[P220]

Увод у архитектуру рачунара 4

Саша Малков
Универзитет у Београду
Математички факултет
2013/2014

[P271]

Увод у архитектуру рачунара

Саша Малков

Тема 5

Секвенцијалне мреже

(наставак)

Бројачи

- Бројачи су секвенцијалне мреже које са сваким циклусом повећавају вредност регистра за 1
- Веома често се употребљавају
- Бинарни бројач користи за бројање регистар са B битова и омогућава бројање од 0 до 2^B-1
  - Након "прекорачења" се почиње поново од нуле
  - Назива се и бројач "по модулу 2^B"
- Бројач по модулу 10 се употребљава за рад са декадним цифрама

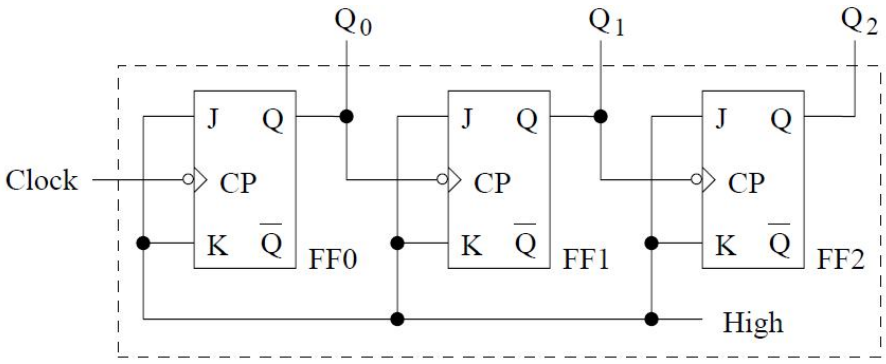
Имплементација бројача

- Имплементирамо бројач по модулу 8
- Низ битова се посматра као низ бинарних цифара
  - Најнижи бит се мења у сваком циклусу
  - Виши бит је потребно променити сваки пут када се претходни нижи бит промени из 1 у 0

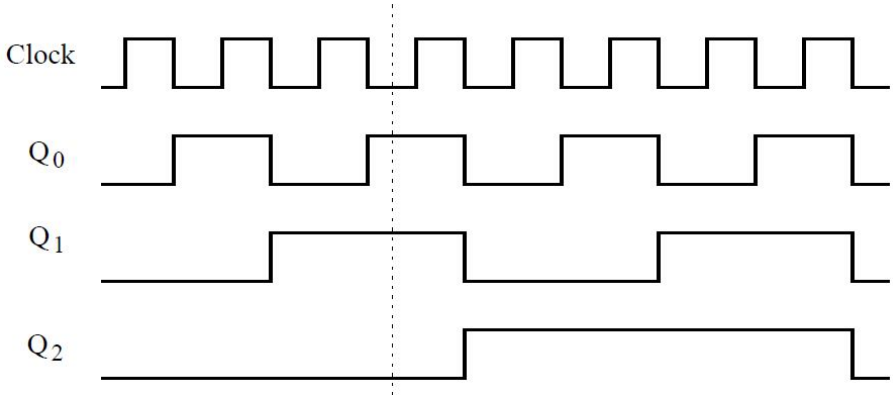
Имплементација бројача (2)

- Подсећање:
  - Ако се улази на *JK* флип-флопу поставе на 1, онда ће се излаз мењати у сваком циклусу
- Идеја имплементације је:
  - на низ *JK* флип-флопова се доведу на улазе активни сигнали
  - на контролни улаз првог (који одговара најнижем биту бројача) се доведе сигнал часовника
  - на контролне улазе осталих се доведу излазни сигнали претходних

Имплементација бројача (3)



Понашање бројача



Неке примене бројача

- Бројачи се могу употребљавати
  - за мерење времена
  - за бројање података на улазу или излазу
  - за генерисање часовника са споријим тактом
  - и друго

Таласасти бројачи

- Код таласастих бројача се бројање таласасто преноси од најнижег према највишем биту
- То има за последицу релативно велико трајање пропагације
  - које се повећава са дужином бројача
    - (слично као у случају сабирача)
  - ако је трајање пропагације за један *JK* флип-флоп око 20ns, онда је за 16-битни бројач време пропагације око 320ns

Универзитет у Београду - Математички факултет

Синхрони бројачи

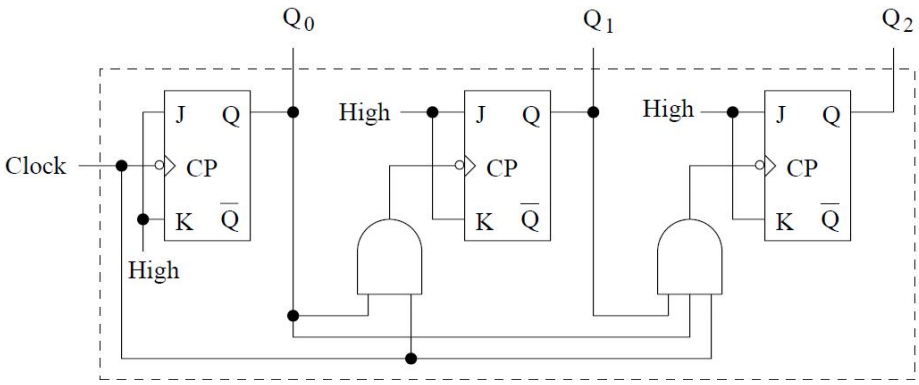
- Проблем се решава на сличан начин као у случају сабирача, пропагирањем унапред
- Користимо чињеницу да *JK* флип-флоп мења стање на силазном рубу контролног сигнала
- Како се сва стања мењају на силазном рубу, све до тог тренутка се очитава претходно стање
- На флип-флоп елемент (осим првог) се доводи активан контролни сигнал ако су нова стања за све претходне елементе неактивна
  - тј. ако су претходна стања за све претходне елементе активна

Универзитет у Београду - Математички факултет

Синхрони бројачи (2)

- Контролни сигнал се изражава као конјункција свих претходних излаза и часовника
  - силазни руб контролног сигнала настаје на силазном рубу часовника и то само ако су сви претходни елементи били активни
- Код синхроног бројача сви флип-флопови раде истовремено
  - укупно трајање пропагације је исто као за један флип-флоп
  - плус, трајање пропагације једне конјункције

Имплементација синхроног бројача по модулу 8



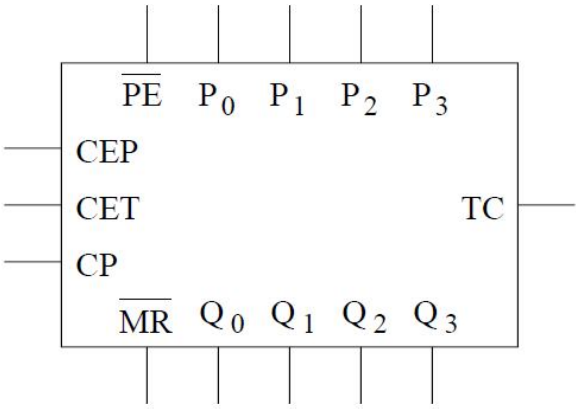
Индустријски бројачи

- Производе се и готови бројачи, који се могу прилагођавати потребама
- Бројач 74161
  - бинарни бројач по модулу 16 (4 бита)
  - подржава брзине до 120MHz
- Бројач 74160
  - декадни бројач по модулу 10

Индустријски бројачи

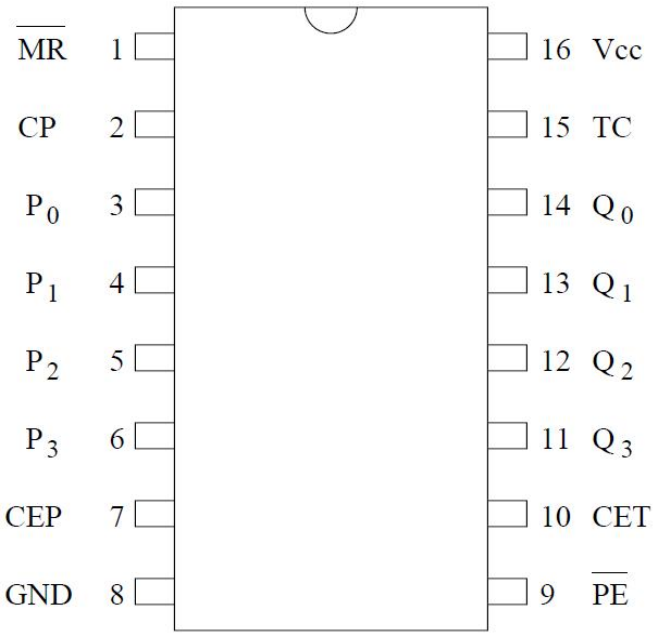
- Повезују се на исти начин
  - анулирајући улаз (*MR' - master reset*)
    - анулира бројач ако се доведе низак ниво сигнала
  - улаз за постављање почетне вредности (*PE' - preset enable*)
    - ако се доведе низак ниво сигнала (при високом нивоу на MR'), поставља бројач на вредност улаза $P_0P_1P_2P_3$
  - терминални излаз (*TC*) је активан ако је бројач у завршном стању (после кога следи 0)
  - улаз представљају *CET* и *CEP*
    - ако су оба активна, бројач ради
    - ако је бар један неактиван, бројач само чува стање

Логички симбол бројача 74161 и 74160



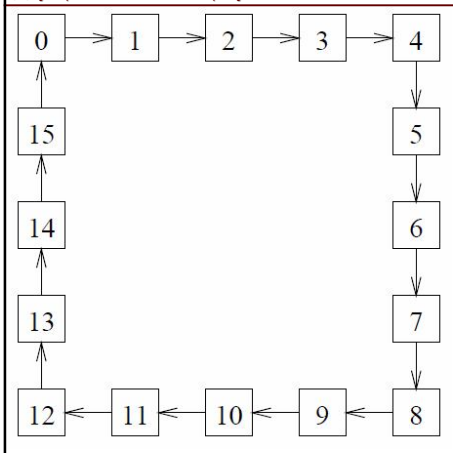
Универзитет у Београду - Математички факултет

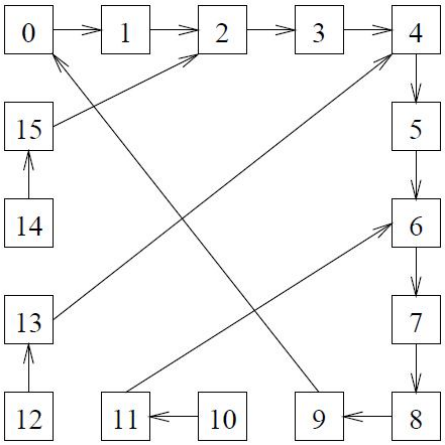
Дијаграм повезивања бројача 74161 и 74160



Универзитет у Београду - Математички факултет

Дијаграми стања бројача 74161 и 74160





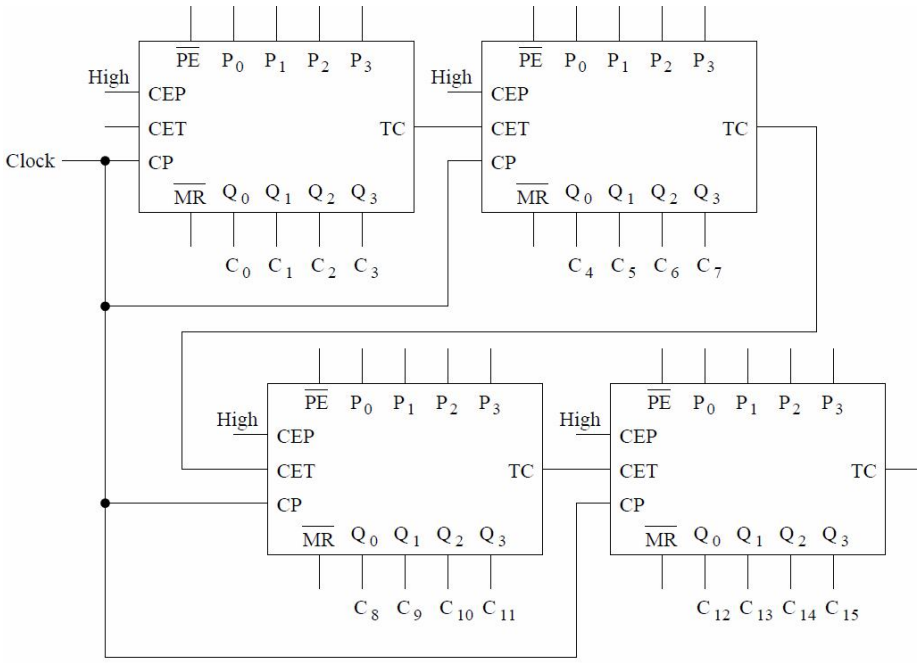
Универзитет у Београду - Математички факултет

Вишестепени бројач

- Помоћу више бројачких кола могу се имплементирати тзв. *вишестепени бројачи*
- За *пренос* се употребљава терминални излаз *TC*
  - употребљава се на исти начин као при везивању више флип-флопова у једном бројачу

Универзитет у Београду - Математички факултет

Вишестепени 16-битни бројач (по модулу 65536)



Пројектовање секв. мрежа

- Ради илустрације поступка ограничићемо се на употребу *JK* флип-флопова
- Поступак је аналоган и за друге врсте елемената
- Пример је синхрони бројач по модулу 8

Анализа понашања

- Анализирамо потребно понашање
  - знамо излазно понашање елемената
  - потребно је да пронађемо одговарајућу комбинацију улаза
- За сваки елемент који употребљавамо морамо знати таблицу прелазака елемента
  - *Таблица прелазака елемента* описује којим улазима се остварују које промене стања елемента
  - Небитне улазе означавамо са *n* или *d*

Понашање *JK* флип-флопа

- На основу истинитосне таблице елемента правимо таблицу ексцитације (прелазака) елемента

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Q_n	Q_{n+1}	J	K
0	0	?	
0	1		
1	0		
1	1		

Таблица ексцитације кола

- *Таблица ексцитације кола* описује који скуп улаза за све флип-флопове одговара сложеним преласцима стања читавог кола
  - прву групу колона чине тренутна стања свих битова
  - другу групу колона чине наредна стања свих битова
  - трећа група се састоји од потребних улаза за све обухваћене елементе
- Прве две групе колона се попуне на основу циљног понашања секвенцијалне мреже
- Последња група се попуњава на основу прве две и таблица ексцитација одговарајућих елемената

Универзитет у Београду - Математички факултет

Таблица ексцитације бројача по модулу 8

Present state			Next state			JK flip-flop inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	d	0	d	1	d
0	0	1	0	1	0	0	d	1	d	d	1
0	1	0	0	1	1	0	d	d	0	1	d
0	1	1	1	0	0	1	d	d	1	d	1
1	0	0	1	0	1	d	0	0	d	1	d
1	0	1	1	1	0	d	0	1	d	d	1
1	1	0	1	1	1	d	0	d	0	1	d
1	1	1	0	0	0	d	1	d	1	d	1

Универзитет у Београду - Математички факултет

Пресликавање стања у улазе

- Следећи корак је пресликавање стања у улазе
  - Сваки од потребних улаза изражавамо као функцију која као аргументе има постојећа стања
  - Ту функцију минимизујемо (на пример Карноовим мапама)

Универзитет у Београду - Математички факултет

Улази елемента A

A \ BC	00	01	11	10
0	0	0	1	0
1	d	d	d	d

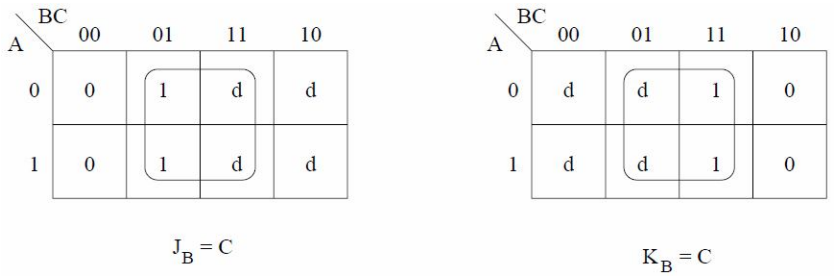
$J_A = B\,C$

A \ BC	00	01	11	10
0	d	d	d	d
1	0	0	1	0

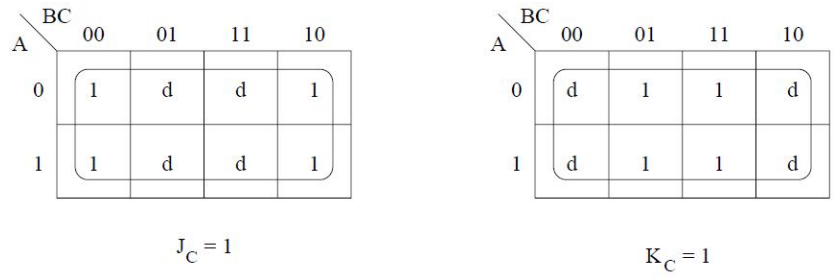
$K_A = B\,C$

Универзитет у Београду - Математички факултет

Улази елемента B



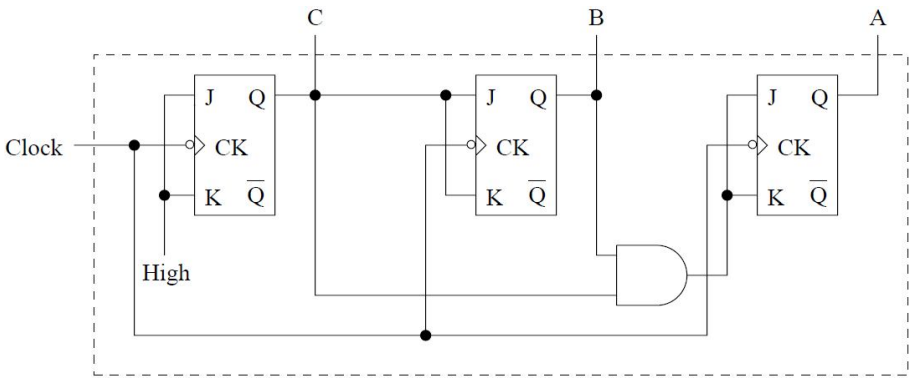
Улази елемента C



Цртање логичког дијаграма

- На крају преостаје још само да нацртамо дијаграм логичког кола који на одговарајући начин пресликава стања у улазе

Дијаграм логичког кола бинарног бројача по модулу 8



За вежбу

- Задатак:
  - Направити логичко коло апстрактног бројача чија се стања мењају у редоследу: 0→3 →5 →7 →6 →0

Уопштен алгоритам пројектовања секвенцијалних мрежа

- Уопштен алгоритам почива на представљању стања и промена стања помоћу коначног аутомата (*finite state machine*)
  - корак 1 – извођење коначног аутомата
  - корак 2 – додељивање стања коначног аутомата флип-флоповима
  - корак 3 – прављење таблице ексцитације кола
  - корак 4 – извођење функција улаза флип-флопова
  - корак 5 – имплементација

Представљање коначног аутомата

- Може се представљати графички
- Често је употребљивије представљање у облику таблице
  - колона 1 – стање
  - група колона 2 – ново стање (за сваки улаз по колона)
  - група колона 3 – излаз (за сваки улаз по колона)

Смернице за корак 2

- У општем случају, број флип-флопова се рачуна као $ceil(log_2 N)$, где је N број стања
- Да би се добио оптималан дизајн кола, користе се три смернице
  - правило 1 – ако два стања имају исто наредно стање за исти улаз, онда би требало да буду суседна
  - правило 2 – сва наредна стања за неко стање би требало да буду међусобно суседна
  - правило 3 – стања која имају исти излаз за дати улаз би требало да буду суседна
    - (“суседна” у смислу Хамингове удаљености)
- Затим се стања означе битовима тако да што више правила буде задовољено

Пример – Задатак

- Направити секвенцијално коло које има излаз 1 ако и само ако у последња три прочитана бита постоје тачно два бита 0

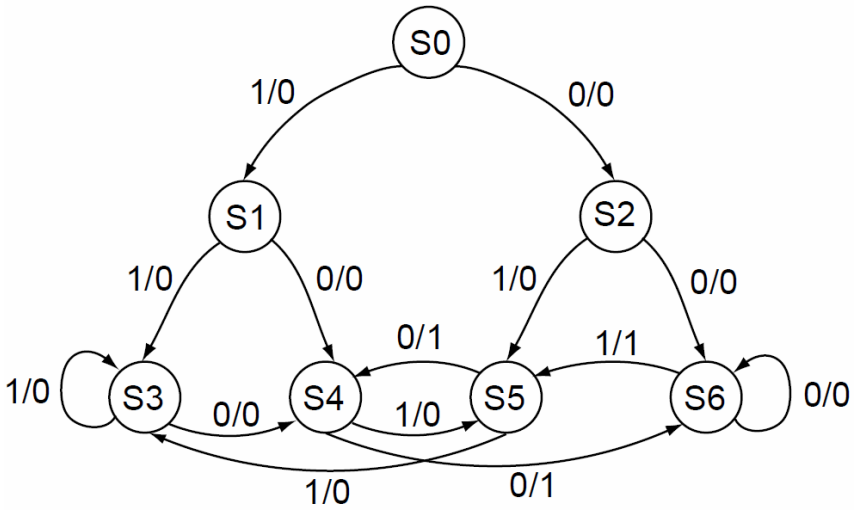
Пример – Стања

- На основу претходна два прочитана бита и једног који је на улазу може се увек одредити излаз
- Због тога је довољно стањима описати различите случајеве за до два последња прочитана бита
  - S0 – није прочитан ниједан бит
  - S1 – прочитан је тачно један бит, и то је 1
  - S2 – прочитан је тачно један бит, и то је 0
  - S3 – последња два прочитана бита су 11
  - S4 – последња два прочитана бита су 10
  - S5 – последња два прочитана бита су 01
  - S6 – последња два прочитана бита су 00

Пример – Коначни аутомат

- Преласке између стања означавамо са X/Y
  - X означава бит на улазу
  - Y означава бит на излазу
- Претпостављамо да читамо сдесна улево

Пример – Коначни аутомат



Пример – Коначни аутомат

Present state	Next state		Output	
	X = 0	X = 1	X = 0	X = 1
S0	S2	S1	0	0
S1	S4	S3	0	0
S2	S6	S5	0	0
S3	S4	S3	0	0
S4	S6	S5	1	0
S5	S4	S3	1	0
S6	S6	S5	0	1

Универзитет у Београду - Математички факултет

Пример – Флип-флопови (1)

- правило 1 – ако два стања имају исто наредно стање за исти улаз, онда би требало да буду суседна
- Примена правила 1 сугерише да би требало да буду суседна стања
  - S1, S3 и S5 (прелазе у S4 ако је улаз X=0)
  - S1, S3 и S5 (прелазе у S3 ако је улаз X=1)
  - S2, S4 и S6 (прелазе у S6 ако је улаз X=0)
  - S2, S4 и S6 (прелазе у S5 ако је улаз X=1)

Универзитет у Београду - Математички факултет

Пример – Флип-флопови (2)

- правило 2 – сва наредна стања за неко стање би требало да буду међусобно суседна
- Примена правила 2 сугерише да би требало да буду суседна стања
  - S1 и S2 (у њих се прелази из S0)
  - S3 и S4 (у њих се прелази из S1)
  - S5 и S6 (у њих се прелази из S2)
  - S3 и S4 (у њих се прелази из S3)
  - S5 и S6 (у њих се прелази из S4)
  - S3 и S4 (у њих се прелази из S5)
  - S5 и S6 (у њих се прелази из S6)

Пример – Флип-флопови (3)

- правило 3 – стања која имају исти излаз за дати улаз би требало да буду суседна
- Примена правила 3 сугерише да би требало да буду суседна стања
  - S0, S1, S2, S3 и S6 (излаз је 0 за улаз 0)
  - S4 и S5 (излаз је 1 за улаз 0)
  - S0, S1, S2, S3, S4 и S6 (излаз је 0 за улаз 1)
  - S5 (излаз је 0 за улаз 1)
- Због величина група, последње две се могу занемарити

Пример – Флип-флопови (4)

- Прикупимо све групе
  - у “експоненту” означимо колико правила сугерише групу
- $(S1, S3, S5)^2$
- $(S2, S4, S6)^2$
- $(S1, S2)$
- $(S3, S4)^3$
- $(S5, S6)^3$
- $(S4, S5)$
- $(S0, S1, S2, S3, S6)$

Универзитет у Београду - Математички факултет

Пример – Флип-флопови (5)

- Код 000 доделимо стању S0 (произвољно)
- Остале кодове додељујемо тако да задовољимо што више правила групаж
- Обично постоји више једнако добрих решења

Универзитет у Београду - Математички факултет

Пример – Флип-флопови (6)

A \ BC	00	01	11	10
0	S0	S3	S5	S1
1		S4	S6	S2

State		A	B	C
S0	=	0	0	0
S1	=	0	1	0
S2	=	1	1	0
S3	=	0	0	1
S4	=	1	0	1
S5	=	0	1	1
S6	=	1	1	1

Пример – Флип-флопови (7)

- Изаберу се флип-флопови
  - У овом случају се бирају 3 *JK* флип-флопа
- Направи се одговарајућа таблица ексцитације

Пример – Таблица експцитације

A	B	C	X	A	B	C	Y	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	1	1	0	0	1	d	1	d	0	d
0	0	0	1	0	1	0	0	0	d	1	d	0	d
0	0	1	0	1	0	1	0	1	d	0	d	d	0
0	0	1	1	0	0	1	0	0	d	0	d	d	0
0	1	0	0	1	0	1	0	1	d	d	1	1	d
0	1	0	1	0	0	1	0	0	d	d	1	1	d
0	1	1	0	1	0	1	1	1	d	d	1	d	0
0	1	1	1	0	0	1	0	0	d	d	1	d	0
1	0	1	0	1	1	1	1	d	0	1	d	d	0
1	0	1	1	0	1	1	0	d	1	1	d	d	0
1	1	0	0	1	1	1	0	d	0	d	0	1	d
1	1	0	1	0	1	1	0	d	1	d	0	1	d
1	1	1	0	1	1	1	0	d	0	d	0	d	0
1	1	1	1	0	1	1	1	d	1	d	0	d	0

Пример – Минимизација

AB \ CX	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	d	d	d	d
10	d	d	d	d

$J_A = \overline{X}$

AB \ CX	00	01	11	10
00	d	d	d	d
01	d	d	d	d
11	0	1	1	0
10	d	d	1	0

$K_A = X$

Пример – Минимизација

AB \ CX	00	01	11	10
00	1	1	0	0
01	d	d	d	d
11	d	d	d	d
10	d	d	1	1

$J_B = \overline{C} + A$

AB \ CX	00	01	11	10
00	d	d	d	d
01	1	1	1	1
11	0	0	0	0
10	d	d	d	d

$K_B = \overline{A}$

Универзитет у Београду - Математички факултет

Пример – Минимизација

AB \ CX	00	01	11	10
00	0	0	d	d
01	1	1	d	d
11	1	1	d	d
10	d	d	d	d

$J_C = B$

AB \ CX	00	01	11	10
00	d	d	0	0
01	d	d	0	0
11	d	d	0	0
10	d	d	0	0

$K_C = 0$

Универзитет у Београду - Математички факултет

Пример – Излазна функција

- Излазна функција се опише у зависности од стања и улаза
- Минимизује се
- У овом случају примењујемо Карноове мапе

Универзитет у Београду - Математички факултет

Пример – Излазна функција

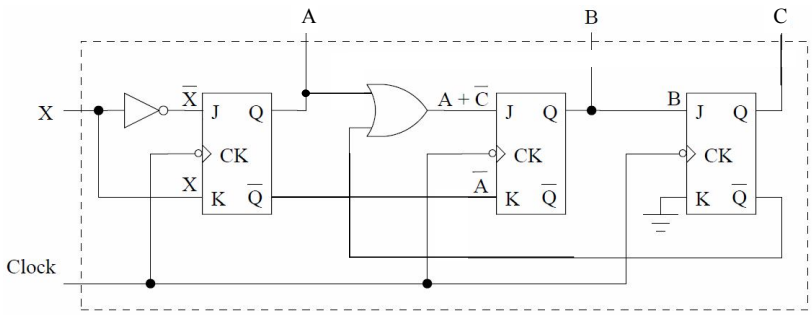
AB \ CX	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	0	0	1	0
10	d	d	0	1

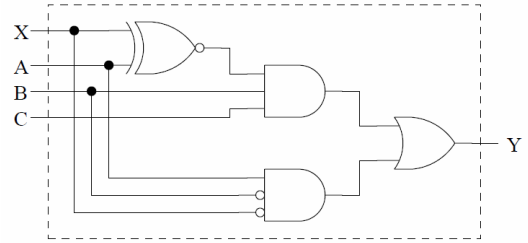
$Y = \bar{A}\,B\,C\,\bar{X} + A\,B\,C\,X + A\,\bar{B}\,\bar{X}$

- Може да се запише и као
  - Y = BC(A'X+AX')' + AB'X'

Универзитет у Београду - Математички факултет

Пример – Секвенцијална мрежа





Универзитет у Београду - Математички факултет

[P271]

Увод у архитектуру рачунара

Саша Малков

Тема 6

Елементи архитектуре

Појам архитектуре рачунара

- Архитектура рачунара
  - *Опис рачунарског система на логичком нивоу, из угла "програмера" (тј. техничког корисника)*
  - Обухвата различите аспекте који се тичу "програмера"
    - Основне функционалне јединице
      - процесор
      - меморија
      - ...
    - њихово функционисање
      - нпр. скуп инструкција, начин адресирања меморије,...
    - начини повезивања

Појам организације рачунара

- Организација рачунара
  - Односи се на начин имплементације и повезивања хардверских компоненти у циљу остваривања архитектуром одређених својстава

Појам пројектовања рачунара

- Пројектовање рачунара
  - Поступак који преводи архитектурне одреднице система у имплементацију применом конкретних организационих решења
  - Пројекат рачунара (*computer design*) се назива и имплементација рачунара

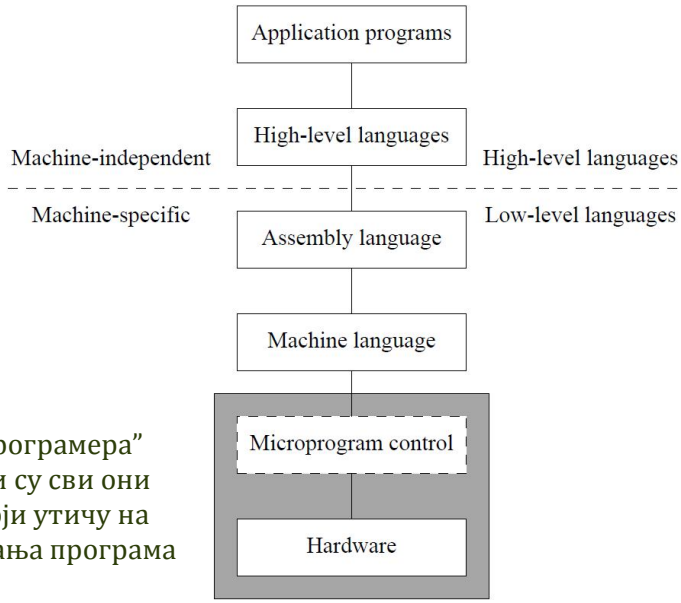
Универзитет у Београду - Математички факултет

Појам програмирања рачунара

- Програмирање рачунара
  - Изражавање конкретних проблема на језику који рачунар може да разуме
  - У контексту архитектуре и организације рачунара од интереса су само језици ниског нивоа, који се непосредно ослањају на конкретну архитектуру и организацију:
    - машински језик
    - асемблер

Универзитет у Београду - Математички факултет

Нивои програмирања рачунара



- Из угла "програмера" најважнији су сви они аспекти који утичу на начин посања програма

Универзитет у Београду - Математички факултет

Архитектура скупа инструкција

- Архитектура скупа инструкција (енгл. *Instruction Set Architecture – ISA*)
  - Апстракција архитектуре скупа инструкција
  - Пружа програмерима информације о рачунару неопходне за развој програма
  - Представља интерфејс између хардвера и најнижег нивоа софтвера

Универзитет у Београду - Математички факултет

Асемблерски језици

- Два основна мотива за употребу:
  - уштеда меморијског простора
  - временска ефикасност

Из угла архитектуре

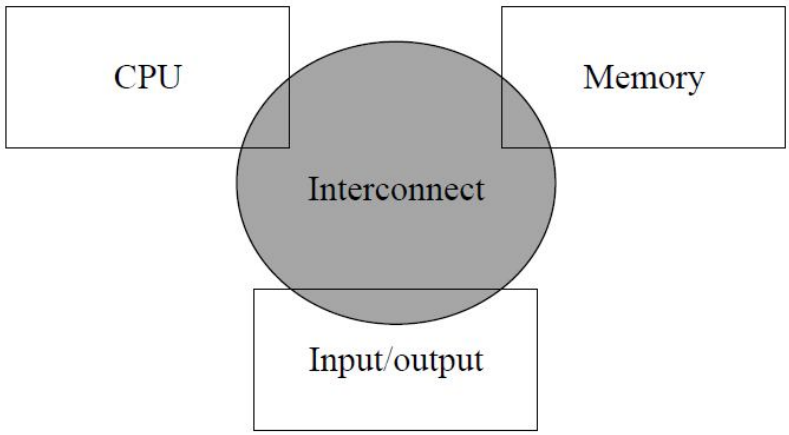
- Пројекат (имплементација) рачунара се посматра са вишег нивоа апстракције
  - Значајне су основне компоненте и њихови односи
  - Нису важни детаљни аспекти имплементације (нпр. којим логичким колом се имплементира нека функција АЛЈ)

Основне компоненте

- Из угла ахитектуре, основне компоненте рачунарског система су:
  - централна јединица за обраду (процесор, *CPU*)
  - меморијска јединица (меморија)
  - улазно/излазни уређаји (*I/O*)
  - њихово повезивање

Универзитет у Београду - Математички факултет

Основне компоненте



Универзитет у Београду - Математички факултет

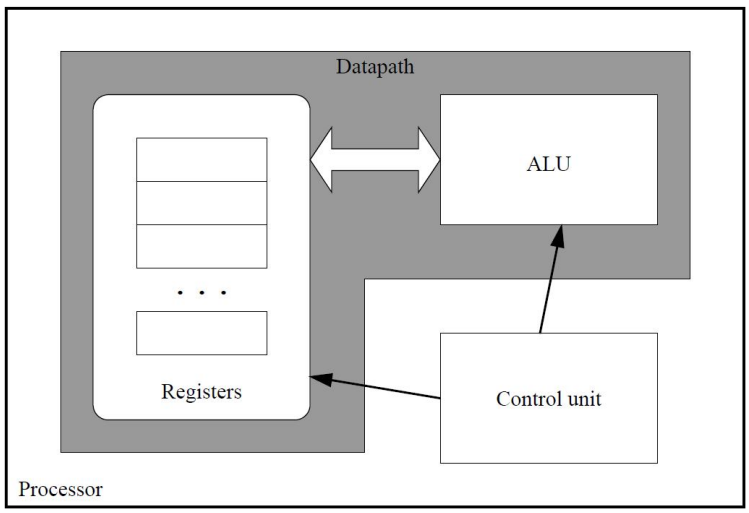
Из угла имплементатора

- Имплементатори раде на нивоу дигиталне логике
- Имплементирају логичким колима функције прописане архитектуром

Компоненте процесора

- Имплементатори виде три основне компоненте процесора:
  - контролна јединица
    - чита инструкције из главне меморије
    - декодира их и распознаје тип
    - управља радом процесора
  - регистри
    - локални меморијски простор процесора
    - начелно су сви исте величине
  - аритметичко логичка јединица (једна или више)
    - имплементација конкретних аритметичких и логичких операција

Компоненте процесора



Универзитет у Београду - Математички факултет

[P271]

Увод у архитектуру рачунара

Саша Малков

Тема 7

Магистрала

Магистрала

- Магистрала је подсистем који повезује компоненте рачунарског система
- Може да се састоји од компоненти, као што су:
  - адресна магистрала
  - магистрала података
  - контролна магистрала

Магистрала (2)

- Адресна магистрала преноси податке о меморијским адресама
  - Њена ширина одређује величину адресног простора
- Магистрала података преноси податке
  - Њена ширина одређује величину података који се преносе
- Контролна магистрала преноси контролне сигнале (кодиране операције)

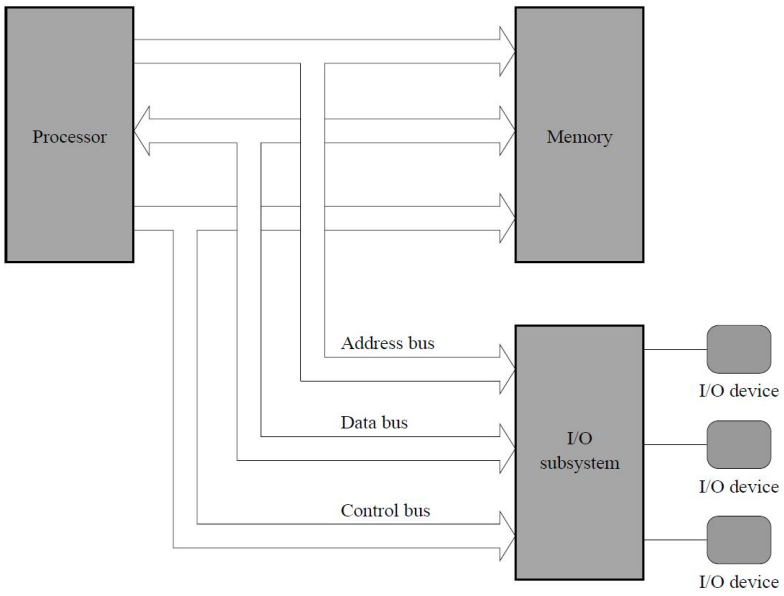
Системска магистрала

- Налази се унутар процесорског система
- Повезује процесорске јединице са меморијом и улазно/излазним подсистемом
- Употребљава се и термин *интерна (унутрашња) магистрала*

Системска магистрала

- Мрежа која повезује компоненте рачунарског система назива се *системска магистрала* (енгл. *system bus*)
- Исти термин се употребљава за означавање сигнала и проводника који проводе те сигнале
- Системска магистрала се састоји од три основне компоненте:
  - адресне магистрале
  - магистрале података и
  - контролне магистрале

Системска магистрала



Спољашња (екстерна) магистрала

- Повезује уређаје који су ван процесорског система
  - *USB*
  - *FireWire*
  - серијски интерфејс
  - паралелни интерфејс

Дељење магистрале

- Магистрала је дељени ресурс
  - Свака компонента повезана магистралом је корисник магистрале
- При дељењу магистрале постоји могућност истовремених активности на магистрали
  - Истовремена употреба магистрале од стране више компоненти доводи до неисправности

Трансакције магистрале

- Трансакција магистрале (енгл. *bus transaction*) је целовит низ поступака на магистрали које чине *добро дефинисану активност*
- Примери активности:
  - читање из меморије
  - писање у меморију
  - читање са улазног уређаја
  - писање на улазном уређају
  - ...
- Једна трансакција може да обухвати више операција
  - нпр: агресивно читање (енгл. *burst read*)

Трансакције магистрале (2)

- У оквиру једне трансакције препознају се:
  - главни корисник (енгл. *master*)
    - започиње трансакцију
  - подређени корисник (енгл. *slave*)
    - одговара на захтев главног корисника
- У једном тренутку највише једна трансакција на магистрали
- Свака трансакција има тачно једног главног корисника
- Неки уређаји могу бити само подређени корисници магистрале
- Други уређаји могу бити главни или подређени (али не у исто време)

Универзитет у Београду - Математички факултет

Посвећене магистрале

- Магистрала може да буде
  - посвећена једној улози
    - нпр. адресна магистрала служи само за преношење адреса
    - има већу пропусност
    - сложенија за имплементацију
  - мултиплексирана магистрала
    - иста магистрала преноси адресе, податке и контролне
    - једноставнија за имплементацију
    - има нижу пропусност

Универзитет у Београду - Математички факултет

Контролни сигнали

- Радом магистрале се управља посредством контролних сигнала путем
  - посвећене контролне магистрале или
  - мултиплексиране магистрале

Контролни сигнали (2)

- *Memory Read, Memory Write*
  - Означавају да је трансакција једна од операција са меморијом
- *I/O Read, I/O Write*
  - Означавају да трансакција обухвата улазно/излазну операцију
- *Ready*
  - Овај сигнал обично поставља подређена компонента
  - Обавештава главну компоненту да је потребно још времена да би се извршила операција
  - Главна операција обично реагује преласком у стање чекања

Контролни сигнали (3)

- *Bus Request, Bus Grant*
  - Пре сваке трансакције компоненте најпре морају захтевати да *добију* магистралу (*Bus Request*)
  - Арбитар магистрале одабире на који захтев може да се одговори позитивно и шаље одговарајући сигнал (*Bus Grant*) одговарајућој компоненти
- *Interrupt, Interrupt Acknowledgment*
  - Ови сигнали управљају системом прекида
  - Уређај који захтева прекид шаље сигнал *Interrupt*
  - Када је процесор у стању да обради прекид, шаље сигнал *Interrupt Acknowledgment* уређају
  - И у случају прекида је неопходна арбитража

Универзитет у Београду - Математички факултет

Контролни сигнали (4)

- *DMA Request, DMA Acknowledgment*
  - Ови сигнали се употребљавају за непосредан пренос података између улазно/излазних уређаја и меморије
  - Уобичајен начин комуникације уређаја са меморијом је *програмирани У/И*
    - посредством процесора (и система прекида)
  - Непосредан приступ меморији (енгл. *Direct Memory Access – DMA*) ослобађа процесор
  - Контролер *DMA* се понаша као главна компонента током употребе непосредног приступа меморији

Универзитет у Београду - Математички факултет

Контролни сигнали (5)

- *Clock*
  - Сигнал који служи за синхронизацију рада свих компоненти рачунарског система
- *Reset*
  - Сигнал који иницијализује рад система

Синхрона и асинхрона магистрала

- Магистрала може бити
  - Синхрона
    - Часовник обезбеђује синхронизацију свих операција
  - Асинхрона
    - Не користи се часовник за синхронизацију
    - Користе се операције руковања и додатни синхонизациони сигнали

Карактеристике магистрале

- Основне карактеристике магистрале се одређују тако да уравнотеже захтеве и трошкове:
  - Ширина магистрале
    - Односи се на магистрале адресе и података
    - Шира магистрала података подиже перформансе
    - Шира адресна магистрала повећава адресни простор
  - Тип магистрале
    - Посвећена или мултиплексирана
  - Операције магистрале
    - читање, писање, пренос блокова, читање са мењањем (енгл. *read-modify-write*) и прекиди
  - Арбитража
    - може да буде централизована и дистрибуирана
  - Подешавање времена
    - може бити синхроно или асинхроно

Универзитет у Београду - Математички факултет

Ширина магистрале

- Односи се (првенствено) на ширине магистрала адресе и података
  - Ширина контролне магистрале зависи од других фактора

Универзитет у Београду - Математички факултет

Ширина магистрале података

- Ширина магистрале података одређује величину података који се преносе магистралом
  - Основна мотивација за проширивање је подизање пропусности магистрале, а тиме и перформанси
  - Основна мотивација за сужавање је смањивање сложености и смањивање трошкова

Универзитет у Београду - Математички факултет

Ширина магистрале података (2)

- Ширина магистрале података не мора бити иста као ширина речи процесора
  - процесор *Intel 8086* је 16-битни процесор, а ширина магистрале података је 16 бита
  - процесор *Intel 8088* је 16-битни процесор (компатибилан са 8086), а ширина магистрале података је 8 бита
  - процесор *Intel Pentium* је 32-битни процесор, а ширина магистрале података је 64 бита
  - процесор *Intel Itanium* је 64-битни процесор, а ширина магистрале података је 128 бита

Универзитет у Београду - Математички факултет

Ширина адресне магистрале

- Ширина адресне магистрале одређује величину адресног простора
  - Ако је ширина магистрале n адресних линија, број адресибилних локација је 2^n
    - Једна адресибилна локација садржи једну меморијску реч
    - Меморијска реч је обично величине 1 бајт, али не мора бити тако
- Основна мотивација за проширивање је подизање пропусности магистрале, а тиме и перформанси
- Основна мотивација за сужавање је смањивање сложености и смањивање трошкова

Универзитет у Београду - Математички факултет

Ширина адресне магистрале (2)

- Примери:
  - 16-битни процесор *Intel 8086* (1979.) је имао 20 адресних линија
  - 16-битни процесор *Intel 80286* (1982.) је имао 24 адресне линије
  - 32-битни процесор *Intel 80386* (1985.) је имао 32 адресне линије

Универзитет у Београду - Математички факултет

Литература

- *Ненад Митић*, ***Увод у организацију рачунара***, *Математички факултет, 2009.*
- *Sivarama Dandamudi*, ***Fundamentals of Computer Organization and Design***, *Springer, 2002.*