Набацана ОРТ Скрипта, надам се да ће некоме бити од помоћи 🙂

Садржај

[Секвенцијална кола 2](#_Toc188569173)

[Употреба 2 комплемента 3](#_Toc188569174)

[Флип-Флопови 3](#_Toc188569175)

[РС Флип-Флоп 3](#_Toc188569176)

[JK Флип Флоп 4](#_Toc188569177)

[Т Флип-флоп 5](#_Toc188569178)

[Д Флип-флоп 6](#_Toc188569179)

[Врсте излаза 7](#_Toc188569180)

[Регистри 7](#_Toc188569181)

[Бројачи 8](#_Toc188569182)

[Паралелни бројачи 9](#_Toc188569183)

[Серијски бројачи 9](#_Toc188569184)

[Машина са коначним бројем стања 9](#_Toc188569185)

[Меморијске јединице 10](#_Toc188569186)

[РАМ – Random Access Memory 10](#_Toc188569187)

[Врсте РАМ меморије 11](#_Toc188569188)

[SRAM 11](#_Toc188569189)

[DRAM 11](#_Toc188569190)

[Формуле 12](#_Toc188569191)

# Секвенцијална кола

Код комбинационих кола, на излазу стања зависе само од трентних стања, што значи да она немају могућност праћења стања, односно меморисања. На примјер, ф-ја ***f(ABCD)*** зависи од А у датом тренутку, али већ у следећем тренутку када се А промијени, функција ***f*** зависи од новог стања и заборавља претходне податке.

Комбинациона кола су кола која зависе само од тренутног улаза, док су секвенцијлна кола кола која зависе и од **тренутног** и од **претходног** улаза.

Нека од секвенцијалних кола су:

1. Флип-Флопови
   1. РС Флип-флоп
   2. Д Флип-флоп
   3. ЈК Флип-флоп
   4. Т Флип-флоп
2. Регистри
   1. Паралелни
   2. Shift регистри
3. Бројачи
   1. Асихрони
   2. Сихрони
   3. Усмјерени
   4. Декадни
   5. Модуло бројачи
4. Меморијски
   1. SRAM
   2. DRAM
   3. ROM

## Употреба 2 комплемента

Реализација одузимача је знатно једноставнија уколико посматрамо одузимање као сабирање негативних бројева него као на саме негативне бројеве. Претварањем бројева у други комплемент, исто коло можемо користити и за сабирање и за одузимање.

## Флип-Флопови

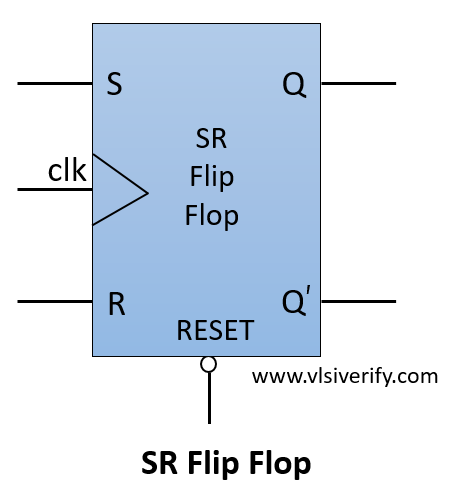
### РС Флип-Флоп

Код РС Флип-флопова почетно стање је логичка 0. А стање коме одговара логичка 1 назива се сетовано (SET).   
У овом флип-флопу постоје два улаза:

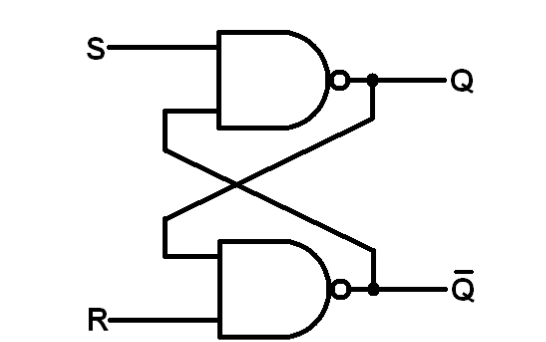
1. R (Reset) – служи за уписивање логичке 0.
2. S (Set) – Служи за уписивање логичке 1

Табела истинитости:

|  |  |  |  |
| --- | --- | --- | --- |
| Тn | | | Tn+1 |
| R | | S | Qn+1 |
| 0 | 0 | | Qn |
| 0 | 1 | | 1 |
| 1 | 0 | | 0 |
| 1 | 1 | | Недефинисано стање |



Слика 1 - Симбол RS флип флопа

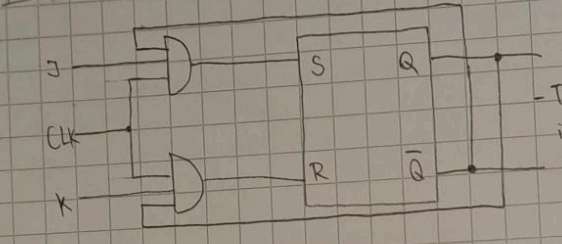


Слика 2 - Имплементација помоћу NAND кола

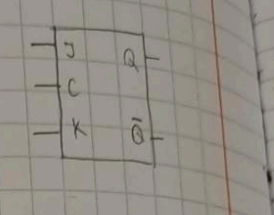
### JK Флип Флоп

За разлику од РС Флип Флопа, ЈК нема недозвољена стања. Ресетовање се врши при к=1 а сетовање при Ј=1, као код РС.

Може бити и са директном побудом (асихрони). Једина разлика је што се реализује без тактног сигнала (CLK).



Слика 3 - шематски приказ синхроног ЈК флип флопа



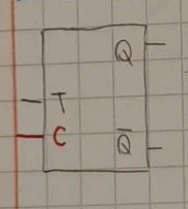
Слика 4 - Симбол ЈК Флип Флопа

**Табела истинитости:**

|  |  |  |
| --- | --- | --- |
| **Ј** | **К** | **Qn+1** |
| 0 | 0 | Qn |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | Qn’ |

### Т Флип-флоп

Т флип-флоп има само један улаз за довођење побудног сигнала. Начин функционисања је да при сваком увођењу сигнала 1 на улаз Т мијења стање на излазу.



Слика 5 - Симбол Т Флип-Флопа

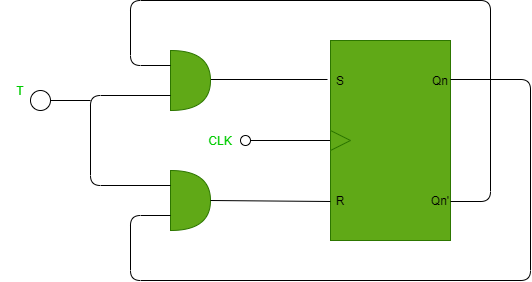
Уколико постоји улаз C значи да је флип-флоп **тактовано**. То значи да да у њега улази CLK и у том слуају, флип-флоп је **СИНХРОН.**

**Табела истинитости:**

|  |  |  |
| --- | --- | --- |
| **Т** | **Qn** | **Qn+1** |
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |

**Qn+1 = T\*Q’+T’Q = T XOR Q**

Овај флип-флоп се може имплементирати преко РС флип флопа тако да је Q=TQ а S=TQ’



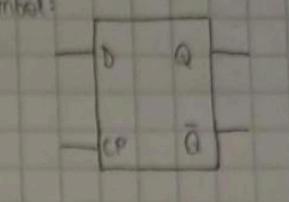
Слика 6 - шема Т флип флопа преко РС флип флопа

### Д Флип-флоп

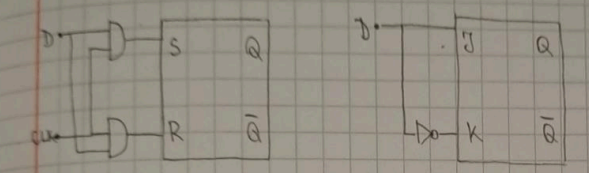
Д флип флоп је један од најкоришћенијих флип флопова и служи за чување једног бита података. Његова функција је да прослиједи вриједност на улазу на излаз у тренутку актиирања такта (CLK).

Табела истинитости:

|  |  |  |
| --- | --- | --- |
| D | Qn | Qn+1 |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



Слика 7 - симбол Д Флип-флопа



Слика 8 - Реализација преко СР и ЈК флип флопа

### Врсте излаза

1. Активан излаз – излаз је укључен/активиран; нпр. Активан **SET** улаз на РС флип-флопу, излаз је постављен на 1 – активан излаз
2. Неактиван излаз - излаз је искључен; нпр. Ако су **SET** и **RESET** РС флип-флопа деактивирани, излаз остаје исти – неактиван излаз

## Регистри

Регистар је стандардна секвенцијална мрежа које се користи за привремено складиштење података, те се сходно томе често назива и – ***привремена меморија.*** Њена улога је да прихвати и привремено меморише улазне податке, дјеломичне резултате или коначне резултате у *процесу обраде података*. Користе се за довођење података и прихватање резултата при аритметичким операцијама.

Регистар је скуп меморијских јединица – флип флопова. Број флип-флопова у регистру зависи од његовог предвиђеног капацитета. Како флип-флоп може меморисат само један податак, број бита информација је и укупан број флип-флопова у регистру.

Главна карактеристика регистра је велика брзина функционисања с обзиром на њихову примјену треба бити јако велика. **Веома су скупи**.

Регистри се дијеле према:

1. Начину извођења:
   1. Стационарни
   2. Динамички
      1. Помјерачки (*shift*) регистри
      2. Кружни регистри
2. Улази и излази података
   1. Серијски
      1. Подаци се преносе бит по бит, повезани су један за другим у низу, захтијевају мање физичког простра, али пренос података може трајати дуже јер се преноси бит по бит.
   2. Паралелни
      1. Сви битови се преносе истовремено, сваки елемент повезан је са одговарајућиим елементима у другим регистрима; бржи су али захтијевају више ресурса и енергије.

### Бројачи

Бројачи су секвенцијалне мреже које се користе у дигиталним системима и дигиталним рачунарима за бројење импулса и мјерење временских интервала. Он се реализује тако што се мјере промјене сигнала које му долазе на улаз. Бројачи имају један улаз на који улази сигнал и више излаза на којима се додају комбинације које одговарају броју доведених инпулса на улаз.

На излазима се генеришу бинарне комбинације у одређеном редослиједу тако да се могу интерпретирати као низ сукцесивних бројева.

Бројачи могу бити:

1. Инкременторски
2. Декременторски
3. Обострани

### Паралелни бројачи

Нису погодни за велике бројевне вриједности, флип-флопови су серијски везани, а спреге између флип-флопова се остварују преко одговарајућих комбинационих мрежа које требају обезбиједити адекватан начин функционисања.

У оваквој врсти бројача, сви флип-флопови су синхронизовани са истим **CLK** сигналом. Логичка кола (обично И, ИЛИ и НИЛИ) одређују када ће сваки флип-флоп промијенити стање.

**Један улаз може ићи на више улаза, али више излаза се НЕ МОГУ кратко спојити!**

### Серијски бројачи

Захтијевају мање хардвера у односу са паралелним бројачима, једноставнији су за дизајн и имплементацију. Код њих, излаз једног флип-флопа служи као улаз за следећи; први флип-флоп добија улазни ***CLK*** док остали флип-флопови реагују на излазе претходних

Нису погодни за примјене које захтијевају високе брзине.

## Машина са коначним бројем стања

Понашање секенцијалних блокова зависи од скупа испушених услога али и од садржаја интерне меморије, претходног стања и слања система. Зато се они не могу описати таблицом иситнитости него се називају ***машине са коначним бројем стања***

**Садржи две функције:**

1. **Функција следећег стања (Next State):**
   * Одређује следеће стање на основу тренутног стања и улаза.
2. **Функција излазне вредности (Output Function):**
   * Дефинише излаз на основу тренутног стања (или улаза и стања).

**Компоненте:**

* Регистар стања (меморија за чување тренутног стања).
* Функција за одређивање следећег стања.
* Функција излазне вредности.

**Врсте машина са коначним бројем стања:**

* **Мурова машина:**
  + Излаз зависи само од тренутног стања.
  + Једноставнија је и захтева мање стања.
* **Милијева машина:**
  + Излаз зависи од тренутног стања и улазних сигнала.

**Реализација и примена:**

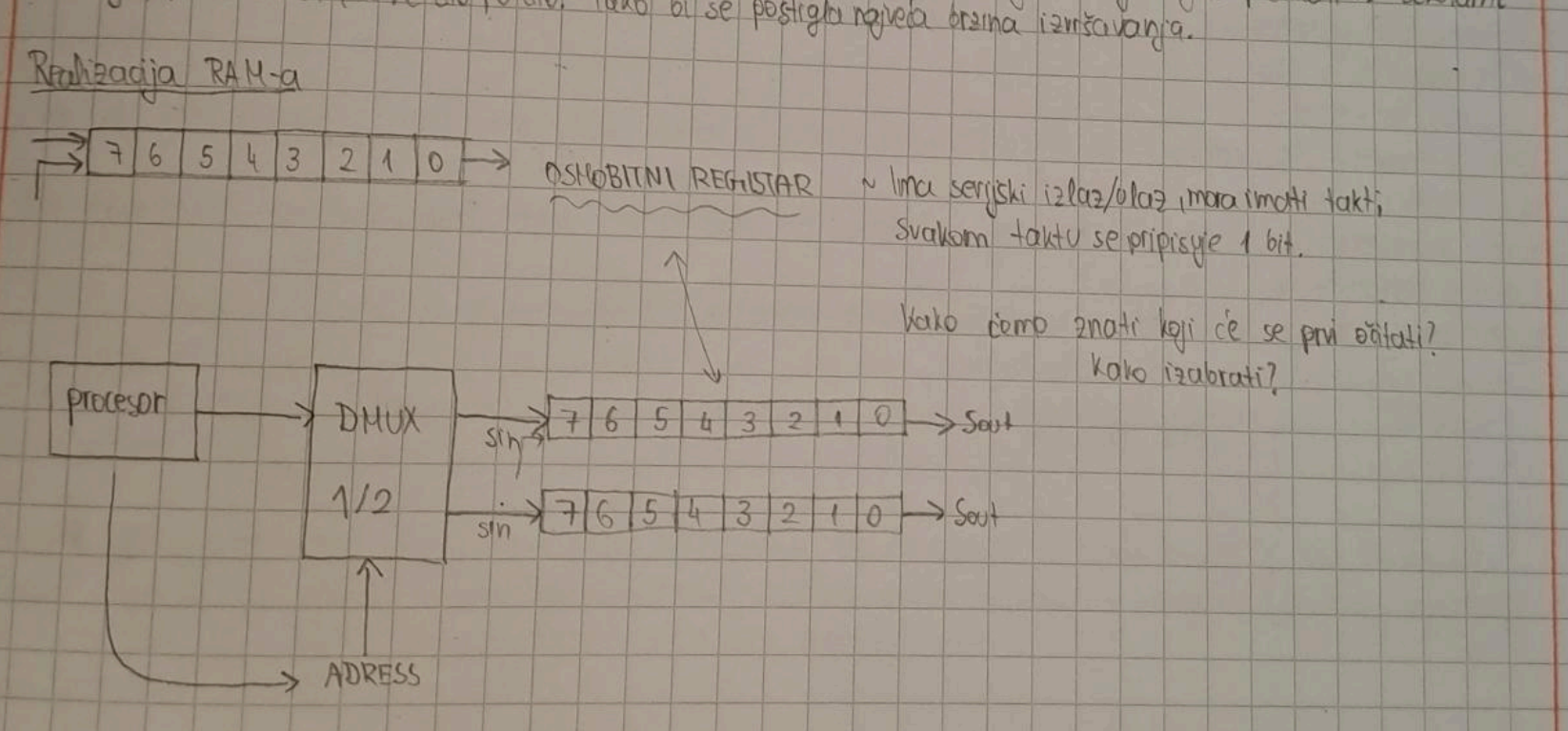
* Машине се могу представити:
  + **Дијаграмом стања:** Приказује сва стања и прелазе између њих.
  + **Табелом истине:** Садржи све могуће комбинације стања, улаза и одговарајуће излазе.
  + **Логичким колима:** Комбиновање функција стања помоћу логичких капија.

# Меморијске јединице

## РАМ – Random Access Memory

RAM, означава врсту меморије која је директно адресибилна и њеном садржају се може приступити по произвољној локацији, а не само редом (секвенцијално, као код трака). RAM дозвољава да се подаци узимају директно у насумичном редоследу.

Најчешће се користи у рачунарима као примарна или главна меморија, мада то није неопходно. Садржај меморије РАМ-а је пожељно да буде што већег капацитета да би се читав програм могао пренијети из секундарне меморије у РАМ и извршавати, а не да се извршава дио по дио. Као посљедица тога, долази до најбржег извршавања.



## Врсте РАМ меморије

РАМ се, у општем случају, може подијелити на двије главне врсте, ***SRAM*** и ***DRAM.***

### SRAM

Представља статички тип меморије. У њему податак задржава вриједност све докле год постоји напајање. Битови се имплементују преко флип-флопова, оптимално састављених од 6 транзистора. Брзина одзива је од 2-4 милисекунде.

Његова најчешћа примјена, због вискоке брзине, је употреба у кеш меморији.

### DRAM

Представља динамички тип меморије. У њему се сваки податак постављен преко једног транзистора, што га чини мање сложеног од SRAM-a. Главна карактеристика је динамички карактер податка, што значи да се подаци могу периодично освјежавати како би остале непромјењене.

Примарна употреба је главна меморија на рачунару. А вријеме одзива је око 6 милисекунди.

# Формуле

Вријеме извршавања процесора: T=N×CPI×1/f  
 T – vrijeme izvršavanja

N – broj instrukcija u program

CPI: Broj ciklusa po instrukciji

F - frekvencija

Број циклуса: N\*CPI

Перформансе: 1/T

ЦПИ: Broj ciklusa/Broj instrukcija

ИПЦ: N/Broj instrukcija

Међусобна повезаност