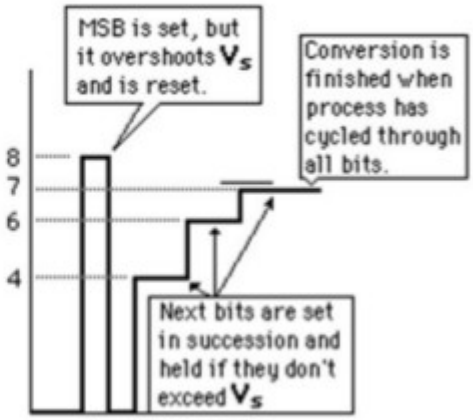
1 1 1 0

1 1 1 1

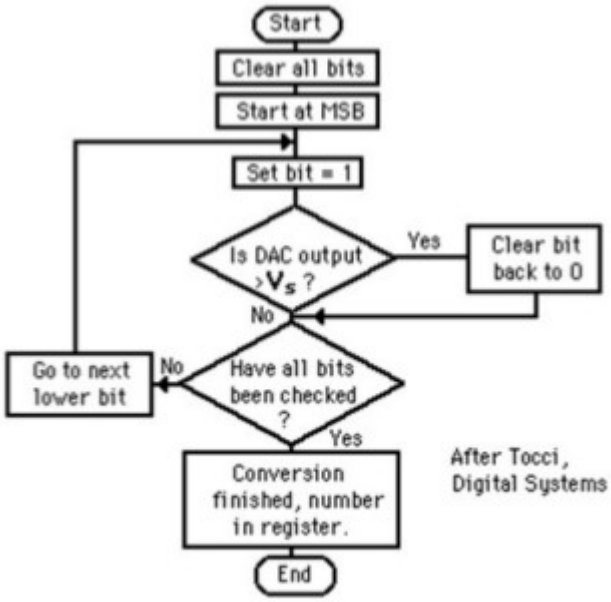


في الاخر القيمة بتاعتنا بقت 0111 ... بص بقى كدة احنا عملنا comparing كام مرة .. عملنا 5 مرات في حين ان في ال digital ramp كنا هنعمل ال compare دة 8 مرات .. وطبعا كل ما عدد ال bits يزيد كل ما يبقى الفرق اكبر

عايزين بقى نرسم ال outputs اللي خرجت من ال DAC من اول ما بدأنا conversion لحد ما خلصنا :
في الأول بدأنا ب 0 وبعد كدة طلعنا ل 8 (بعد كدة المفروض بيحصل reset فنرجع لل 0 تاني بعد كدة نروح لل 4 زي ما موجود في الرسمة اللي جاية .. بس الدكتور قال انه مفترض ان احنا فاهمين ان دة اللي بيحصل فاحنا مش هنعمل كدة في الامتحان وهنروح على ال 4 على طول من غير مانرجع لل 0) .. بعد كدة روحنا على 4 بعد كدة 6 وبعد كدة 7 :

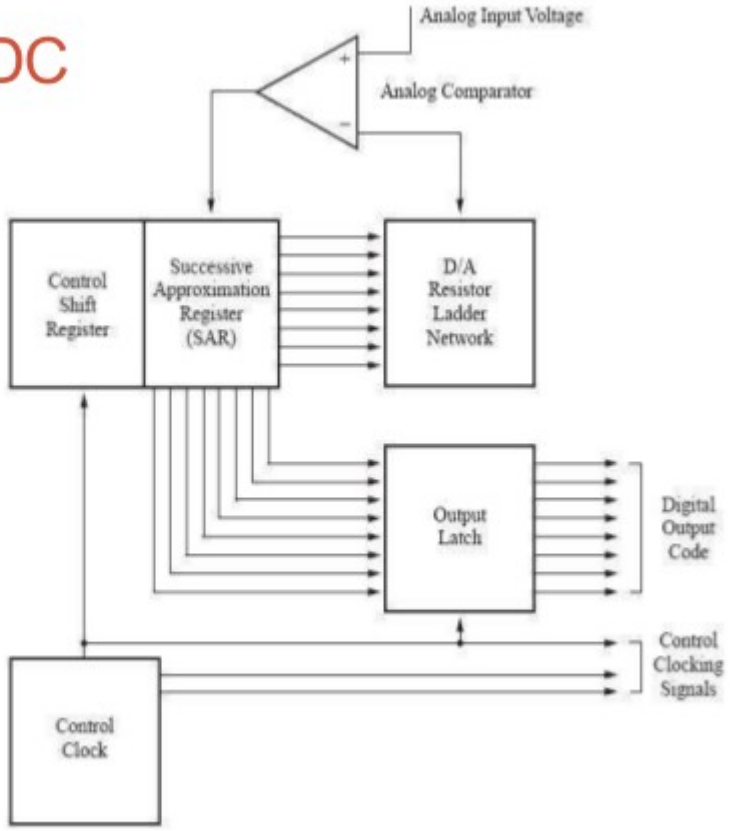


ودة flow chart ملخص كل الكلام اللي فوق دة:



ودة ال block diagram بتاع ال SAR :

Successive Approximation ADC



Flash ADC:

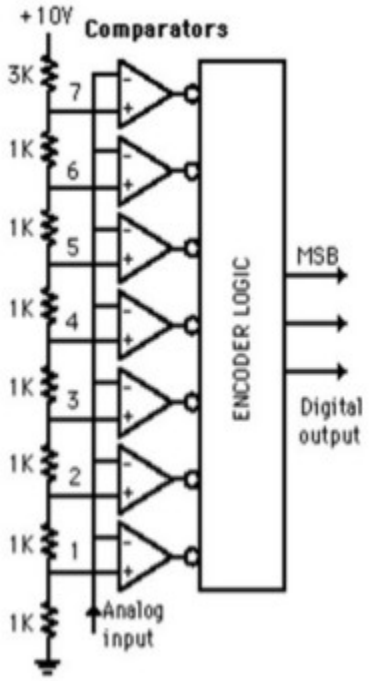
دة زي ماقلنا فوق هو اسرع واحد في ال 3 لأن مافيهوش كذا stage .. هو بس comparator و encoder وخلاص شكرا

وعدد ال comparators فيه بيكون عدد ال Steps - 1

.. يعني لو three bit ADC يبقى فيه seven comparators

Flash ADC

- It is the fastest type of ADC available, but requires a comparator for each value of output
  - (63 for 6-bit, 255 for 8-bit, etc.)
- Such ADCs are available in IC form up to 8-bit and 10-bit flash ADCs (1023 comparators) are planned
- The encoder logic executes a truth table to convert the ladder of inputs to the binary number output



Illustrated is a 3-bit flash ADC with resolution 1 volt

فوق خالص هتلاقي فيه V_{ref} اللي هو حاططها هنا ب 10 فولت وال input بيدخل من تحت خالص ومتوصل بال negative port بتاعت كل comparator
فكرة البتاع دة معتمدة على ال voltage divider بحيث ان ال comparator اللي فوق خالص يبقى بيقارن بأكبر value
تعالى ناخد مثال :
لو دخلنا input = 4.5 v .. طبعا ال input دة هيدخل على كل ال comparators في نفس الوقت .. هيقارن قيمته بأول comparator .. اللي هو داخل عليه :

$$\text{Voltage} = (10\,volt) * \frac{1K\Omega}{(3+1+1+1+1+1+1+1)K\Omega}$$

قيمة ال voltage دة هتطلع ب 1 volt (اصغر من 4.5) فال comparator هيخرج 0 .. وهكذا ال input هيفضل يتقارن بكل comparator ويشوف هيطلع 1 ولا 0 .. وبعد كدة كل ال outputs اللي خرجت من ال comparators دي هتدخل على ال encoder يخرج ال digital output

من الكلام اللي احنا قلناه دة واضح ان ال flash ADC مافيهوش اي delay غير ال propagation delay بس في المقابل فيه مشاكل زي ال size مثلا اللي هيكبر جدا كل ما عدد ال bits يزيد وبالتالي power وال cost هيزيد

فلو جينا نقارن ال conversion time بتاع ال digital ramp وال flash ADC هنلاقي ان ال flash بيبقى ال delay بتاع بال micro seconds وال digital ramp بال Melliseconds

السلايدز اللي جاية دي بتلخص كلام المحاضرة كله في main points فممكن تشوفها :

Interfacing Analog to Digital Converters

- ADC is treaded as an input device by the microprocessor.
- Microprocessor sends an initializing signal to the ADC to start the A-D data conversation process
  - Start of conversation (SOC) signal is a pulse of a specific duration
- The process of analog to digital conversion is a slow process
  - Microprocessor has to wait for the digital data till the conversion is over
- After the conversion is over, the ADC sends end of conversion (EOC) signal to inform the microprocessor that the conversion is over and the result is ready at the output buffer of the ADC
- The tasks of issuing an SOC pulse to ADC, reading EOC signal from the ADC and reading the digital output of the ADC are carried out either directly by the CPU or using 8255 I/O ports

Interfacing Analog to Digital Converters

- The time taken by the ADC from the active edge of SOC pulse till the active edge of EOC signal is called as the *conversion delay* of the ADC
- It may range any where from a few microseconds in case of fast ADC to even a few hundred milliseconds in case of slow ADCs
- The available ADC in the market use different conversion techniques for conversion of analog signal to digitals.
  - Successive approximation techniques and dual slope integration techniques are the most popular techniques used in the integrated ADC chip

General Algorithm for ADC Interfacing

1. Ensure the stability of analog input, applied to the ADC
   - Sample and hold circuit
     - Samples the analog signal and holds it constant for a specific time duration
     - Microprocessor may issue a hold signal to the sample and hold circuit
2. Issue start of conversion pulse to ADC
3. Read end of conversion signal to mark the end of conversion processes
4. Read digital data output of the ADC as equivalent digital output
   - If the applied input changes before the complete conversion process is over, the digital equivalent of the analog input calculated by the ADC may not be correct

ADC0808/0809:

السلايد اللي جاية دي الدكتور ماركزش فيها غير على اخر نقطة:

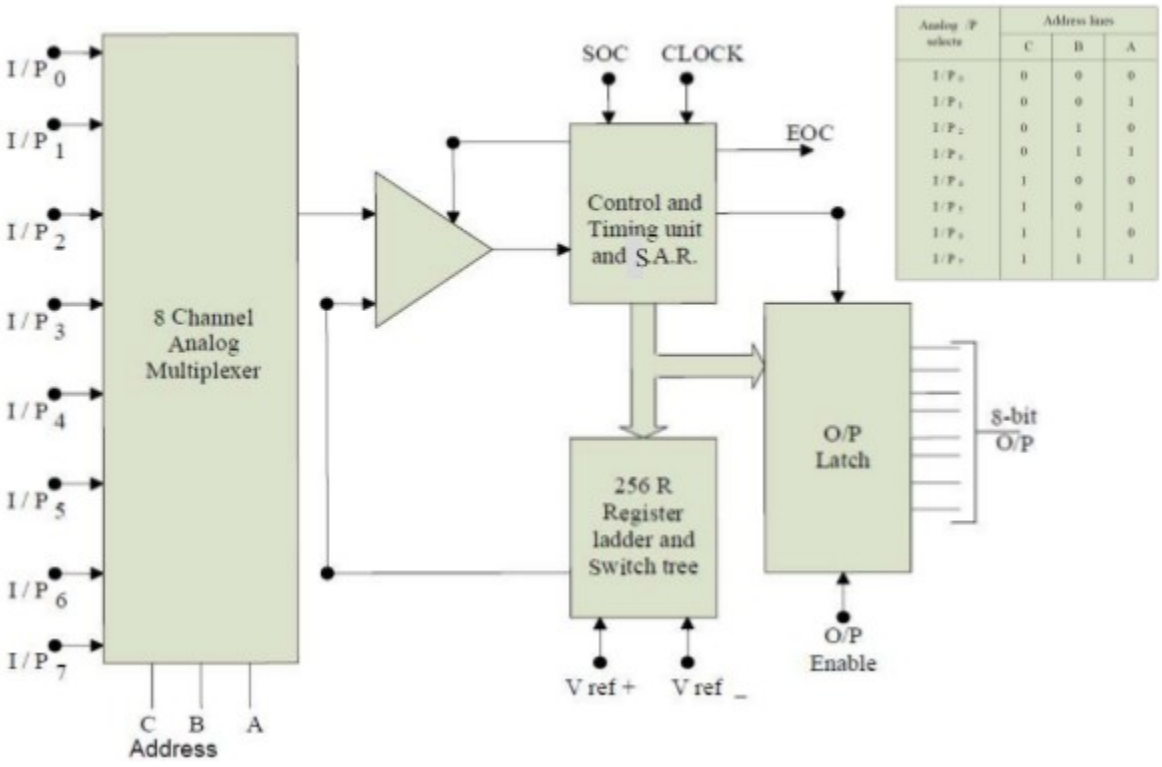
ADC 0808/0809

- 8-bit CMOS, successive approximation converters
  - One of the fast techniques for analog to digital conversion
    - Conversion delay is 100μs at a clock frequency of 640 KHz, which is quite low as compared to other converters
- There are unipolar analog to digital converters, i.e. they are able to convert only positive analog input voltage to their digital equivalent
- These chips do not contain any internal sample and hold circuit
- These converters do not need any external zero or full scale adjustments as they are already taken care of by internal circuits
- These converters internally have a 3:8 analog multiplexer so that at a time 8 different analog conversion by using address lines - ADD A, ADD B, ADD C
  - Using these address inputs, multichannel data acquisition system can be designed using a single ADC
    - The CPU may drive these lines using output port lines
  - In case of single input applications, these may be hardwired to select the proper input

من الاخر يعني احنا نقدر نستخدم كذا channel من ال ADC في نفس الوقت .. نحط مثلا heat sensor مع smoke detector مع LDR مع مش عارف ايه لحد ما نحط 8 حاجات مع بعض على ال ADC

ودة لأن ال ADC0808 دة فيه mux الاول بيختار كل مرة مين منهم اللي هيدخل على ال ADC عن طريق 3 control signals اسمهم ADD A, ADD B, ADD C

Block Diagram of ADC 0808 / 0809



ال pin description بتاعه مايفرقش حاجة عن ال ADC's اللي فاتوا ماعدا حاجة واحدة:

ADC 0808 / 0809 Pin Description

- Vcc Supply pins +5V
- GND GND
- Vref+ Reference voltage positive +5 Volts maximum.
- Vref- Reference voltage negative 0Volts minimum.
- I/P0 –I/P7 Analog inputs
- ADD A,B,C Address lines for selecting analog inputs.
- O7 – O0 Digital 8-bit output with O7 MSB and O0 LSB
- SOC Start of conversion signal pin
- EOC End of conversion signal pin
- OE Output latch enable pin, if high enables output
- CLK Clock input for ADC

الحاجة الزيادة هنا عن اللي فاتوا هي ال OE .. ودي مالهاش لازمة اوي بس محطوطة كزيادة تأكيد إن مافيش حاجة هتخرج من ال output latch غير لو ال EOC & OE هم الاتنين ب 1

اخر example في المنهج بقى ♥♥

Example: Interfacing 0808 with 8086

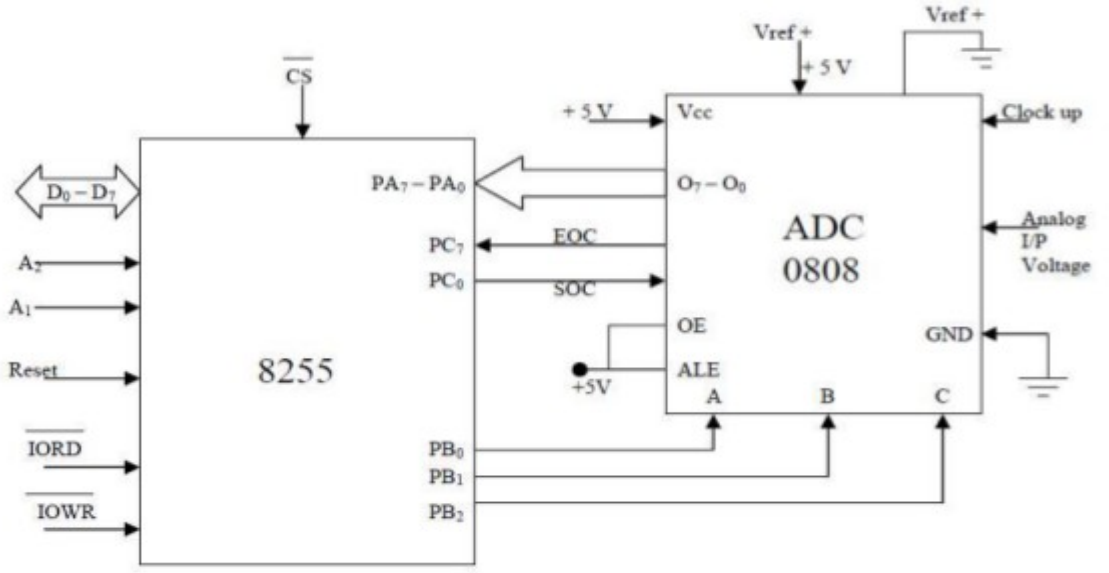
- Interfacing ADC 0808 with 8086 using 8255 ports. Use port A of 8255 for transferring digital data output of ADC to the CPU and port C for control signals. Assume that an analog input is present at I/P2 of the ADC and a clock input of suitable frequency is available for ADC.

فيه حاجة ماتكلمناش فيها واحنا بنتكلم عن ال 8255 في اول المحاضرة .. ال IC دي ليها three ports .. Port A, B, C
port A, B نقدر نخليهم كلهم input او كلهم output
بس port C هو الوحيد اللي ممكن نخلي النص الأولاني منه.. ال (lower part) بتاعه input والنص التاني output
نرجع بقى لل example .. هو عايزنا نستخدم port A عشان ياخد الداتا من ال ADC لل CPU .. و port C لل control signal (بس خلي بالك هو ماكانش لازم يقول ان port C يبقى هو ال control عشان احنا عارفين ان هو الوحيد اللي ممكن يبقى شوية منه input وشوية output فمافيش غيره هو اللي ينفع لل control)
فيبقى اكيد port B هو اللي هيدخل على ال address signals بتاعت ال ADC
واحنا عايزين ندخل i/p 2 على ال ADC يعني المفروض ال ***address pins (A,B,C)*** يبقوا 010 …
خلي بالك ABC دول ال address pins بتاعت ال ADC مش ال ports بتاعت 8255

- **Solution**: The analog input I/P2 is used and therefore address pins A,B,C should be 0,1,0 respectively to select I/P2. The OE and ALE pins are already kept at +5V to select the ADC and enable the outputs.
  - 8255 Port C upper acts as the input port to receive the EOC signal while port C lower acts as the output port to send SOC to the ADC
  - 8255 Port A acts as a 8-bit input data port to receive the digital data output from the ADC

وهيبقى دة ال block diagram بتاع ال interface :

Interfacing 0808 with 8086



زي ماهو واضح وصلنا اللي خارج من ال ADC على port A واللي منه دخل على ال microprocessor و أخدنا اول three pins من port B ودخلناهم على ال address pins A,B,C

وأخدنا اول pin في port C اللي هي output وبقت هي ال SOC و اخر pin اللي هي input وبقت هي ال EOC والباقي مايهمناش

الكود بقى .. دة ال configuration اللي مالناش دعوة به وهيبقى given في الامتحان:

```
MOV    AL, 98h          ;initialise 8255 as
OUT    CWR, AL          ;discussed above.
```

عايزين ن select ال second input فهنخرج 02 على port B :

```
MOV    AL, 02h          ;Select I/P2 as analog
OUT    Port B, AL       ;input.
```

كدة ظبطنا الدنيا .. نبدأ conversion بقى … اول حاجة محتاجين نعمل ال SOC، بيتعمل لما يحصل transition بين low -high – low

ففي الأول هنحط 00 على port C ونـ out .. بعد كدة هنحط 01 ونـ out وبعد كدة 00 ونـ out وبكدة ال SOC اتعملت :

```
MOV    AL, 00h        ;Give start of conversion
OUT    Port C, AL     ; pulse to the ADC
MOV    AL, 01h
OUT    Port C, AL
MOV    AL, 00h
OUT    Port C, AL
```

كدة ال ADC شغال فاحنا عايزين نقعد نـ check كل شوية على ال EOC عشان نعرف ال conversion هيخلص امتى .. طب هنعمل كدة ازاي؟؟
هنعمل loop نقعد نقرا فيها ال value بتاعت port C كل مرة ونشوف لو PC7 بقى ب 1 .. فبنعمل compare وطول ما ال compare بيطلع 0 يبقى احنا لسة في ال loop .. اول ما يبقى ب 1 يبقى خلاص ال conversion خلص ونقدر ناخد الداتا ... دي الفكرة اللي الدكتور قالها في المحاضرة

بس في السلايد هو عمل حاجة تانية .. عمل حاجة اسمها rotate carry right (RCR) ... عشان قيمة P_{c7} تتحط في ال carry .. خلي بالك P_{c7} دي هي ال LSB مش ال MSB
وبعد كدة عمل JNC WAIT عشان يـ jump تاني على ال loop لو كان ال carry مش ب 0

```
WAIT: IN     AL, Port C     ;Check for EOC by
      RCR                   ; reading port C upper and
      JNC    WAIT           ;rotating through carry.
```

بس لو كان ب 0 يبقى خلاص كدة ال EOC حصل ونقدر نقرا ال value :

```
IN     AL, Port A     ;If EOC, read digital equivalent in AL
HLT                   ;Stop.
```

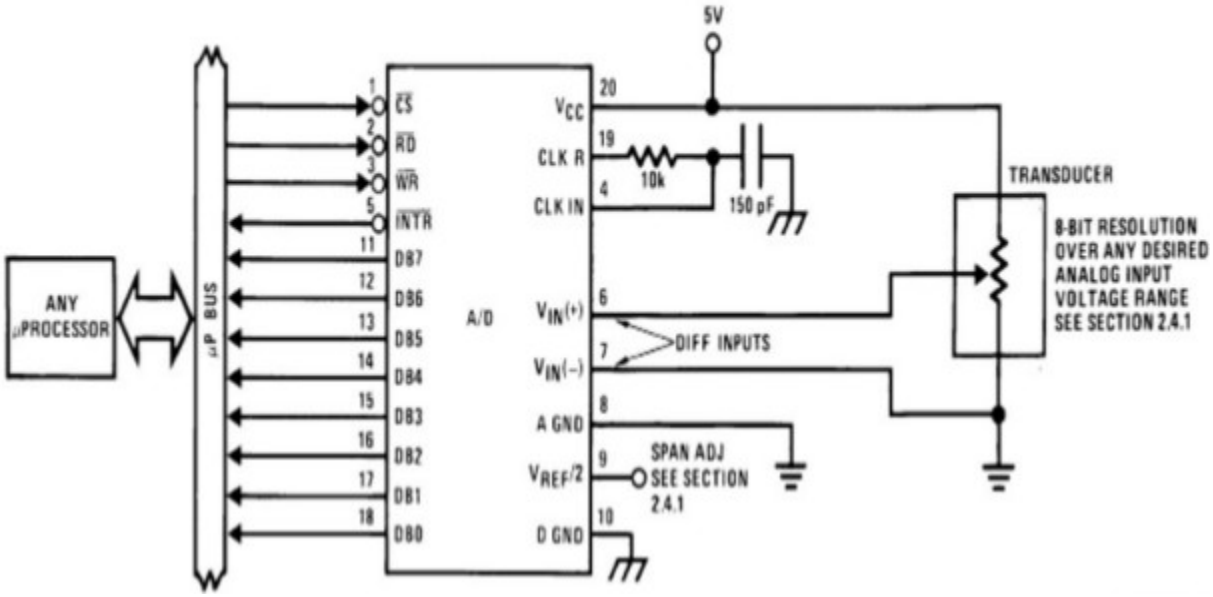
ودة الكود كله من غير التقطيع اللي حصل دة :

ADC Assembly Control Code

```
        MOV    AL, 98h        ;initialise 8255 as
        OUT    CWR, AL        ;discussed above.
        MOV    AL, 02h        ;Select I/P2 as analog
        OUT    Port B, AL     ;input.
        MOV    AL, 00h        ;Give start of conversion
        OUT    Port C, AL     ; pulse to the ADC
        MOV    AL, 01h
        OUT    Port C, AL
        MOV    AL, 00h
        OUT    Port C, AL
WAIT:   IN     AL, Port C     ;Check for EOC by
        RCR                   ; reading port C upper and
        JNC    WAIT           ;rotating through carry.
        IN     AL, Port A     ;If EOC, read digital equivalent in AL
        HLT                   ;Stop.
```

اخر slide دي الدكتور ماتكلمش عنها خالص بس شوفها برضه :

ADC080x, 8-Bit Microprocessor Compatible A/D Converters interfacing



شوية كلام بقى عن الامتحان .. هيبقى فيه جزء MCQ
وجزء تاني عن رسمة معينة (زي الحاجات اللي قبل الميدترم او USB او UART)
وسؤال كود زي اللي في المحاضرة دي (ممكن نكتبه من الاول خالص او يجيلنا مكتوب غلط واحنا نصلحه) ..
وممكن ييجي سؤال interface زي بتاع المحاضرة دي برضه

تم بحمد الله ^^