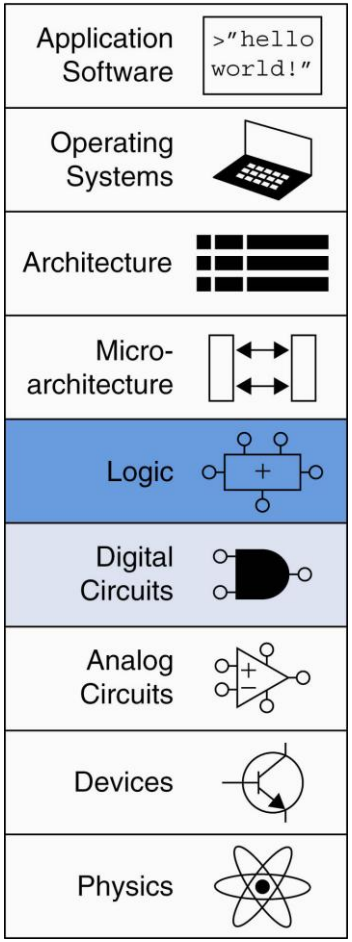
КАРХ: Тема_5: Последователни логически схеми

Въведение.

- Състоянието на изходите на последователните логически схеми зависят от **текущите** и **предишните** състояния на входните величини, т.е. имат **памет**.
- Дефиниции:
  - **Състояние (State):** цялата информация за схемата, необходима за обяснение на бъдещото ѝ поведение.
  - **Тригери (Latches and flip-flops) :** елементи съхраняващи 1 бит състояние (информация).
  - **Синхронни последователни схеми (Synchronous sequential circuits):** комбинационна логика, съчетана с известно количество тригери.



КАРХ: Тема_5: Последователни логически схеми

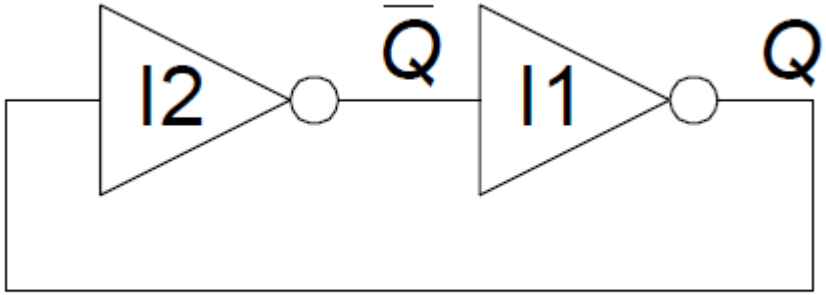
Характеристики и елементи.

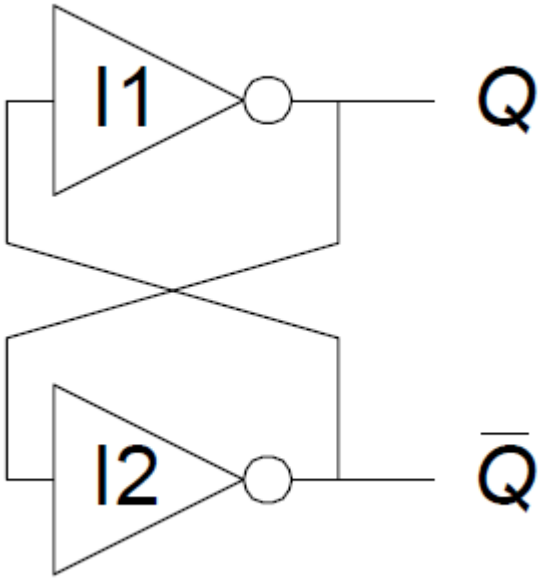
- Представят последователност от събития.
- Имат памет (кратковременна).
- Използват обратна връзка (от изхода – към входа) за съхранение на информация.
- Текущото състояние на схемата влияе на бъдещото ѝ поведение.
- Елементи съхраняващи състоянието:
  - Bistable circuit (Бистабилни схеми)
  - SR Latch (RS Тригери)
  - D Latch (D Тригери)
  - D Flip-flop (Двустъпални D Тригери, Мастър-слейв D Тригери)

КАРХ: Тема_5: Последователни логически схеми

<u>Би-стабилна схема.</u> (Bistable Circuit)

- Основен изграждащ блок на останалите елементи на състояние.
- Два изхода: $Q, \overline{Q}$
- Недостатък – няма входове.

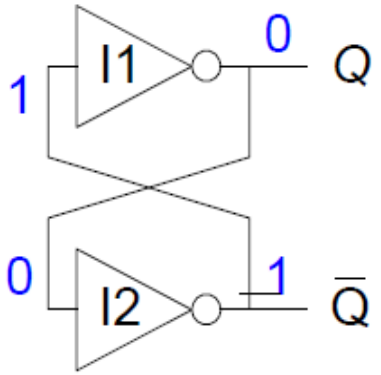




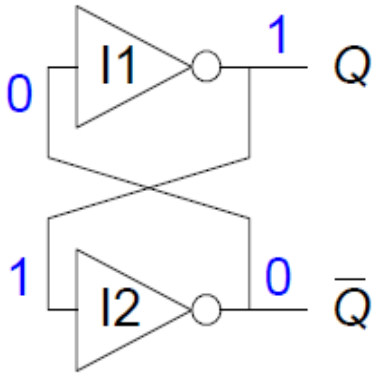
КАРХ: Тема_5: Последователни логически схеми

Анализ на би-стабилната схема.

- Разглеждат се два възможни случая:
  - Допускане, че **Q = 0:**
  - тогава $\bar{Q} = 1, Q = 0$ (в съгласие с допускането)
  - Допускане, че **Q = 1:**
  - тогава $\bar{Q} = 0, Q = 1$ (в съгласие с допускането)



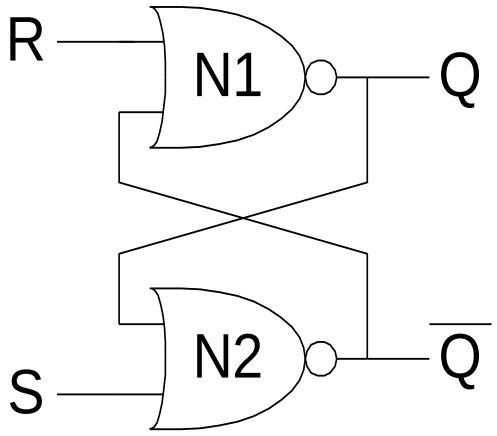
- Запомня се 1 bit състояние на променливата – Q (или $\bar{Q}$).
- **Но няма входове за контрол на състоянието!**



КАРХ: Тема_5: Последователни логически схеми

RS Тригер. (SR (Set/Reset) Latch)

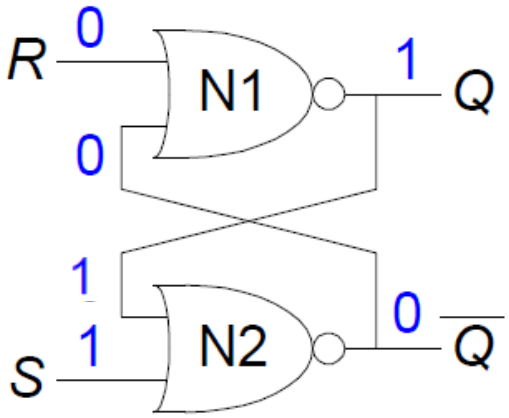
- Разглеждат се 4 възможни случая:
  - $S = 1, R = 0$
  - $S = 0, R = 1$
  - $S = 0, R = 0$
  - $S = 1, R = 1$



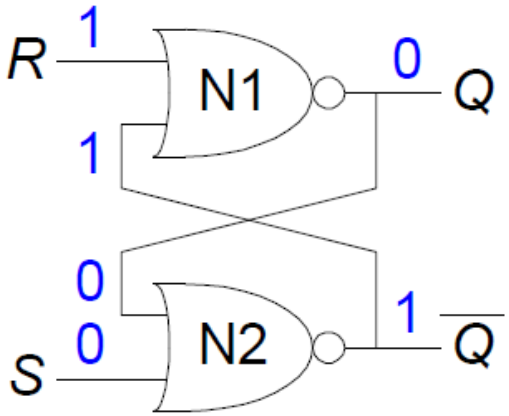
КАРХ: Тема_5: Последователни логически схеми

Анализ на RS Тригер.

- **$S = 1, R = 0$:**
- тогава $Q = 1$ и $\overline{Q} = 0$



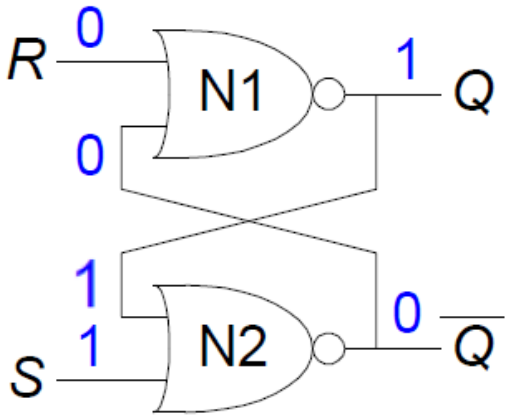
- **$S = 0, R = 1$:**
- тогава $Q = 0$ и $\overline{Q} = 1$



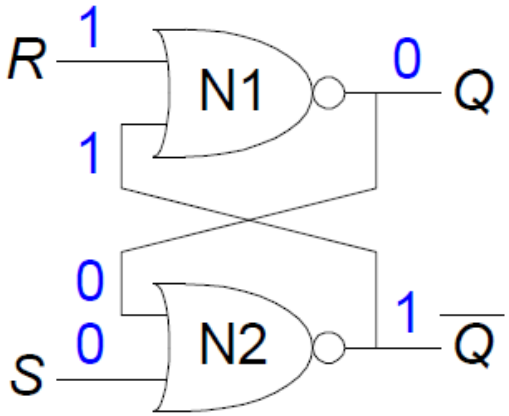
КАРХ: Тема_5: Последователни логически схеми

Анализ на RS Тригер.

- **$S = 1, R = 0$:**
- тогава $Q = 1$ и $\overline{Q} = 0$
- ***Set* (1) на изхода *Q***



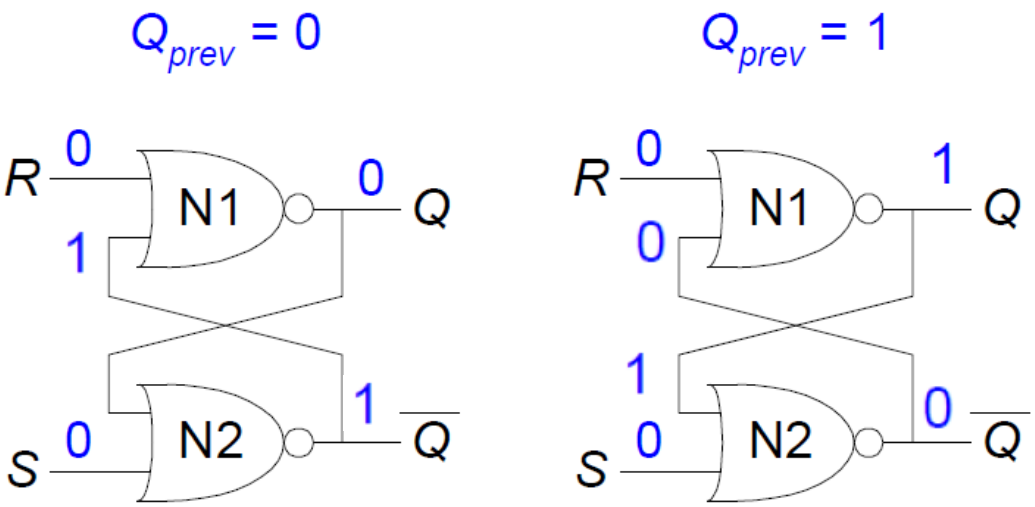
- **$S = 0, R = 1$:**
- тогава $Q = 0$ и $\overline{Q} = 1$
- ***Reset* (0) на изхода *Q***



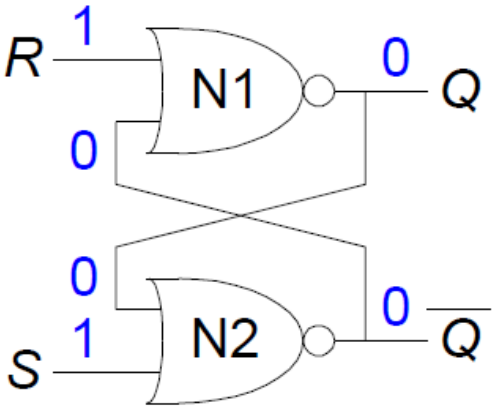
КАРХ: Тема_5: Последователни логически схеми

Анализ на RS Тригер.

- **$S = 0, R = 0$:**
- тогава $Q = Q_{prev}$



- **$S = 1, R = 1$:**
- тогава $Q = 0$, $\overline{Q} = 0$

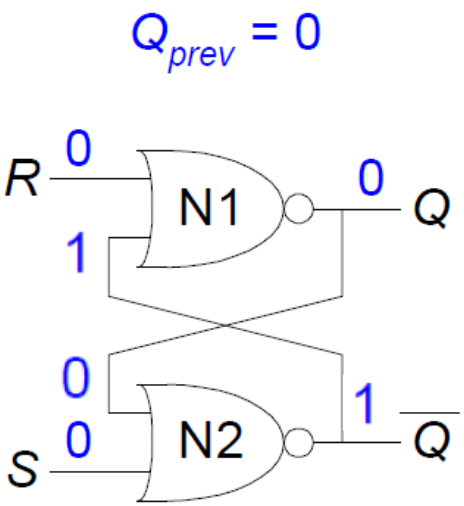


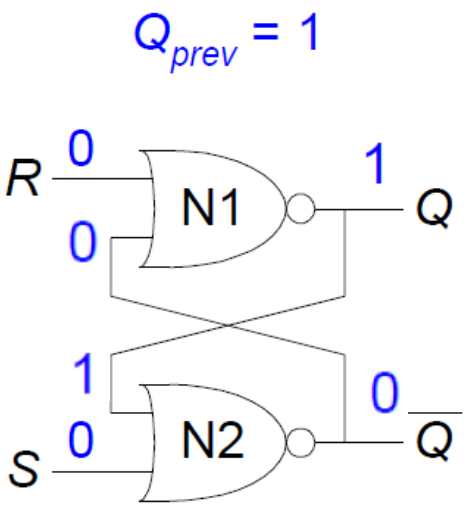
КАРХ: Тема_5: Последователни логически схеми

Анализ на RS Тригер.

– ***S* = 0, *R* = 0:**
– тогава $Q = Q_{prev}$

Памет!

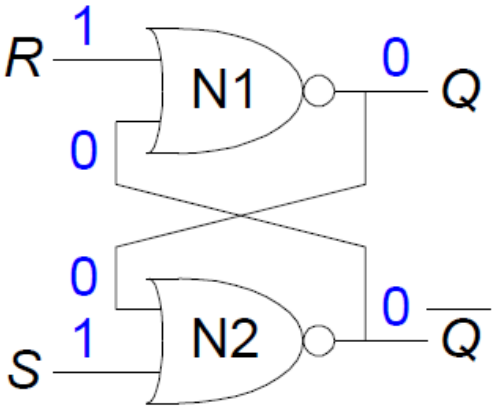




– ***S* = 1, *R* = 1:**
– тогава $Q = 0, \overline{Q} = 0$

Невалидно състояние

$\overline{Q} \neq \text{NOT } Q$

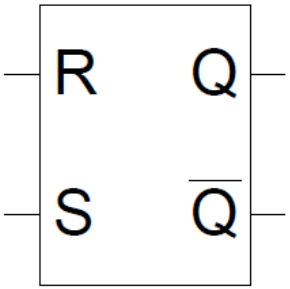


КАРХ: Тема_5: Последователни логически схеми

Анализ на RS Тригер.

- SR е съкращение за Set/Reset Latch
  - Съхранява 1 bit състояние (Q)
- Контролира (Управлява) записаната стойност чрез S, R входовете
  - **Set:** Прави изхода 1
  - ($S = 1$, $R = 0$, $Q = 1$)
  - **Reset:** Прави изхода 0
  - ($S = 0$, $R = 1$, $Q = 0$)

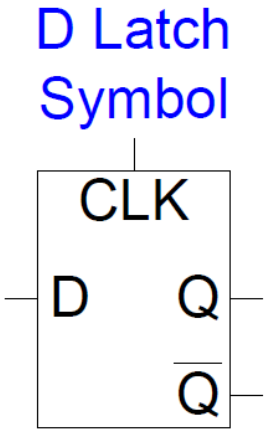
SR Latch Symbol



КАРХ: Тема_5: Последователни логически схеми

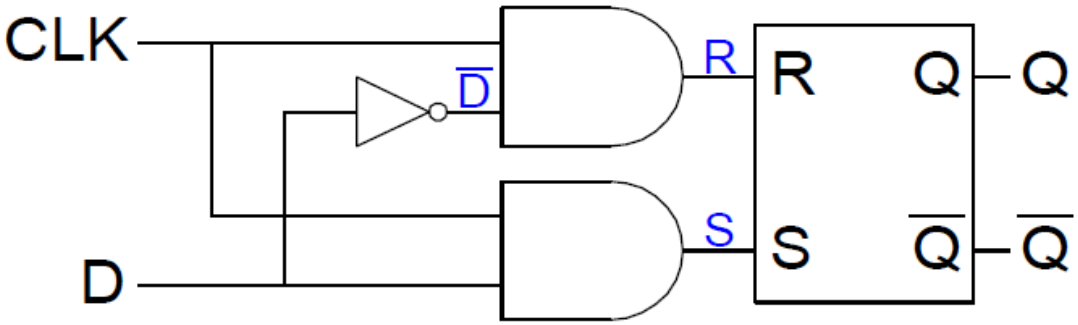
D Тригер. (D Latch)

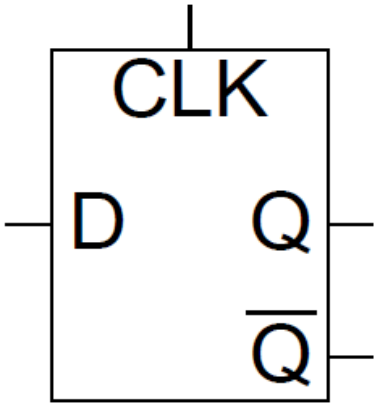
- Два входа: *CLK*, *D*
  - ***CLK*:** контролира *кога* да стане смяна на изхода
  - ***D*** (the data input): контролира *какво* да излезе на изхода
- Функция
  - Когато ***CLK* = 1**,
  - *Състоянието на D* се прехвърля на *Q* (*прозрачен*)
  - Когато ***CLK* = 0**,
  - *Q* запазва предишното си състояние (стойност) (*непрозрачен*)
- Избягва се невалидния случай когато $Q \neq \text{NOT } \bar{Q}$



КАРХ: Тема_5: Последователни логически схеми

D Тригер. Схемна реализация.

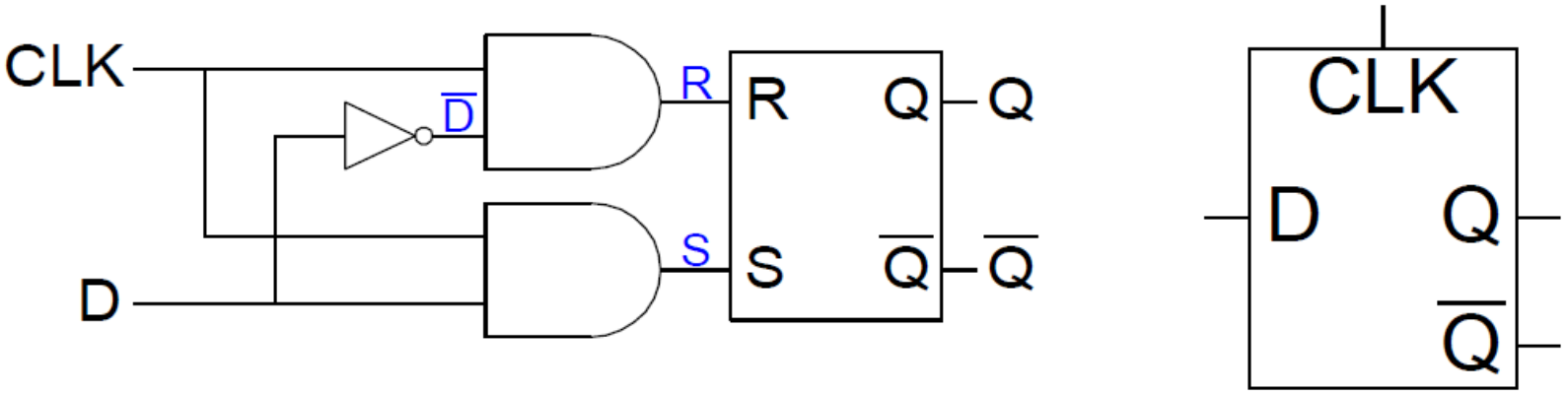




CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X					
1	0					
1	1					

КАРХ: Тема_5: Последователни логически схеми

D Тригер. Схемна реализация.

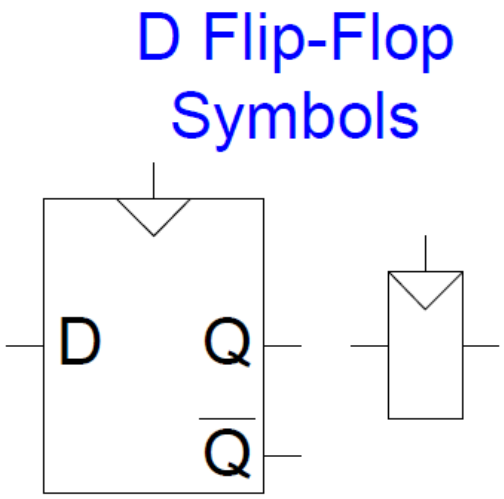


CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

КАРХ: Тема_5: Последователни логически схеми

MS (Master – Slave) D Тригер. (D Flip-Flop)

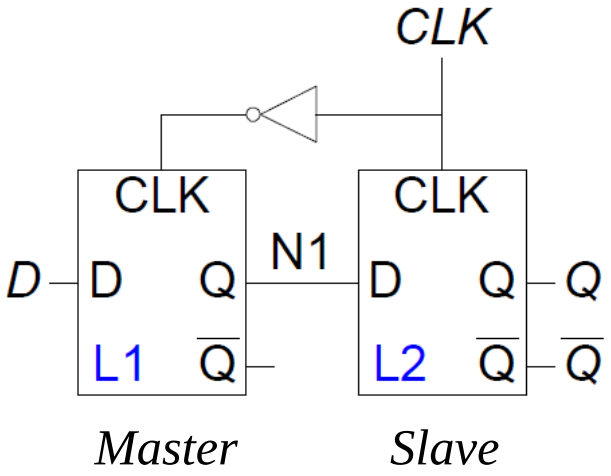
- **Входове:** *CLK*, *D*
- **Функция**
  - Отчита стойността на *D* при нарастващия фронт на *CLK*:
    - Когато *CLK* нараства от 0 към 1, стойността на *D* се прехвърля на *Q*;
    - В останалите случаи *Q* запазва предишната си стойност.
  - *Q* се променя *само* при *нарастващия фронт на CLK*.
- Нарича се превключващ се (активиращ се) по фронт *(edge-triggered)*.



КАРХ: Тема_5: Последователни логически схеми

MS (Master – Slave) D Тригер. Схемна реализация.

- Два последователно свързани D latch (L1 and L2) контролирани комплементарно.
- Когато ***CLK* = 0**
  - L1 е прозрачен (transparent)
  - L2 е непрозрачен (opaque)
  - *D* се прехвърля към N1

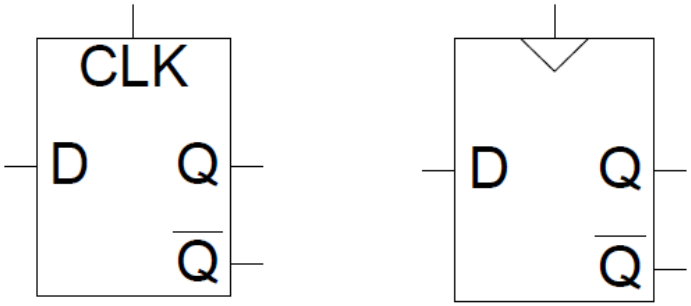


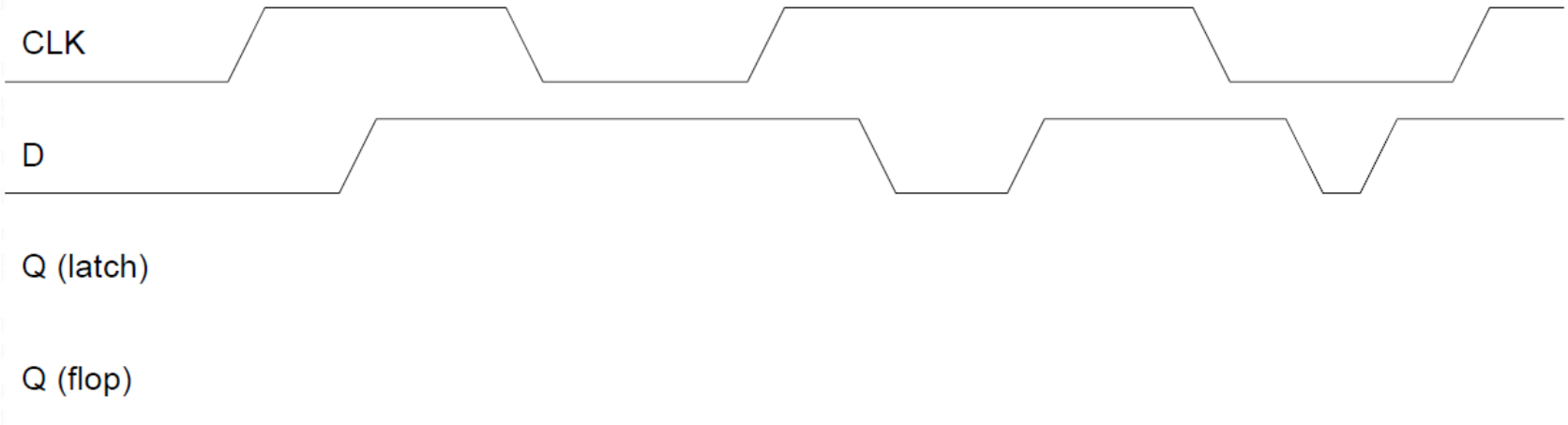
- Когато ***CLK* = 1**
  - L2 е прозрачен (transparent)
  - L1 е непрозрачен (opaque)
  - N1 се прехвърля към *Q*

- Следователно, по фронта на clock (когато ***CLK* нараства от 0 към 1(0→1)**)
  - *D* се прехвърля към *Q*

КАРХ: Тема_5: Последователни логически схеми

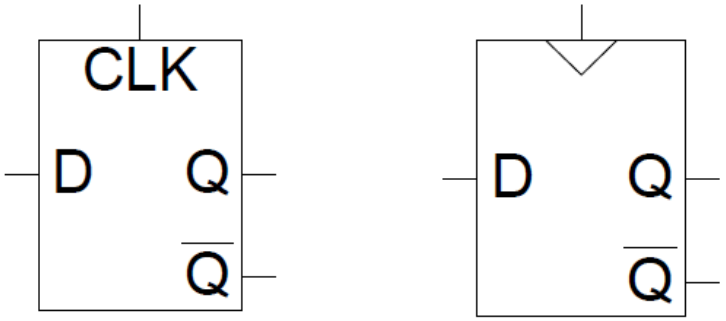
Сравнение на двата типа D Тригери. (D Latch vs. D Flip-Flop)

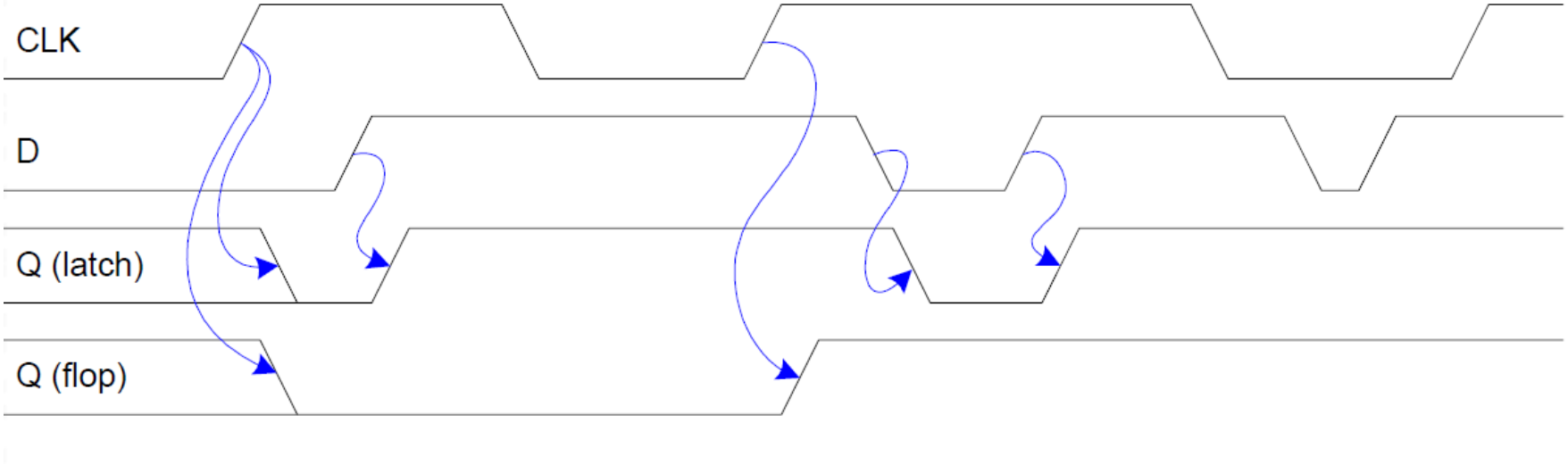




КАРХ: Тема_5: Последователни логически схеми

Сравнение на двата типа D Тригери. (D Latch vs. D Flip-Flop)

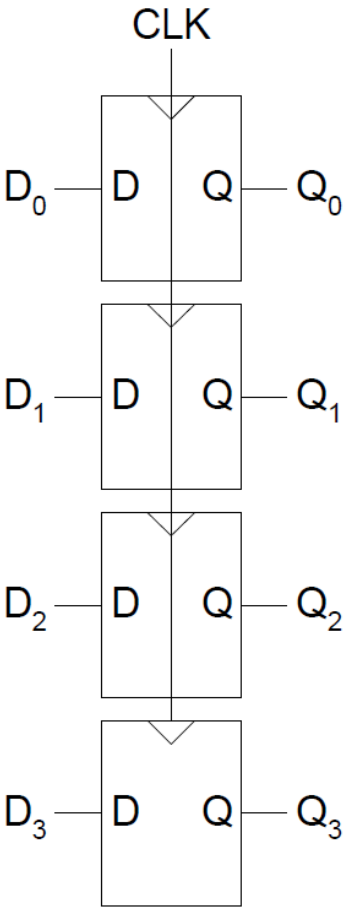


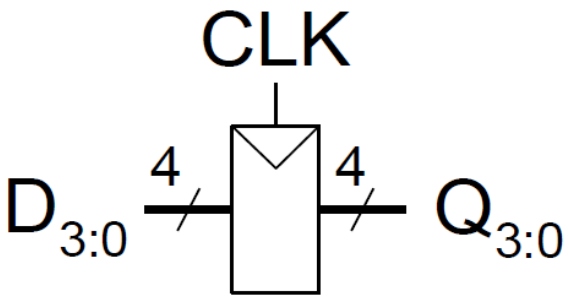


КАРХ: Тема_5: Последователни логически схеми

Регистър (регистър-памет).

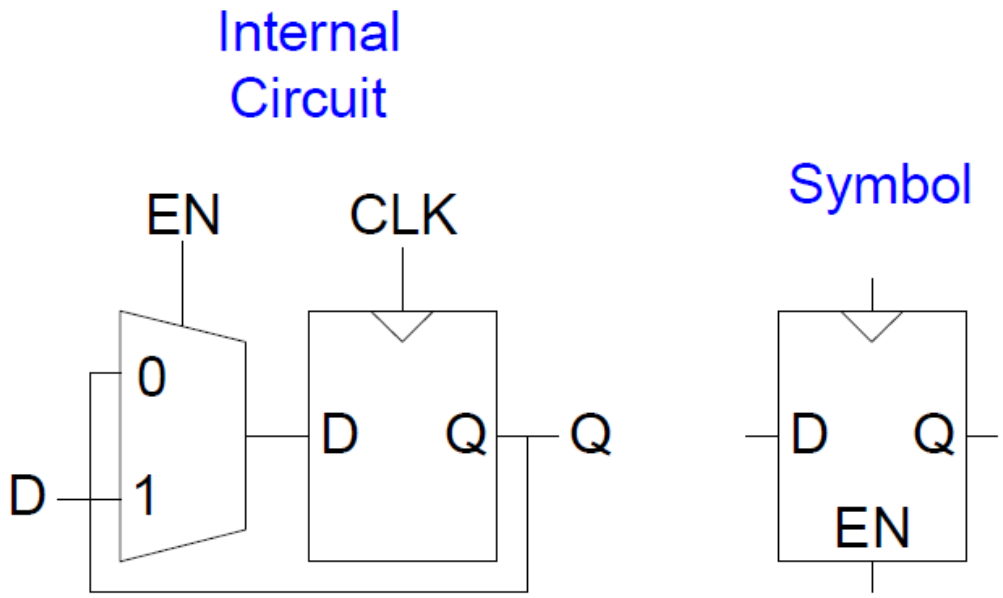
- К-на брой MS D Тригера с общ *CLK*.
- **Пример:** 4 bit регистър.





MS (Master – Slave) D Тригер с разрешение. (Enabled D Flip-Flop)

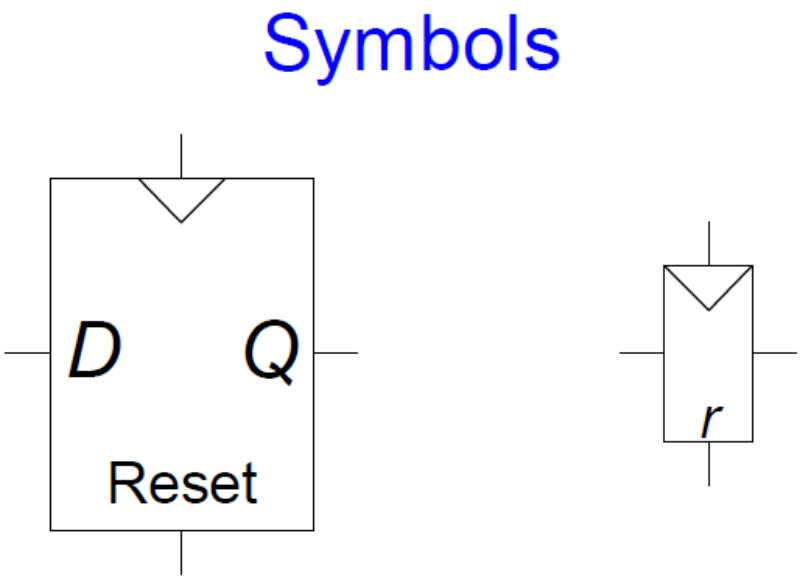
- **Входове:** *CLK*, *D*, *EN*
  - Разрешаващият вход (*EN*) контролира записа на нови данни (*D*).
- **Функция**
  - ***EN* = 1:** Когато *CLK* нараства от 0 към 1, стойността на *D* се прехвърля на *Q*;
  - ***EN* = 0:** *Q* запазва предишната си стойност.



КАРХ: Тема_5: Последователни логически схеми

MS (Master – Slave) D Тригер с нулиране. (Resettable D Flip-Flop)

- **Входове:** *CLK*, *D*, *Reset*
  - Нулиращият вход (*Reset*) записва 0 на изхода ($Q = 0$).
- **Функция**
  - ***Reset* = 1:** нулира Q ($Q = 0$).
  - ***Reset* = 0:** тригерът се държи като обикновен D flip-flop.



КАРХ: Тема_5: Последователни логически схеми

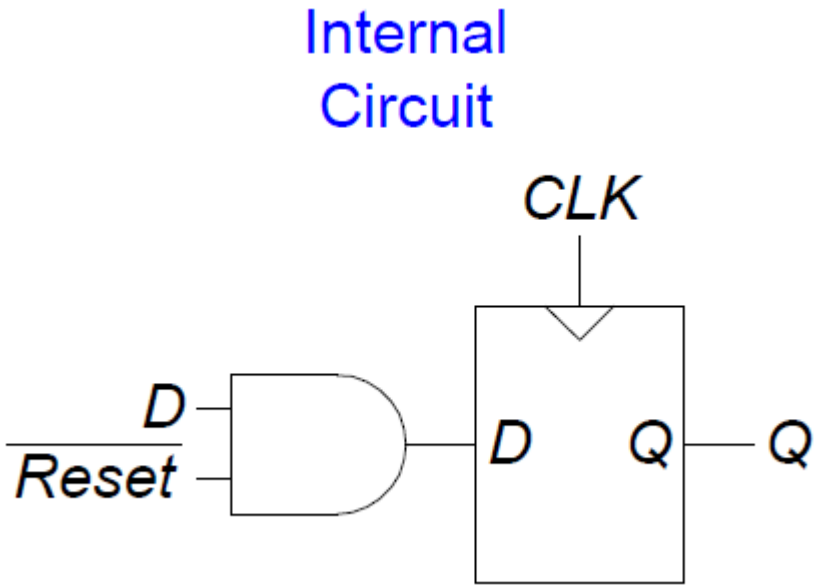
MS (Master – Slave) D Тригер с нулиране. (Resettable D Flip-Flop)

- **Два вида:**
  - **Синхронни:** нулирането се извършва само по фронта на *CLK*
  - **Асинхронни:** нулирането се извършва веднага щом *Reset* = 1
- Асинхронните resettable flip-flop изискват промяна на вътрешната схема на тригера.
- А синхронните resettable flip-flop?

КАРХ: Тема_5: Последователни логически схеми

MS (Master – Slave) D Тригер с нулиране. (Resettable D Flip-Flop)

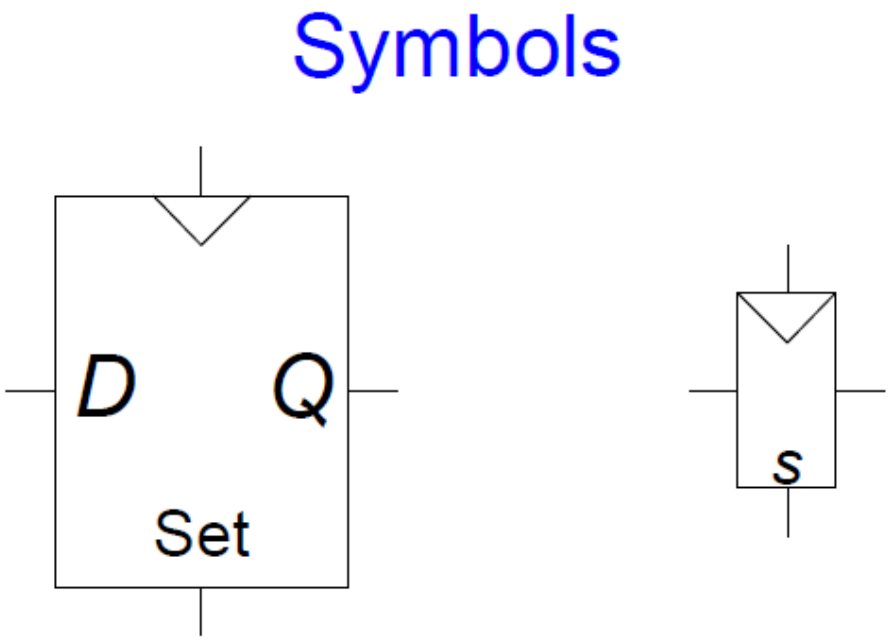
- **Два вида:**
  - **Синхронни:** нулирането се извършва само по фронта на *CLK*
  - **Асинхронни:** нулирането се извършва веднага щом *Reset* = 1
- Асинхронните resettable flip-flop изискват промяна на вътрешната схема на тригера.
- А синхронните resettable flip-flop?



КАРХ: Тема_5: Последователни логически схеми

MS (Master – Slave) D Тригер с установяване. (Settable D Flip-Flop)

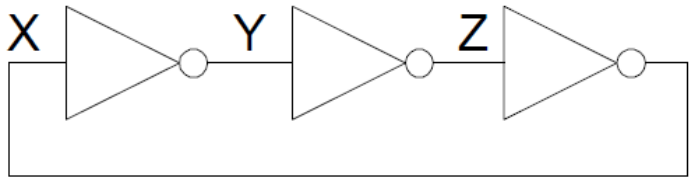
- **Входове:** *CLK*, *D*, *Set*
  - Установяващият вход (*Set*) записва 1 на изхода (*Q* = 1).
- **Функция**
  - ***Set* = 1 :** установява *Q* (*Q* = 1).
  - ***Set* = 0 :** тригерът се държи като обикновен D flip-flop.

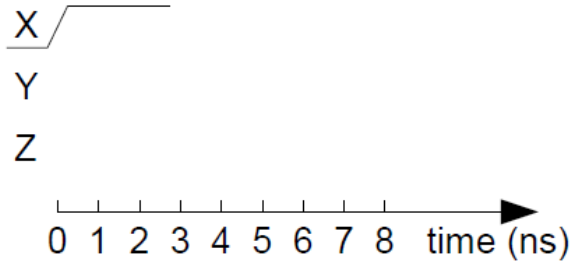


КАРХ: Тема_5: Последователни логически схеми

А-стабилни схеми. (Astable circuits)

- Последователни логически схеми – всички логически схеми, които не са комбинационни.
- Проблематични схеми:

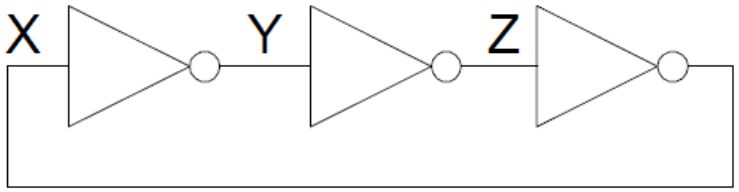


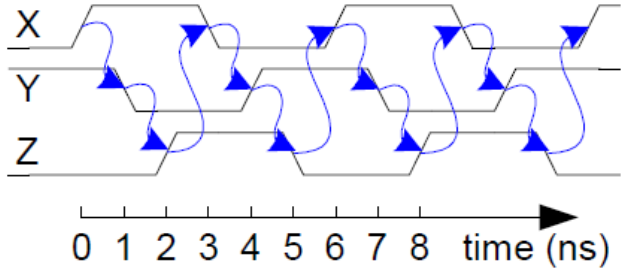


КАРХ: Тема_5: Последователни логически схеми

А-стабилни схеми. (Astable circuits)

- Последователни логически схеми – всички логически схеми, които не са комбинационни.
- Проблематични схеми:





- Няма входове, има 1-3 изхода.
- А-стабилна схема (Astable circuit), осцилира (генерира).
- Периодът на осцилациите зависи от закъснението на инверторите.
- Има циклична (обратна) връзка (***cyclic path***): изходът е свързан с входа.

КАРХ: Тема_5: Последователни логически схеми

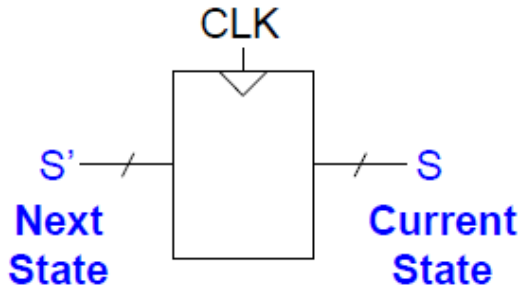
Дизайн на синхронна последователна логика.

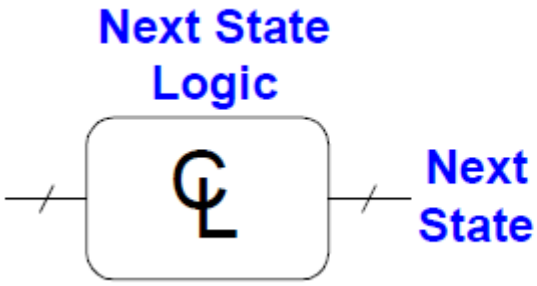
- Разкъсват се цикличните (кръгови) пътища чрез поставяне на регистри.
- Регистрите съдържат **състоянието** на системата.
- Състоянието се променя по фронта (edge) на тактовия сигнал (clock), т.е. ситемата е **синхронизирана** с clock.
- **Правила** при изграждането на синхронни последователни схеми:
  - Всеки елемент от схемата е или регистър или комбинационна схема;
  - Поне един елемент от схемата е регистър;
  - Всички регистри получават еднакъв тактов сигнал (clock);
  - Всеки цикличен път съдържа поне един регистър.
- Има два вида синхронни последователни схеми:
  - Крайни автомати (Finite State Machines) (**FSMs**);
  - Конвеери (Pipelines).

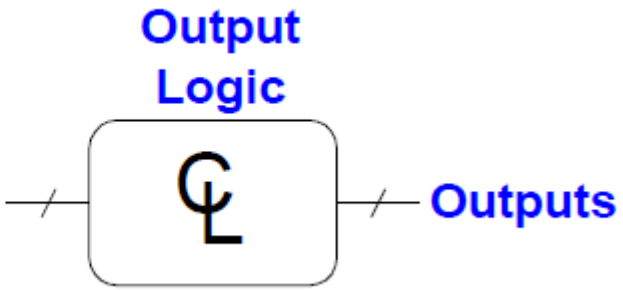
КАРХ: Тема_5: Последователни логически схеми

Дизайн на крайни автомати (Finite State Machines) .

- Състоят се от :
  - **Регистър на състоянието (State register)**
    - Съхранява текущото състояние.
    - Зарежда се със следващото сътояние по фронта на тактовия сигнал (clock edge)
  - **Комбинационна логика**
    - Определя (Задава) следващото състояние
    - Определя (Задава) състоянието на изходите



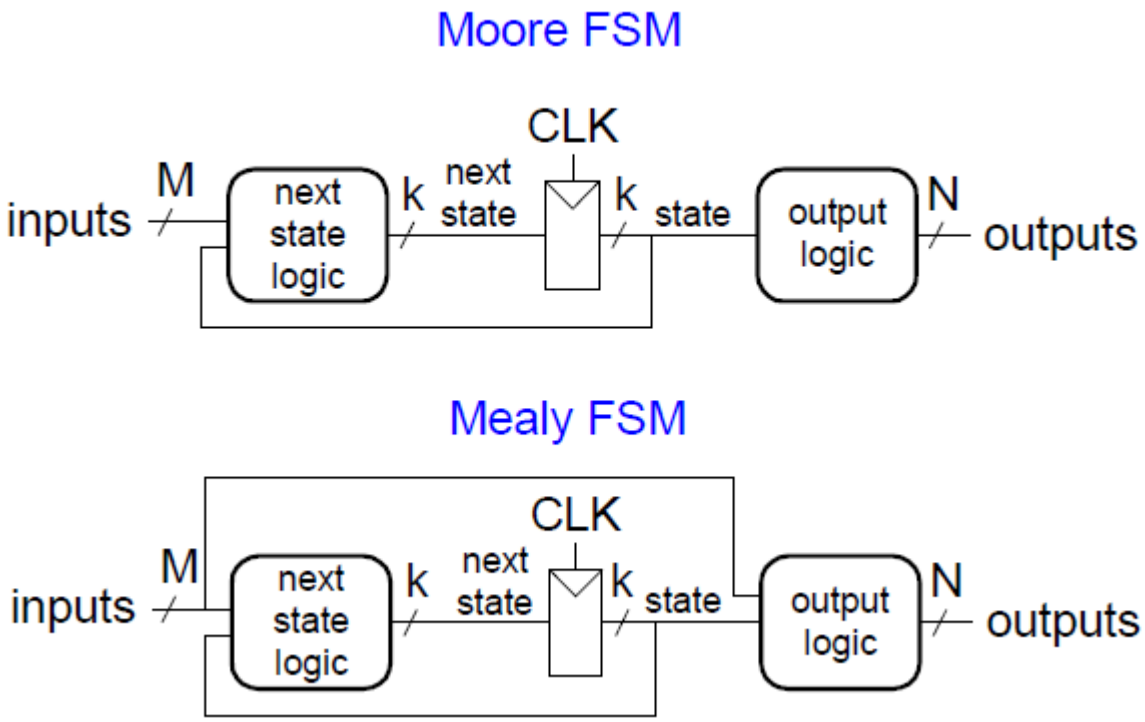




КАРХ: Тема_5: Последователни логически схеми

Дизайн на крайни автомати (Finite State Machines) .

- Следващото състояние се определя от текущото състояние и входовете.
- Има два типа крайни автомати, които се различават в изходната логика:
  - **Moore FSM:** изходите зависят само от текущото състояние;
  - **Mealy FSM:** изходите зависят от текущото състояние **и** входовете.



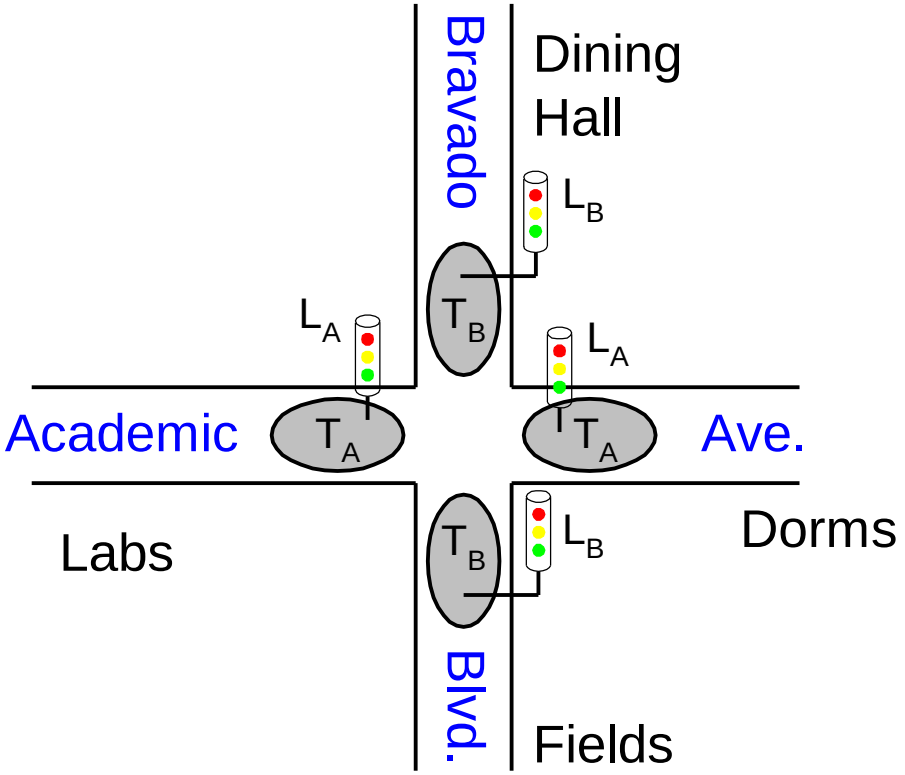
КАРХ: Тема_5: Последователни логически схеми

Крайни автомати (Finite State Machines) .

- Пример :

Светофарен контролер (Traffic light controller).

- Traffic sensors: T_A, T_B (TRUE when there's traffic)
- Lights: L_A, L_B



КАРХ: Тема_5: Последователни логически схеми

Крайни автомати (Finite State Machines) .

Кодиране на състоянието :

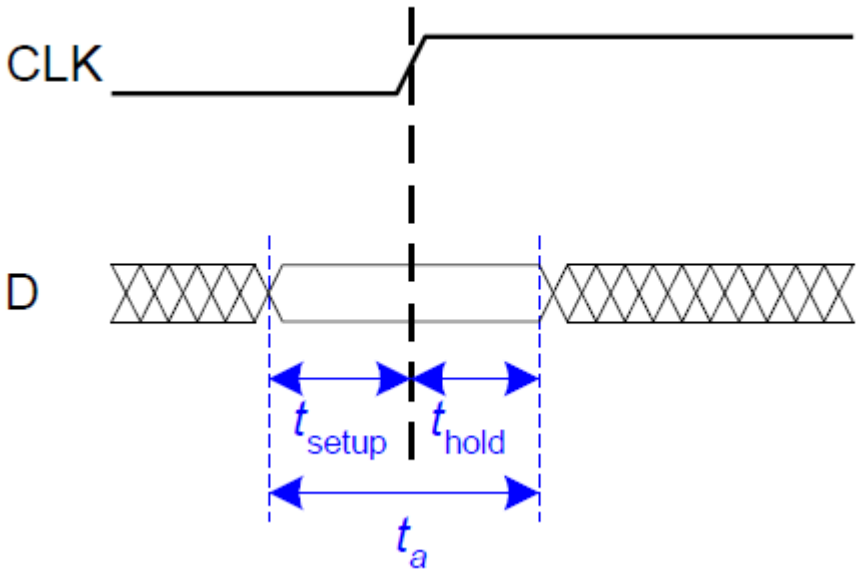
- **Двоично** кодиране:
  - т.е., за четири състояния: 00, 01, 10, 11
- **С активен бит (One-hot)** кодиране
  - По един бит за всяко състояние
  - Само битът на активното състояние е HIGH
  - т.е., за четири състояния: 0001, 0010, 0100, 1000
  - Това изисква използването на повече тригери
  - Води често до опростяване на останалата логика

КАРХ: Тема_5: Последователни логически схеми

Времеви спецификации (Timing).

Вход

- Данните (*D*) се прехвърлят в тригерите по **фронта на clock** (**clock edge**).
- Данните (*D*) трябва да са **стабилни** по време на прехвърлянето.
- Данните (*D*) трябва да са **стабилни и около** фронта на clock (clock edge).
- **Setup time:** t_{setup} = time *before* clock edge data must be stable (i.e. not changing)
- **Hold time:** t_{hold} = time *after* clock edge data must be stable
- **Aperture time:** t_a = time *around* clock edge data must be stable ($t_a = t_{setup} + t_{hold}$)

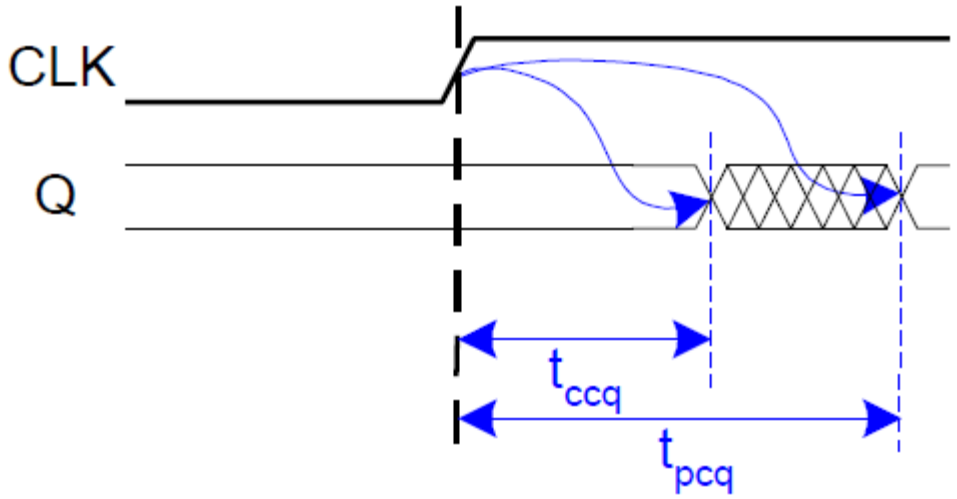


КАРХ: Тема_5: Последователни логически схеми

Времеви спецификации (Timing).

Изход

- **Propagation delay:** t_{pcq} = time after clock edge that the output *Q* is guaranteed to be stable (i.e., to stop changing)
- **Contamination delay:** t_{ccq} = time after clock edge that *Q* might be unstable (i.e., start changing)



КАРХ: Тема_5: Последователни логически схеми

Времеви спецификации (Timing).

Изводи (Dynamic Discipline).

- Входните сигнали при синхронна последователна логика трябва да бъдат стабилни по времето на апертурата (setup and hold) на фронта (clock edge).
- Специално входовете трябва да са стабилни
  - най-малко t_{setup} преди фронта (clock edge)
  - най-малко до t_{hold} след фронта (clock edge)

КАРХ: Тема_5: Последователни логически схеми

Паралелизъм (Parallelism).

- **Има два типа паралелизъм:**
  - **Пространствен (Spatial parallelism)**
    - мултиплициран хардуер – много задачи едновременно.
  - **Времеви (Temporal parallelism)**
    - Задачата се разделя на много стъпки (стадии).
    - Наричан още конвееризация (pipelining).

Дефиниции.

- **Token:** Група от входни сигнали водещи до поява на група от изходни сигнали на дадена система.
- **Latency:** Времето за което един token преминава през системата.
- **Throughput (Поизводителност):** Броят на tokens преминали през системата за единица време.
- **Паралелизмът повишава производителността!**

КАРХ: Тема_5: Последователни логически схеми

Езици за хардуерно описание (Hardware Description Lenguages (HDL)).

През 90-те години на миналия век компютрите започват да се използват и за проектиране в различни области на живота – архитектура и строителство, машиностроене и електроника и др., чрез създаването на специализиран софтуер за проектиране – **CAD (Computer-aided Design).**

- Спецификациите в електрониката се дават чрез HDL, като двата основно използвани езика са System Verilog и VHDL. Те имат общи принципи, но различен синтаксис.
  - Описание на хардуера – чрез електронни схеми и посредством HDL.
  - Хардуерни модули – блок от хардуерни елементи с входове и изходи, които се описват функционално и структурно.
  - Основни цели на HDL – **симулация** и **синтез**.
  - Симулация – избягване (изчистване) на „бъгове“ в системите.
  - Синтез – HDL код → netlist (текстови файл) или електронна схема.

HDL не са езици за програмиране!