ΗΥ 232
Οργάνωση και στον Σχεδίαση Η/Υ

Διάλεξη 7

Χρονισμός και Απόδοση Υπολογιστικών Συστημάτων

Νίκος Μπέλλας
Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

Τι σημαίνει απόδοση;

Αεροσκάφος	NYC to Paris	Επιβάτες
Boeing 747	6 hours	500
Concorde	3 hours	125

- Ποιος τύπος αεροσκάφους έχει υψηλότερη απόδοση?
- Εάν είσαι επιβάτης : χρόνος πτήσης (πόση ώρα για να γίνει η εργασία)
  - Χρόνος πτήσης, χρόνος εκτέλεσης, χρόνος απόκρισης (latency)
- Εάν είσαι η Air France: πόσους επιβάτες μπορείς να εξυπηρετήσεις σε δεδομένο χρονικό διάστημα
  - Επιβάτες ανά ώρα, ρυθμός απόδοσης (throughput)

Μετρικές Απόδοσης

- *Χρόνος Απόκρισης* (ή *Χρόνος Εκτέλεσης*)
  - Latency (Execution time)
  - Πόσος χρόνος απαιτείται για την εκτέλεση μιας εργασίας
- *Throughput*
  - Συνολική ποσότητα εργασίας που διεκπεραιώνεται σε ένα δεδομένο χρόνο
    - πχ εργασίες/ώρα, bytes/sec, κοκ.
- Ο *Χρόνος Απόκρισης* και το *Throughput* δεν είναι το ίδιο σημαντικές μετρικές για διαφορετικά συστήματα
  - Database transaction systems (latency)
  - Graphics rendering system (GPU) (throughput)

Συγκριτική Απόδοση

- Ορισμός: *Απόδοση = 1 / Χρόνος Εκτέλεσης*
- "Το ότι το σύστημα *X* είναι *n* φορές πιο γρήγορο από το σύστημα *Y* σημαίνει: "

$$\text{Απόδοση}_X / \text{Απόδοση}_Y =$$
$$\text{Χρόνος Εκτέλεσης}_Y / \text{Χρόνος Εκτέλεσης}_X = n$$

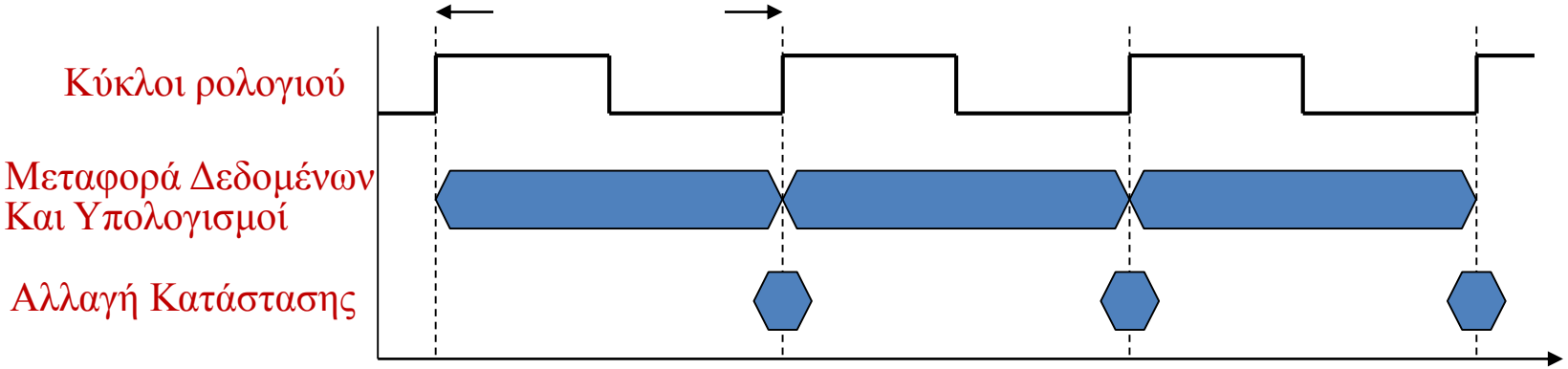
- Για παράδειγμα, ο χρόνος εκτέλεσης ενός προγράμματος:
  - 10s on A, 15s on B
  - Execution Time$_B$ / Execution Time$_A$ = 15s / 10s = 1.5
  - Ο Α είναι 1.5 φορές γρηγορότερος από τον Β

Μέτρηση Χρόνου Εκτέλεσης

- Παρελθών Χρόνος (Elapsed Time)
  - Συνολικός χρόνος εκτέλεσης μιας εργασίας που περιλαμβάνει: επεξεργασία, Input/Output (πχ προσπέλαση στον σκληρό δίσκο), κλήσεις στο λειτουργικό σύστημα, κοκ.
- Χρόνος CPU (CPU time)
  - Περιλαμβάνει μόνο τον χρόνο που η CPU επεξεργάζεται μια διεργασία
    - Δεν περιλαμβάνει Input/Output ή χρόνο που η CPU ασχολείται με άλλες διεργασίες.
  - CPU time = User CPU time + System CPU time

Ρολόι CPU

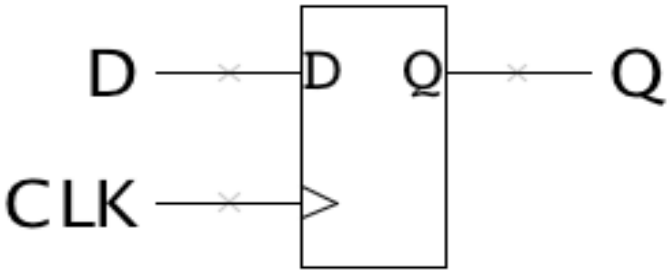
- Η λειτουργία κάθε ψηφιακού συστήματος βασίζεται σε ένα ρολόι που τρέχει με σταθερό ρυθμό και προσδιορίζει πότε προκύπτουν διάφορα συμβάντα στο hardware

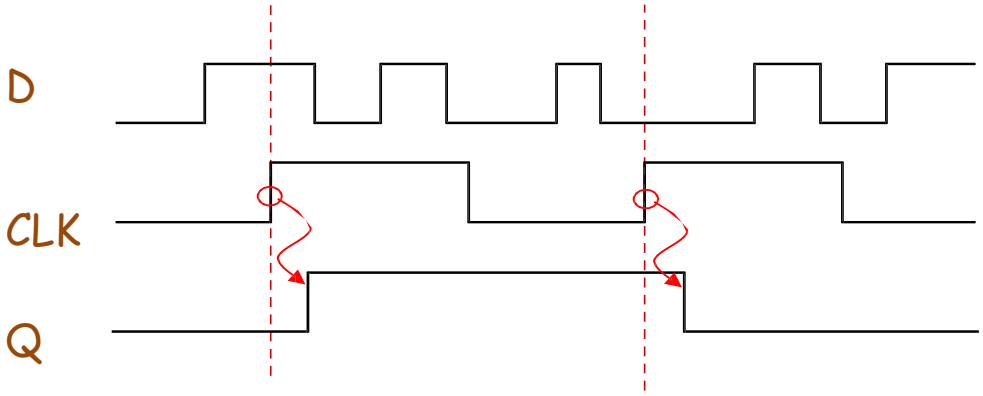


  - Περίοδος Ρολογιού (clock period): ο χρόνος για έναν πλήρη κύκλο ρολογιού
    - πχ, 250ps = 0.25ns = 250×10^{-12}s
  - Συχνότητα ρολογιού (clock frequency or rate): κύκλοι ρολογιού ανά μονάδα χρόνου = 1/ Περίοδος
    - πχ, 4.0GHz = 4000MHz = 4.0×10^{9}Hz
    - Αντιστοιχεί σε περίοδο 1/4GHz = 0.25ns

Το δομικό στοιχείο κάθε ακολουθιακού κυκλώματος: D flip-flop

Ακμοπυροδότητο D flip-flop:
η είσοδος D περνάει στην έξοδο Q μόνο την χρονική στιγμή της θετικής ακμής του ρολογιού.
Σε οποιαδήποτε άλλη χρονική στιγμή, η είσοδος D αγνοείται

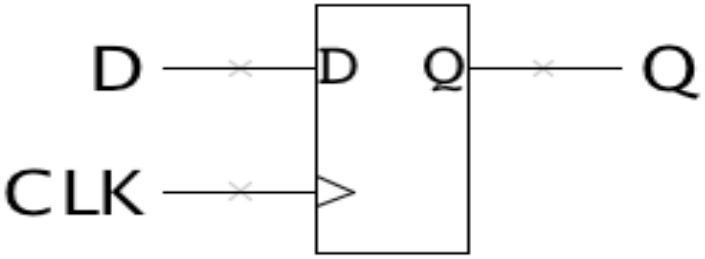


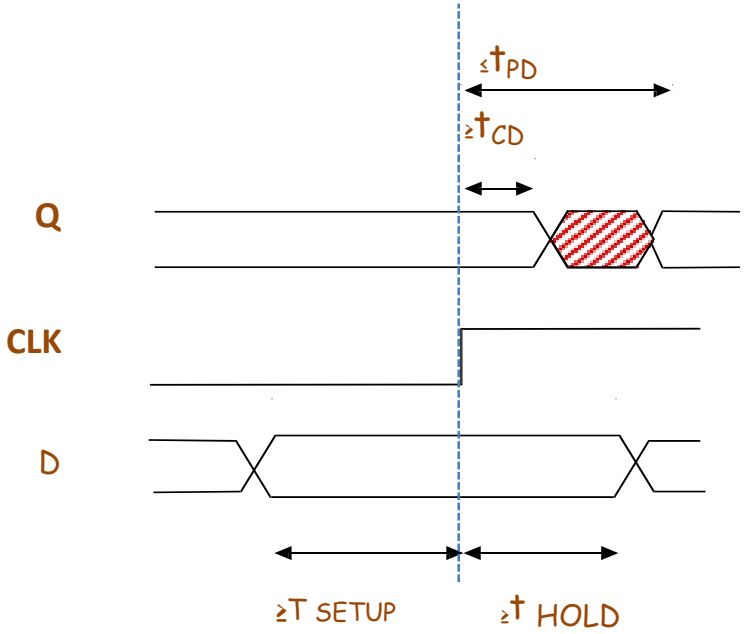



```
module dff (input D, CLK,
                output reg Q);

always @(posedge clk)
   begin
      Q<=D;
   end
endmodule;
```

Χρονισμός του D flip-flop (I)





t_{PD}: maximum propagation delay, CLK → Q

t_{CD}: minimum contamination delay, CLK →Q

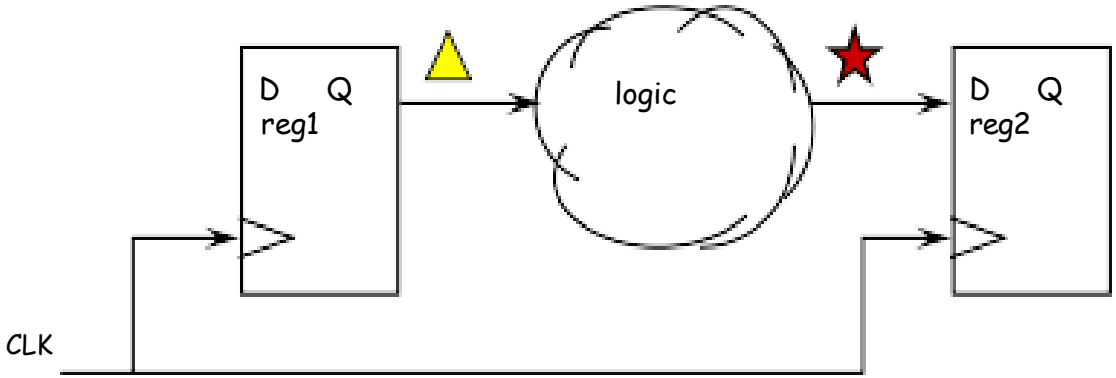
t_{SETUP}: setup time

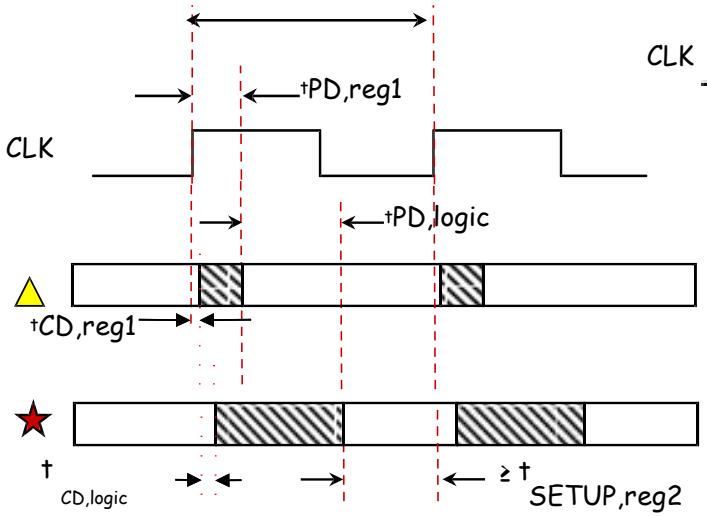
Ο χρόνος που πρέπει η είσοδος D να έχει σταθεροποιηθεί πριν την θετική ακμή του CLK

t_{HOLD}: hold time

Ο χρόνος που πρέπει η είσοδος D να παραμείνει σταθερή μετά την θετική ακμή του CLK

Χρονισμός του D flip-flop (II)

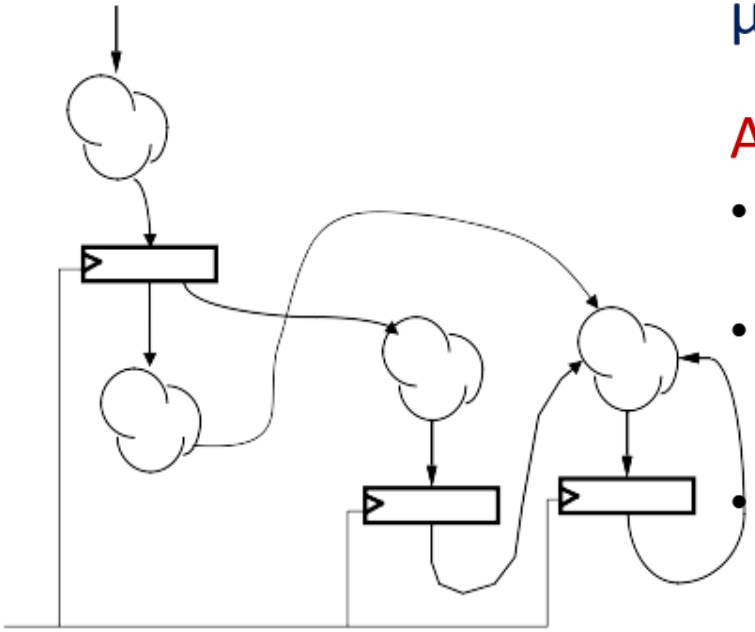




Θέσε T_{clk} που να ικανοποιεί την εξίσωση

$$T_{PD_{reg1}} + T_{PD_{logic}} + T_{SETUP_{reg2}} \leq T_{clk}$$

Σύγχρονα ακολουθιακά Κυκλώματα ενός Ρολογιού



Σχεδόν όλα τα ψηφιακά συστήματα βασίζονται σε ένα ρολόι και συγχρονίζουν τις εργασίες τους σύμφωνα με αυτό.

Από εδώ και πέρα για όλα τα συστήματα:

- Υπάρχει ένα μόνο ρολόι που μοιράζεται σε όλες τις ακολουθιακές συσκευές
- Μας ενδιαφέρει η τιμή των συνδυαστικών κυκλωμάτων λίγο πριν την θετική ακμή του ρολογιού αυτού
- Η περίοδος του ρολογιού T_{clk} είναι μεγαλύτερη από την καθυστέρηση του συνδυαστικού κυκλώματος (βλ. εξίσωση σε προηγ. σελίδα)

Χρόνος CPU

$$\text{Χρόνος CPU} = \text{Αριθμός Κύκλων CPU} \times \text{Περίοδος Ρολογιού}$$

$$= \frac{\text{Αριθμός Κύκλων CPU}}{\text{Συχνότητα Ρολογιού}}$$

- Η απόδοση του συστήματος μπορεί να βελτιωθεί με το να:
  - Μειώσουμε τον αριθμό των κύκλων CPU που απαιτούνται για να εκτελεσθεί το πρόγραμμα
  - Αυξήσουμε την συχνότητα ρολογιού
  - Αυτά τα δύο δεν είναι όμως ανεξάρτητα

Παράδειγμα χρόνου CPU

- Υπολογιστής Α: Ρολόι 2GHz, Χρόνος CPU 10s
- Θέλουμε να σχεδιάσουμε έναν Υπολογιστή Β
  - Στόχος μας να πέσουμε στα 6s χρόνου CPU
  - Πιο γρήγορο ρολόι απαιτεί 1.2Χ περισσότερους κύκλους
  - **Ερώτημα**: Πόσο πρέπει να είναι το ρολόι του Υπολογιστή Β;

$$\text{Συχνότητα}_B = \frac{\text{Αριθμός Κύκλων CPU}_B}{\text{Χρόνος CPU}_B}$$

$$= \frac{1.2 \times \text{Αριθμός Κύκλων CPU}_A}{6\text{s}}$$

$$\text{Αριθμός Κύκλων CPU}_A = \text{Χρόνος CPU}_A \times \text{Συχνότητα}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{Συχνότητα}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

Με λίγη περισσότερη λεπτομέρεια

$$\text{Χρόνος CPU} = \text{Αριθμός Εντολών στο Πρόγραμμα} \times \text{CPI} \times \text{Περίοδος Ρολογιού}$$

$$= \frac{\text{Αριθμός Εντολών} \times \text{CPI}}{\text{Συχνότητα Ρολογιού}}$$

- Αριθμός Εντολών στο Πρόγραμμα
  - Εξαρτάται από το πρόγραμμα, την αρχιτεκτονική συνόλου εντολών (πχ MIPS, x86, ARM) και τον compiler.
- CPI ή Cycles per Instruction
  - Πόσες εντολές μπορεί να εκτελέσει ένας επεξεργαστής σε έναν κύκλο μηχανής.
  - Πάντα θεωρούσαμε CPI=1 στο ΗΥ232.
  - Πιο πολύπλοκες εντολές μπορεί να έχουν CPI>1

Παράδειγμα

- Δύο υπολογιστές MIPS τρέχουν το ίδιο πρόγραμμα
- Υπολογιστής Α: Περίοδος = 250ps, CPI = 2.0
- Υπολογιστής Β : Περίοδος = 500ps, CPI = 1.2
- Ποιος είναι πιο γρήγορος και κατά πόσο;

$$\text{ΧρόνοςCPU}_A = \text{ΑριθμόςΕντολ.} \times \text{CPI}_A \times \text{Περίοδος}_A$$
$$= \text{AE} \times 2.0 \times 250\text{ps} = \text{AE} \times 500\text{ps}$$

Α πιο γρήγορο

$$\text{ΧρόνοςCPU}_B = \text{ΑριθμόςΕντολ.} \times \text{CPI}_B \times \text{CycleTime}_B$$
$$= AE \times 1.2 \times 500\text{ps} = AE \times 600\text{ps}$$

$$\frac{\text{ΧρόνοςCPU}_B}{\text{ΧρόνοςCPU}_A} = \frac{AE \times 600\text{ps}}{AE \times 500\text{ps}} = 1.2$$

κατά 1.2Χ

Παράδειγμα Υπολογισμού Μέσου CPI

- Έστω ότι τα προγράμματα MIPS που τρέχουμε αποτελούνται από τρείς κατηγορίες εντολών A, B, C.

Κατηγορία Εντολών	A (add, sub)	B (lw, sw)	C (beq, bne)
CPI για κατηγορία	1	2	3
AE στο Προγρ. 1	2	1	2
AE στο Προγρ. 2	4	1	1

- Πρόγραμμα 1: AE = 5
  - Αριθμός Κύκλων $= 2 \times 1 + 1 \times 2 + 2 \times 3 = 10$
  - Μέσο CPI = 10/5 = 2.0
- Πρόγραμμα 2 : AE = 6
  - Αριθμός Κύκλων $= 4 \times 1 + 1 \times 2 + 1 \times 3 = 9$
  - Μέσο CPI = 9/6 = 1.5

Παράδειγμα κώδικα MIPS (I)

Συνάρτηση καθαρισμού (*clear*) μιάς περιοχής μνήμης

```
clear1(int array[], int size) {
  int i;
  for (i = 0; i < size; i += 1)
    array[i] = 0;
}
```

```
clear2(int *array, int size) {
  int *p;
  for (p = &array[0]; p < &array[size];
       p = p + 1)
    *p = 0;
}
```

```
      move $t0,$zero   # i = 0
loop1: sll $t1,$t0,2   # $t1 = i * 4
      add $t2,$a0,$t1  # $t2 =
                       #   &array[i]
      sw $zero, 0($t2) # array[i] = 0
      addi $t0,$t0,1   # i = i + 1
      slt $t3,$t0,$a1  # $t3 =
                       #   (i < size)
      bne $t3,$zero,loop1 # if (…)
                          # goto loop1
```

```
      move $t0,$a0     # p = & array[0]
      sll $t1,$a1,2    # $t1 = size * 4
      add $t2,$a0,$t1  # $t2 =
                       #   &array[size]
loop2: sw $zero,0($t0) # Memory[p] = 0
      addi $t0,$t0,4   # p = p + 4
      slt $t3,$t0,$t2  # $t3 =
                       #(p<&array[size])
      bne $t3,$zero,loop2 # if (…)
                          # goto loop2
```

Αριθμός εντολών που εκτελούνται, *clear1*: 6N+1, *clear2*: 4N+3

Παράδειγμα κώδικα MIPS (II)

- Έστω ότι θέλουμε να τρέξουμε τα *clear1* και *clear2* για $N=10^6$ σε δύο υπολογιστές MIPS
- Υπολογιστής Α (*clear1*): Συχνότητα = 1 GHz, CPI = 2.0
- Υπολογιστής Β (*clear2*) : Περίοδος = 500 MHz, CPI = 2.0
- Ποιος τρέχει πιο γρήγορα το πρόγραμμα και κατά πόσο;

$$\text{ΧρόνοςCPU}_A = \text{ΑριθμόςΕντολών}_A \times \text{CPI}_A \times \text{Περίοδος}_A$$

$$= (6*10^6 + 1) \times 2.0 \times 1\,\text{ns} = 12\,\text{us}$$

Α πιο γρήγορος

$$\text{ΧρόνοςCPU}_B = \text{ΑριθμόςΕντολών}_B \times \text{CPI}_B \times \text{CycleTime}_B$$

$$= (4*10^6 + 3) \times 2.0 \times 2\,\text{ns} = 16\,\text{us}$$

κατά 1.33Χ

$$\frac{\text{ΧρόνοςCPU}_B}{\text{ΧρόνοςCPU}_A} = \frac{16}{12} = 1.33$$