ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΕΠΕΑΕΚ
ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΣΥΓΧΡΗΜΑΤΟΔΟΤΗΣΗ
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
ΕΛΛΑΔΑ 2008 ΚΟΙΝΩΝΙΚΟ ΠΛΑΙΣΙΟ ΣΤΗΡΙΞΗΣ
Ανάπτυξη παντού. Ανάπτυξη για όλους.
Η ΠΑΙΔΕΙΑ ΣΤΗΝ ΚΟΡΥΦΗ
Επιχειρησιακό Πρόγραμμα
Εκπαίδευσης και Αρχικής
Επαγγελματικής Κατάρτισης

Μοντέλα Αρχιτεκτονικής στην Σύνθεση

Σχεδιαστικά Στυλ & Αρχιτεκτονική

Ο σχεδιαστής επιλέγει

π.χ. επεξεργαστής

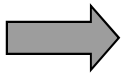
<u>Σχεδιαστικό στυλ</u>

- ✓ prioritized interrupt
- ✓ instruction buffer
- ✓ bus-oriented datapath
- ✓ serial I/O
- ✓ direct memory access …

<u>Αρχιτεκτονική</u>

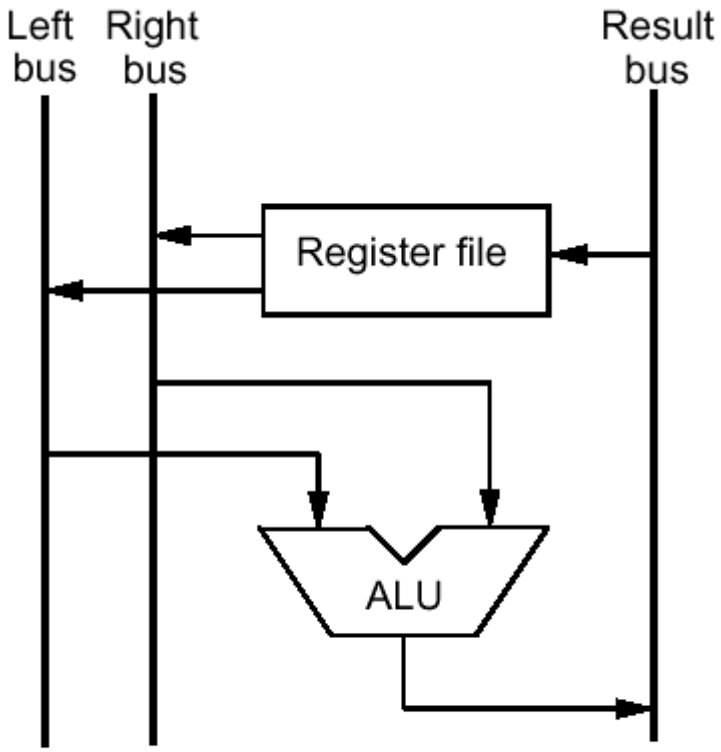
- ✓ αριθμός καταχωρητών
- ✓ αριθμός διαύλων
- ✓ βαθμίδες pipeline
- ✓ υλοποίηση εντολών
- ✓ προσπέλαση μνημών …

Κάθε αλγόριθμος σύνθεσης έχει σαν στόχο συγκεκριμένη αρχιτεκτονική



όσο πιο ρεαλιστική είναι η αρχιτεκτονική αυτή, τόσο καλύτερη η ποιότητα της σχεδίασης, και τόσο πιο εκλεπτυσμένος αλγόριθμος απαιτείται.

Παράδειγμα μη ρεαλιστικής αρχιτεκτ.



3−bus nonpipelined design

Σε έναν κύκλο ρολογιού γίνονται

1. ανάγνωση 2 τελουμένων
2. πράξη
3. αποθήκευση αποτελέσματος

Κύκλος ρολογιού = 100 ns

Εξαρτημένες πράξεις:

```
Program 1:    x <= a + b;  (100ns)
              y <= c − x;  (100ns)
```

Ανεξάρτητες πράξεις: 200ns

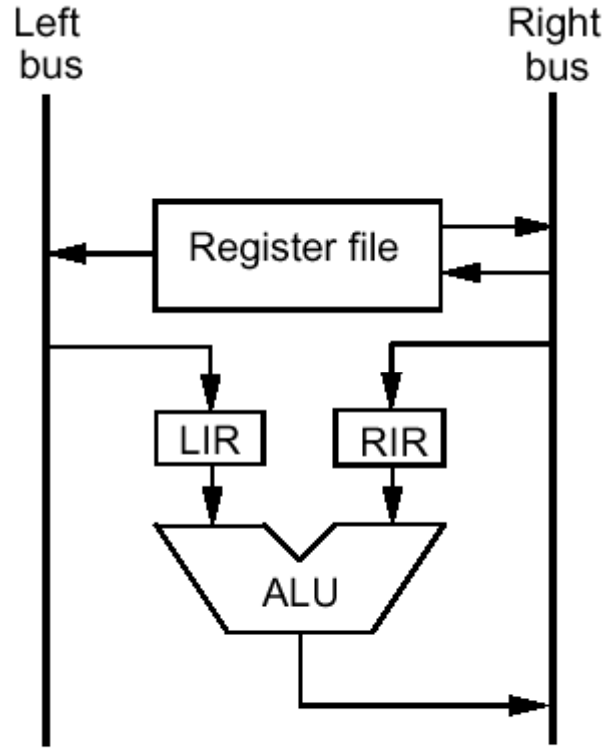
```
Program 2:    x <= a + b;  (100ns)
              y <= c − d;  (100ns)
```

Παράδειγμα μη ρεαλιστικής αρχιτεκτ.

- Το μοντέλο αυτό είναι πολύ απλό και μη-ρεαλιστικό
- Τα 3 bus κοστίζουν πολύ αφού καταλαμβάνουν μεγάλο τμήμα του χώρου.
- Ο κύκλος ρολογιού είναι μεγάλος
- Μεγάλο μέρος του κύκλου οι μονάδες μένουν ανεκμετάλλευτες (αναμονή) αφού λειτουργούν εναλλακτικά.
- Η σύνθεση του είναι απλή

Παράδειγμα ρεαλιστικής αρχιτεκτονικής.

Προσθήκη καταχωρητών LIR, RIR.



2−bus pipelined design

Βελτιώσεις:

1. Μισός κύκλος ρολογιού (50 ns).
2. Αύξηση χρόνου χρήσης μονάδων σε κάθε κύκλο.
3. Απαιτεί μόνο 2 buses

Σε έναν κύκλο ρολογιού γίνονται

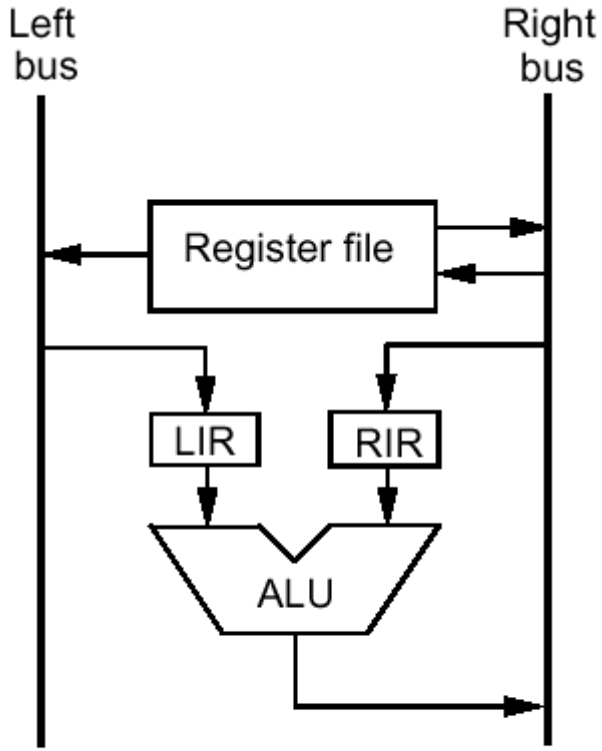
ανάγνωση File=>LIR, RIR ή

ανάγνωση File=>LIR & πράξη με αποθήκευση στο RIR

πράξη & αποθήκευση ALU=>File

Απαιτεί πιο περίπλοκο αλγόριθμο σύνθεσης.

Παράδειγμα ρεαλιστικής αρχιτεκτονικής



2−bus pipelined design

Εξάρτηση πράξεων:

```
LIR <= a; RIR <= b;                    (50ns)
x, RIR <= LIR + RIR; LIR <= c;         (50ns)
y <= LIR − RIR;                        (50ns)
```

150ns = βελτίωση 25%

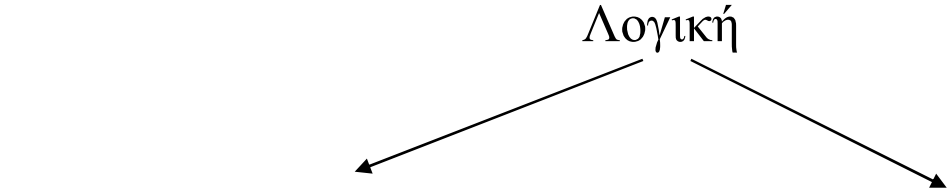
Ανεξάρτητες πράξεις:

```
LIR <= a;  RIR <= b;                   (50ns)
x <= LIR + RIR;                        (50ns)
LIR <= c; RIR <= d;                    (50ns)
y <= LIR − RIR;                        (50ns)
```

200ns = βελτίωση 0%

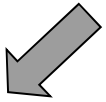
Ο αλγόριθμος σύνθεσης απαιτεί διερεύνηση της εξάρτησης πράξεων για εφαρμογή ή όχι.

Συνδυαστική Λογική



- Καταχωρητές
- Αρχεία καταχωρητών
- Μνήμες

- Αθροιστές / Πολλαπλασιαστές
- ALUs
- Ολισθητές, αυξητές, …



- ✓ Τα είδη λογικής μπορούν να κατηγοριοποιηθούν σαν sliceable (carry ripple adder) και unsliceable (carry look ahead adder).
- ✓ Η λογική sliceable είναι εύκολα συνθέσιμη και υλοποιήσιμη σε κύκλωμα, αλλά συνήθως όχι αποδοτική.

Παράδειγμα: Αθροιστής ριπής – Αθροιστής Πρόβλεψης Κρατουμένου

Συνδυαστική Λογική

Εκφράσεις Boole για αθροιστή

$c_0 = C_{in}$

$c_{i+1} = a_i b_i + c_i(a_i + b_i)$

$sum_i = a_i$ EXOR b_i EXOR c_i

$C_{out} = c_{16}$

Sliceable

Με άμεση εφαρμογή έχουμε αθροιστή ριπής

UnSliceable

Με αντικατάσταση c_{i+1} από λογική c_i αναδρομικά έχουμε αθροιστή πρόβλεψης κρατουμένου.

- Τρόποι Περιγραφής Συνδυαστικής Λογικής:

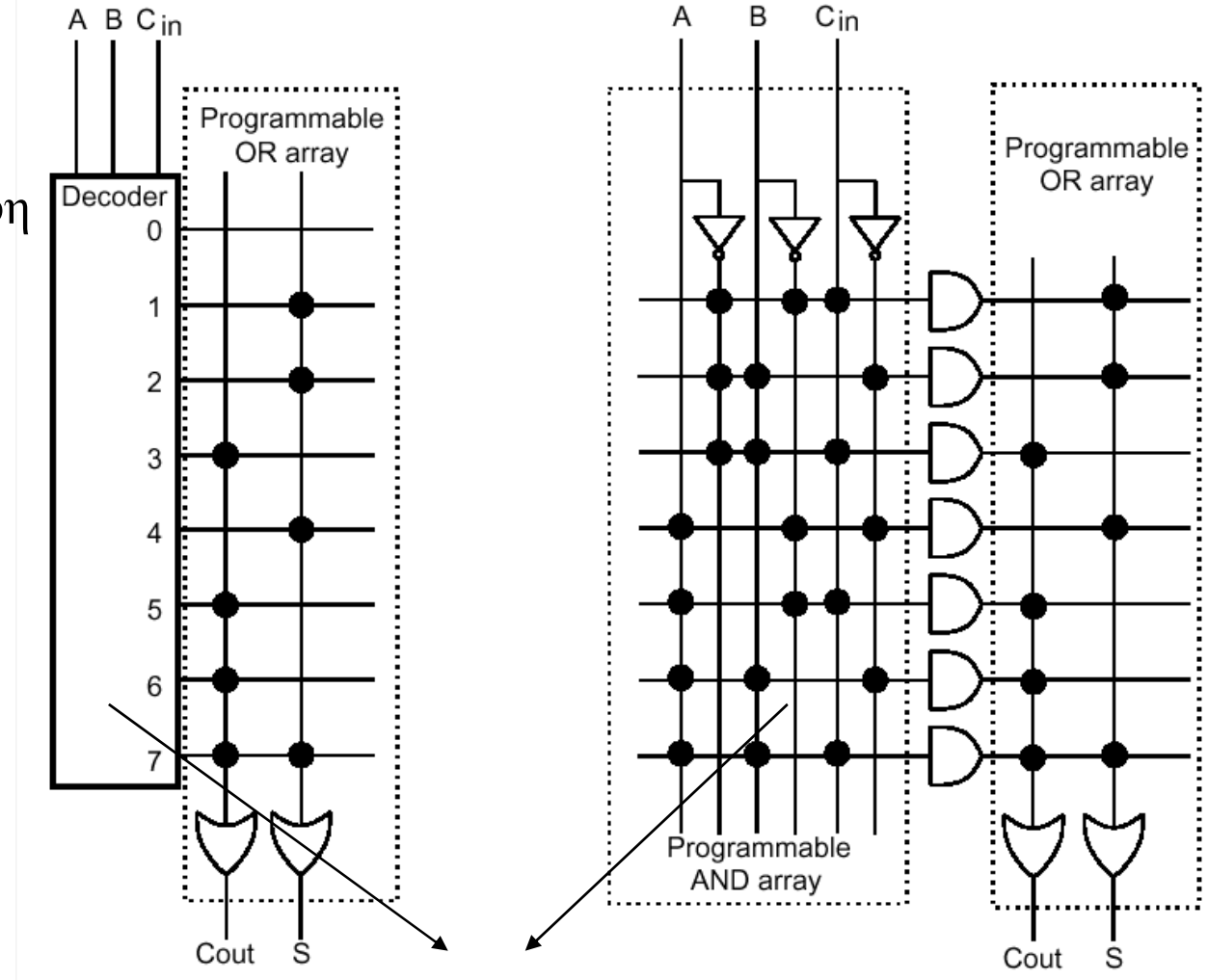
  α) Εκφράσεις Boole,

  β) Πίνακας Αλήθειας (για λίγες εισόδους ή για λίγους άσσους στην έξοδο).

- Τρόποι Υλοποίησης: ROM, PLA, Decoders, Primitive Gates

Συνδυαστική Λογική

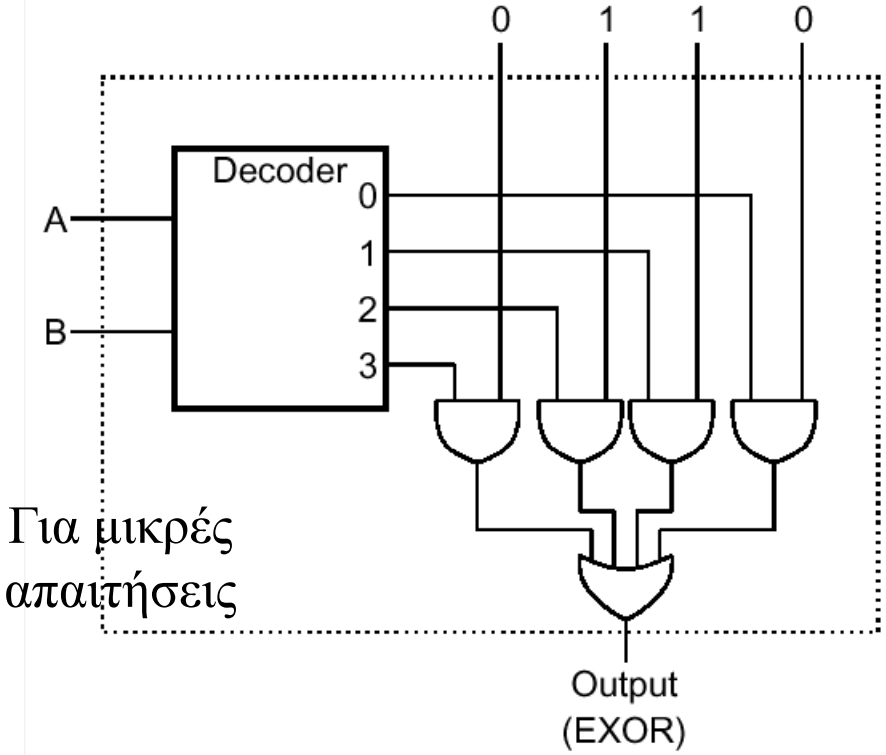
ROM (πλήρη αθροιστή)



PLA

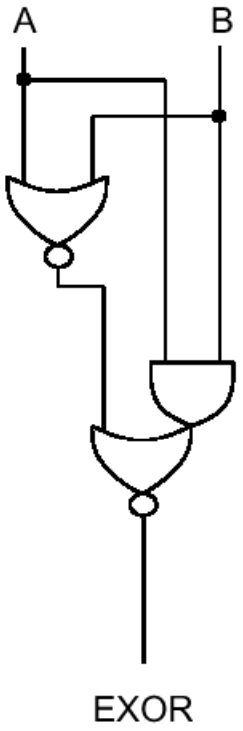
Διαφορά στο μέγεθος του decoder

Συνδυαστική Λογική



Για μικρές απαιτήσεις

Decoder implementation of an EXOR gate



Πιο αποδοτικός τρόπος

Logic gate implementation of an EXOR gate

Συνδυαστική Λογική

Σύγκριση Αρχιτεκτονικών Συνδυαστικών Κυκλωμάτων

- Η ROM είναι η πιο ακριβή.
- Η PLA έχει προγραμματιζόμενο και το AND πίνακα άρα έχει μεγαλύτερη ευελιξία σε μικρότερο χώρο από την ROM.
- Η ROM υπερτερεί όταν έχουμε πολλές εξόδους (πολλαπλή χρήση των ελαχιστόρων).
- Η προσθήκη μίας εισόδου στην ROM απαιτεί διπλασιασμό της επιφάνειας ενώ στην PLA αυξάνει κατά λίγο την επιφάνεια.
- Οι πύλες επιτρέπουν την σχεδίαση λογικής πολλαπλών επιπέδων.
- Οι πύλες έχουν την καλύτερη απόδοση.
- ROM και PLA κατασκευάζονται και ελέγχονται ευκολότερα.

Συνδυαστική Λογική

Διαδικασία Σύνθεσης για Συνδυαστικά Κυκλώματα

- Compilation (μετάφραση περιγραφής σε ενδιάμεση μορφή)
- Minimization (ελαχιστοποίηση αριθμού και μεγέθους παραγόντων)

→ ROM-PLA

- Optimization (μείωση κρίσιμων μονοπατιών)
- Technology Mapping (χρήση βέλτιστων πυλών για κάθε περίπτωση)
- Transistor Sizing (ανάλογα με απαιτήσεις ταχύτητας / επιφάνειας)

Finite State Mashines

- Είναι το πλέον χρησιμοποιούμενο μοντέλο σχεδίασης
- Αποτελείται από ένα σύνολο εισόδων, καταστάσεων, μεταβάσεων, και ενεργειών (εξόδων).

<S, I, O, f: S x I -> S, h: S x I -> O>

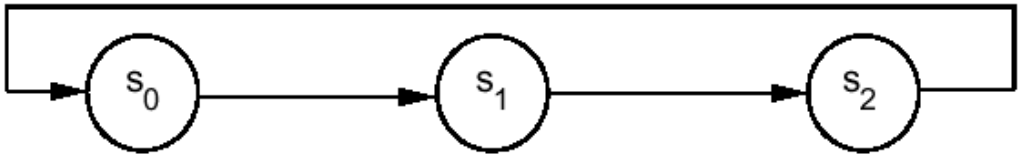
Είδη Μηχανών Κατάστασης

1. Autonomous
2. State based
3. Transition based
4. Machines with datapath
5. Communicating machines

Autonomous FSMs

- Δεν υπάρχουν εξωτερικοί είσοδοι (I=∅).
- Χρησιμοποιείται για μονάδες που τρέχουν ελεύθερα.
- Οι καταστάσεις αποθηκεύονται σε Flip Flops.

Modulo−3 counter



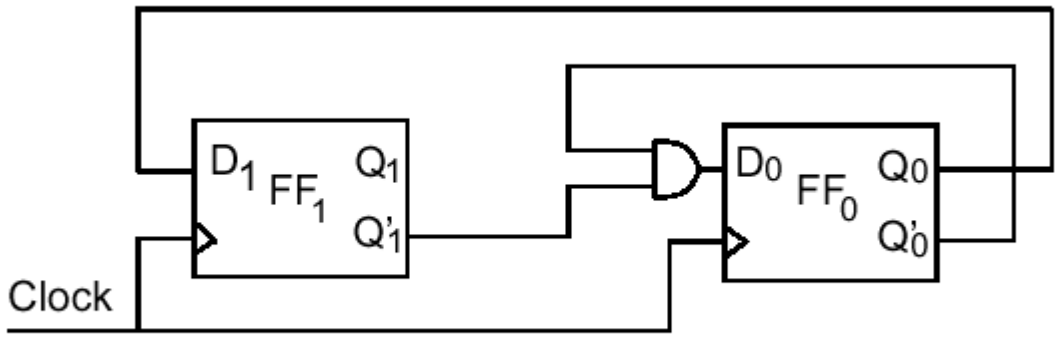
State diagram

Κωδικοποίηση με 2 bits

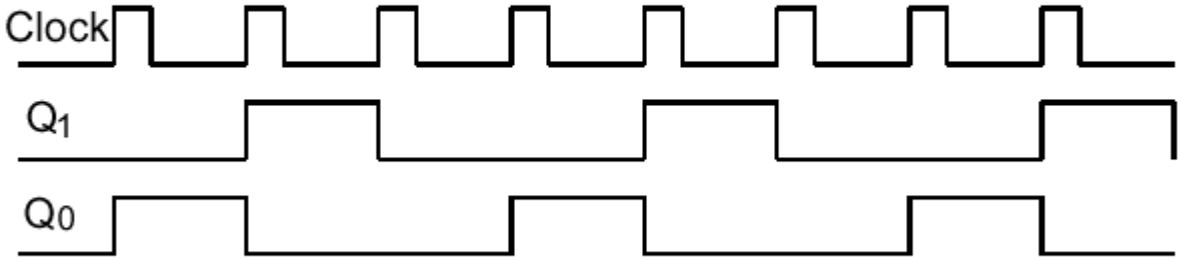
Present state	Next state
$Q_1 Q_0$	$Q_1 Q_0$
s_0 = 0 0	s_1 = 0 1
s_1 = 0 1	s_2 = 1 0
s_2 = 1 0	s_0 = 0 0

Next−state table

Autonomous FSMs



Logic implementation



State waveforms

Autonomous FSMs

Με Προσθήκη εξόδου

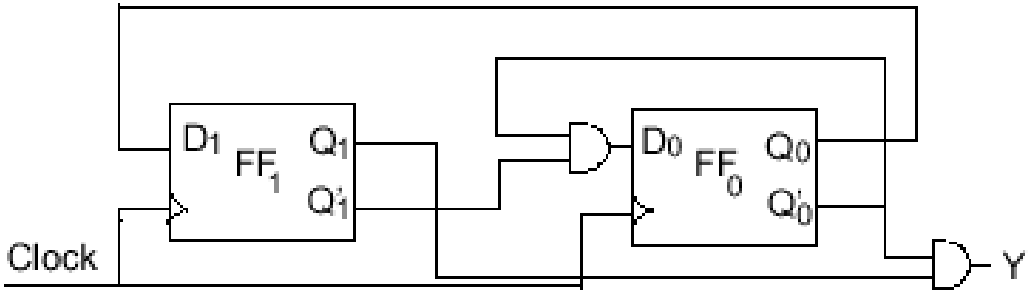
Διαιρέτης με 3

(Κάθε τρεις παλμούς η έξοδος είναι στο 1 για έναν παλμό).

MODULO−3 DIVIDER

Present state	Next state	Output
Q_1Q_0	Q_1Q_0	Y
$s_0 = 0\ 0$	$s_1 = 0\ 1$	0
$s_1 = 0\ 1$	$s_2 = 1\ 0$	0
$s_2 = 1\ 0$	$s_0 = 0\ 0$	1

State table



Logic implementation

State Based/Transition Based FSMs

- Έχουν σύνολο εισόδων.
- Διαφέρουν στον καθορισμό της συνάρτησης εξόδου:

Στο **state based** (Moore) η έξοδος εξαρτάται μόνο από την κατάσταση του FSM (αλλάζει μόνο σε ακμές).

Στο **transition based** (Mealy) η έξοδος εξαρτάται από την κατάσταση του FSM και τις εισόδους (αλλάζει και ενδιάμεσα σε ακμές).

Παράδειγμα State Based FSMs

Παράδειγμα state based FSM (modulo 3 divider)

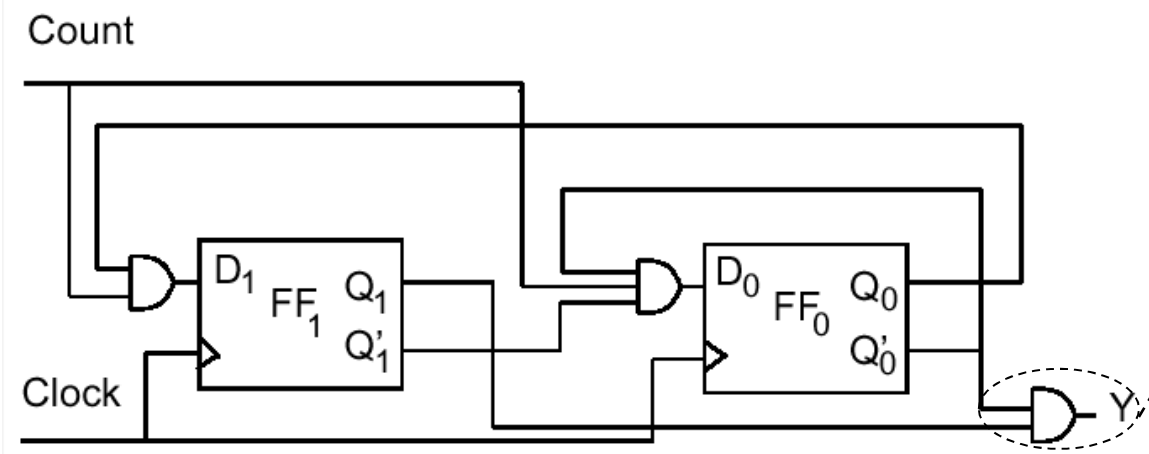
Προσθήκη εισόδου Count που επιτρέπει την μέτρηση.

Present state	Input	Next state
Q_1Q_0	Count	Q_1Q_0
$s_0 = 0\ 0$	1	$s_0 = 0\ 1$
$s_1 = 0\ 1$	1	$s_2 = 1\ 0$
$s_2 = 1\ 0$	1	$s_0 = 0\ 0$
don't care	0	$s_0 = 0\ 0$

Present state	Output
$Q_1\ Q_0$	Y
$s_0 = 0\ 0$	0
$s_1 = 0\ 1$	0
$s_2 = 1\ 0$	1

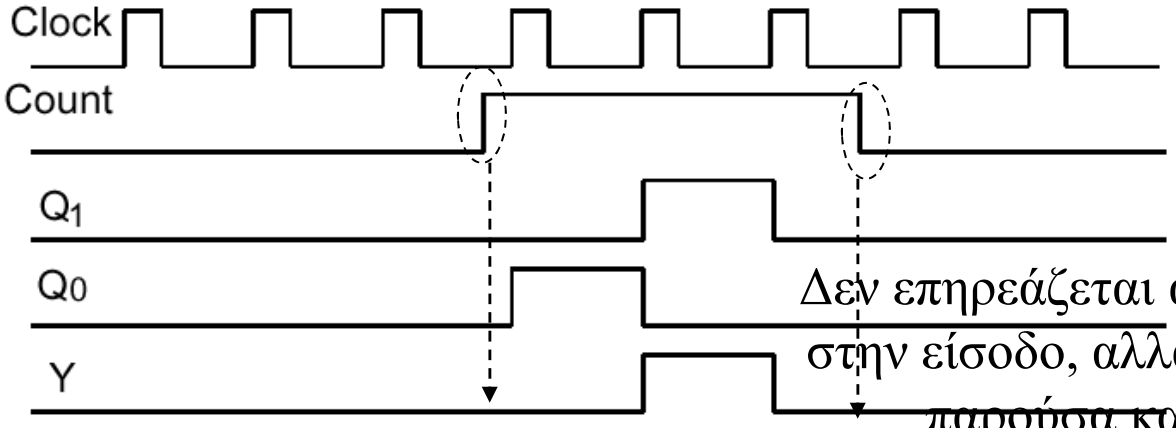
Οι πίνακες είναι χωριστοί αφού η έξοδος εξαρτάται μόνο από την παρούσα κατάσταση και όχι την είσοδο.

Παράδειγμα State Based FSMs



Logic implementation

Εξαρτάται μόνο από την κατάσταση.



Input and output waveforms

Δεν επηρεάζεται από τις αλλαγές στην είσοδο, αλλά μόνο από την παρούσα κατάσταση.

Παράδειγμα Transition Based FSMs

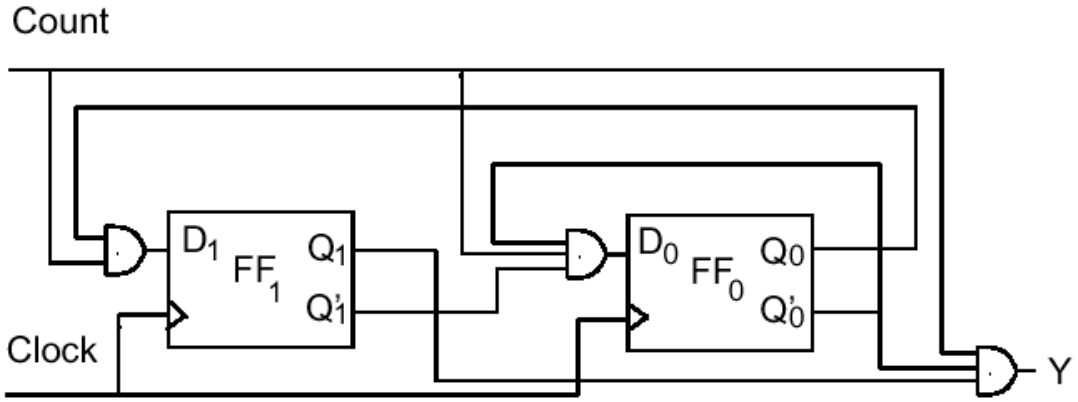
Παράδειγμα transition based FSM (modulo 3 divider)

Present state	Input	Next state	Output
Q_1Q_0	Count	Q_1Q_0	Y
$s_0 = 0\ 0$	1	$s_0 = 0\ 1$	0
$s_1 = 0\ 1$	1	$s_2 = 1\ 0$	0
$s_2 = 1\ 0$	1	$s_0 = 0\ 0$	1
don't care	0	$s_0 = 0\ 0$	0

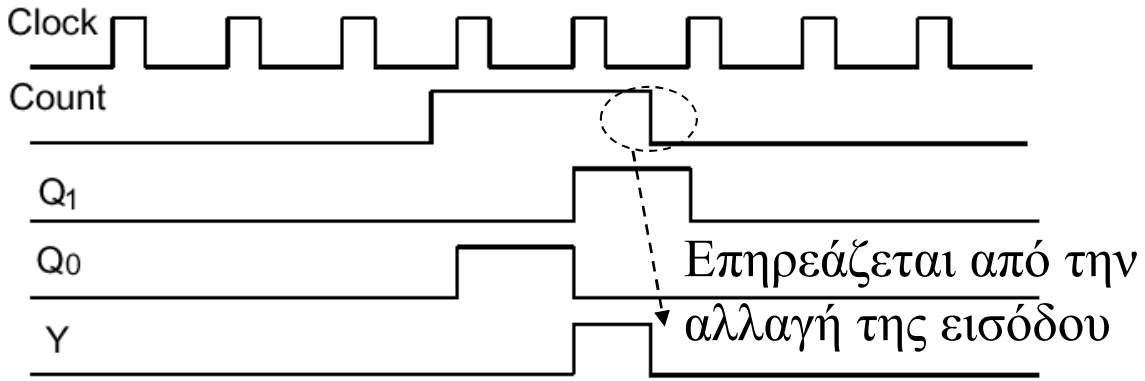
Next−state and output tables

Ο πίνακας είναι ενιαίος αφού η έξοδος εξαρτάται από την παρούσα κατάσταση και την είσοδο.

State Based/Transition Based FSMs



Logic implementation



Input and output waveforms

State Based/Transition Based FSMs

Η διαδικασία σύνθεσης για τα state/transition based FSMs περιλαμβάνει τα ακόλουθα βήματα

- Compilation
- State minimization
  - Ελαχιστ. αριθμού καταστάσεων (απαλοιφή πλεοναζόντων).
  - Ελαχιστ. αριθμού Flip Flops
  - Ελαχιστ. λογικής εξόδου και μετάβασης καταστάσεων (πλεονασμός καταστάσεων)

- State encoding (μπορεί να βοηθήσει στην ελαχιστοποίηση). Με trade-off μπορούμε να ελαχιστοποιήσουμε αριθμό Flip Flops ή συνδυαστικής λογικής μετάβασης/εξόδου
  - Δυαδική Κωδικοποίηση
  - One-Hot
- Σύνθεση συναρτήσεων μετάβασης/εξόδου (συνδυαστική λογική)

FSM with DataPath

- Τα FSMs αποδίδουν για ως και μερικές εκατοντάδες καταστάσεις.
- Απλές μονάδες (I/O interfaces) μπορεί να έχουν χιλιάδες καταστάσεις.
- Μοντέλο FSM με DataPath: Κωδικοποιούμε καταστάσεις με ακέραιες και πραγματικές μεταβλητές (ακέραιος 16 bits = 65536 καταστάσεις).

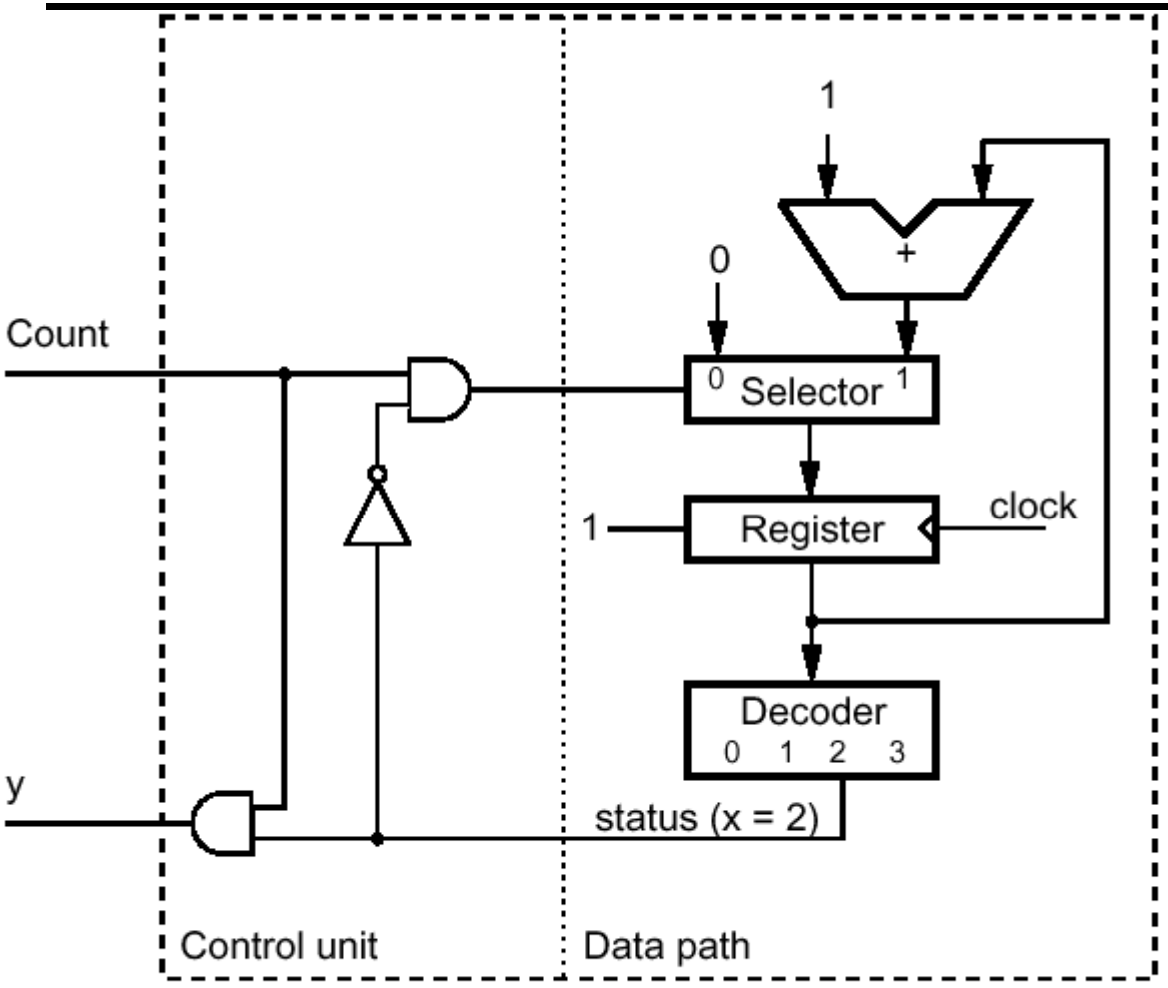
Παράδειγμα

- module-3 divider με **μία** κατάσταση
- Μία μεταβλητή x (2 bits) αναπαριστά τον μετρητή

Present State	Input	Next State	Output
s_0	(Count = 1) AND (x ≠ 2)	s_0	x = x + 1, Y = 0
	(Count = 1) AND (x = 2)		x = 0, Y = 1
	Count = 0		x = 0, Y = 0

Next state and output table

FSM with DataPath (FSMD)



Datapath implementation

Πλεονέκτημα:

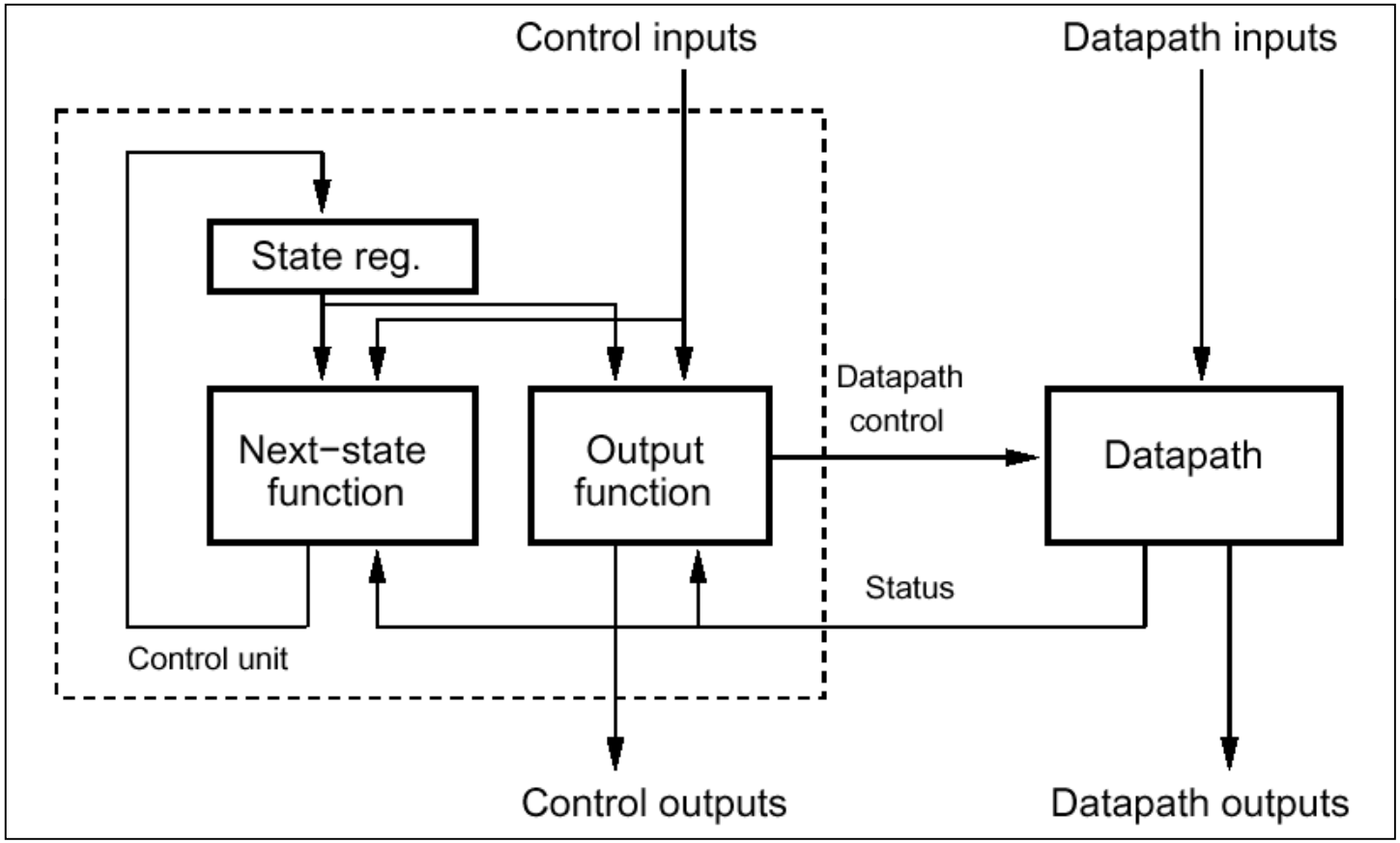
Δεν χρειάζεται κάθε μία κατάσταση του μετρητή να είναι και κατάσταση της μηχανής.

Μοντελοποιούμε την λειτουργία του DataPath δηλ. αύξηση ή μηδένιση και την ελέγχουμε με το FSM

Το κόστος σταθερό για μέτρηση έως το 2 ή μεγαλύτερο αριθμο πχ. 15.678

FSM with DataPath

- Τα μοντέλα FSMD περιγράφουν συστήματα σε RTL.
- Το DataPath εκτελεί τις πράξεις, ενώ το καθεαυτό FSM ελέγχει το DataPath.

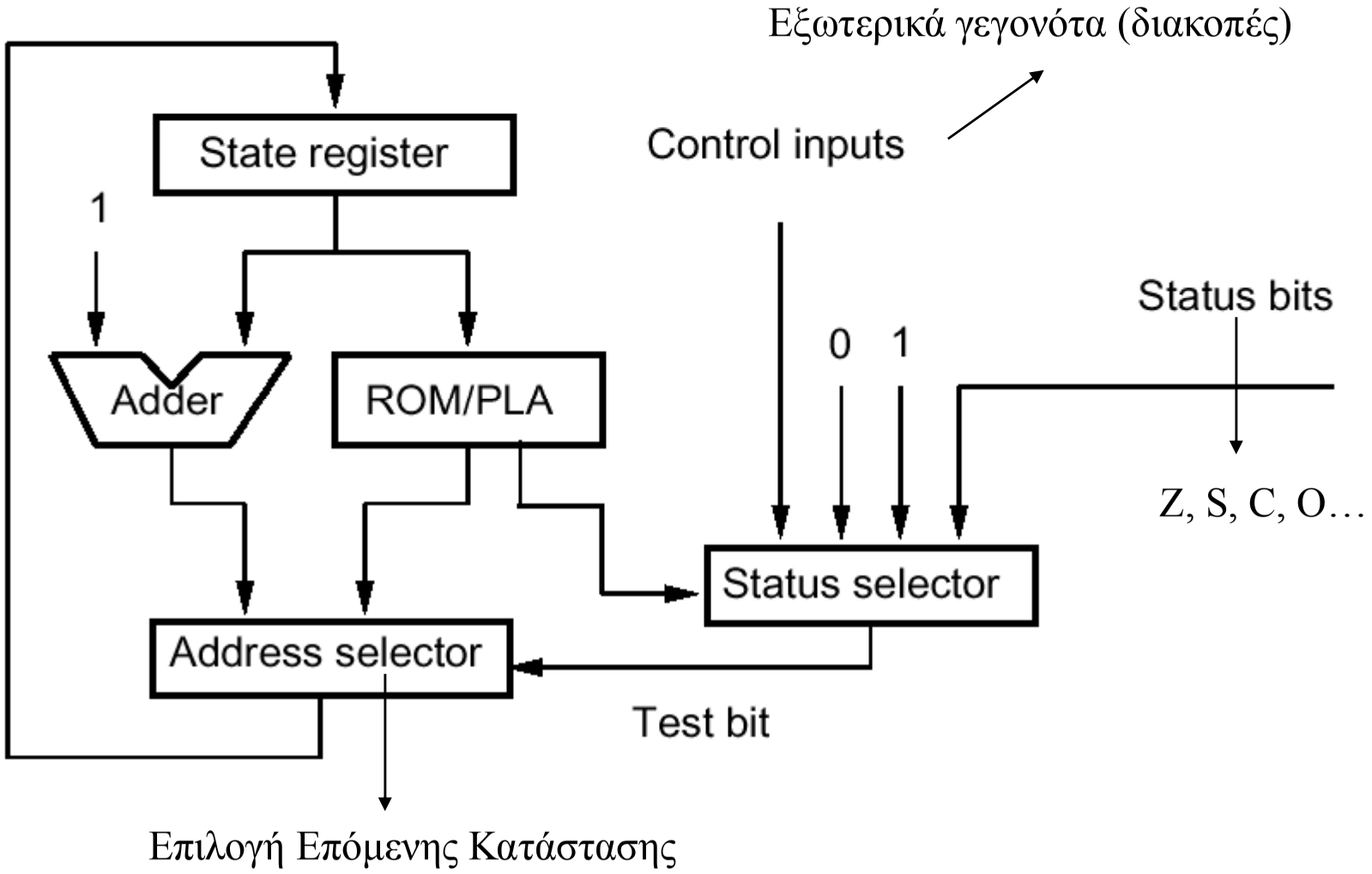


Παράδειγμα: μοντέλο επεξεργαστή σε FSMD

Το μοντέλο επεξεργαστή (στην ροή εκτέλεσης εντολών) μπορεί να χρησιμοποιηθεί για την υλοποίηση FSMD:

- Η επόμενη κατάσταση μπορεί να είναι η παρούσα αυξημένη κατά 1 ή μία κατάσταση διακλάδωσης αποθηκευμένη σε ROM ή PLA.
- Η επιλογή γίνεται από ειδικό bit ελέγχου που ελέγχει πολυπλέκτη.
- Το bit ελέγχου επιλέγεται και αυτό ανάμεσα από εισόδους ελέγχου και bit κατάστασης.
- Οι είσοδοι ελέγχου μπορεί να είναι κάποιες διακοπές.
- Τα bit κατάστασης είναι εσωτερικά (πχ status flags C, Z, N, O, …).
- Για κάθε bit ελέγχου αποθηκεύεται ο κατάλληλος κωδικός – κατάσταση στην ROM.

Παράδειγμα: μοντέλο επεξεργαστή σε FSMD



Σύνθεση για FSMDs

Διαδικασία σύνθεσης για τα FSMDs:

- Compilation
- Unit Selection (αριθμό και τύπο μονάδων αποθήκευσης, λειτουργίας, διασύνδεσης που θα χρησιμοποιηθούν)
- Storage Binding (ανάθεση μεταβλητών σε αποθηκευτικά στοιχεία / διαμοίραση ανάλογα με χρόνο ζωής)
- Unit Binding (ανάθεση λειτουργιών σε μονάδες – λειτουργίες σε διαφορετικές καταστάσεις είναι ανεξάρτητες και εκτελούνται από κοινές μονάδες)
- Interconnection Binding (ανάθεση μονάδων διασύνδεσης)
- Control Definition (δημιουργία Boolean εκφράσεων για κάθε σήμα ελέγχου)
- Control Unit Synthesis
- Functional Unit Synthesis

Αρχιτεκτονική Συστήματος

- Σε επίπεδο συστήματος κάθε FSMD αντιστοιχεί σε μία διαδικασία (process).
- Κάθε process περιγράφεται από δομές ελέγχου (loop, if, case, …).
- Κάθε σύστημα αποτελείται από συνεργαζόμενες processes και άρα από συνεργαζόμενα FSMDs.
- Απαιτείται επικοινωνία κυρίως μεταξύ των μηχανών καταστάσεων.
- Η επικοινωνία μπορεί να είναι σύγχρονη ή ασύγχρονη.

modulo-3 divider

1-state FSMD

```
Loop forever
  if count=1
    then
      if x=2
        then
          begin
            x=0
            y=1
          end
        else
          begin
            x=x+1
            y=0
          end
      endif
    else
      begin
        x=0
        y=0
      end
  endif
endloop
```

Πρωτόκολλο HandShake

FSM1: Request=1 (req. state)

⇓

FSM2: Acknowledge=1 (ack state)

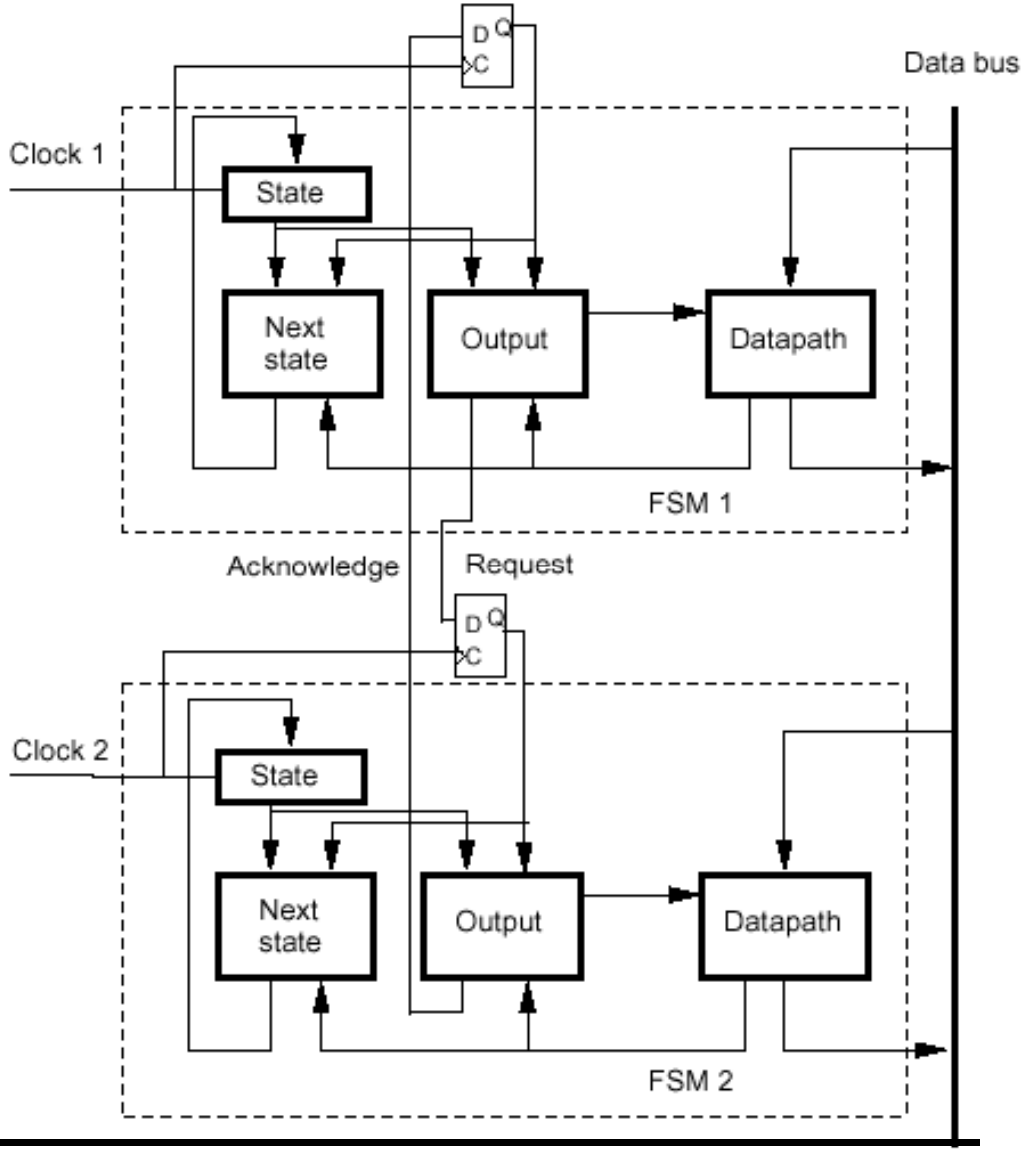
Χρήση BUS ⇓ *από FSM2*

FSM1: Request=0

⇓

FSM2: Acknowledge=0

Με όμοιο τρόπο (DataReady, DataReceived) μπορεί να γίνει η χρήση του Bus για επικοινωνία



Πρωτόκολλο HandShake

Ενδιάμεσα μπορεί να χρησιμοποιηθεί το bus.

Όταν ο ρυθμός επικοινωνίας είναι γνωστός το πρωτόκολλο απλοποιείται.

Παράδειγμα: Προσπέλαση μνήμης

1. Αίτηση στην μνήμης (Request)
2. Ανάγνωση δεδομένων σε γνωστό χρόνο.

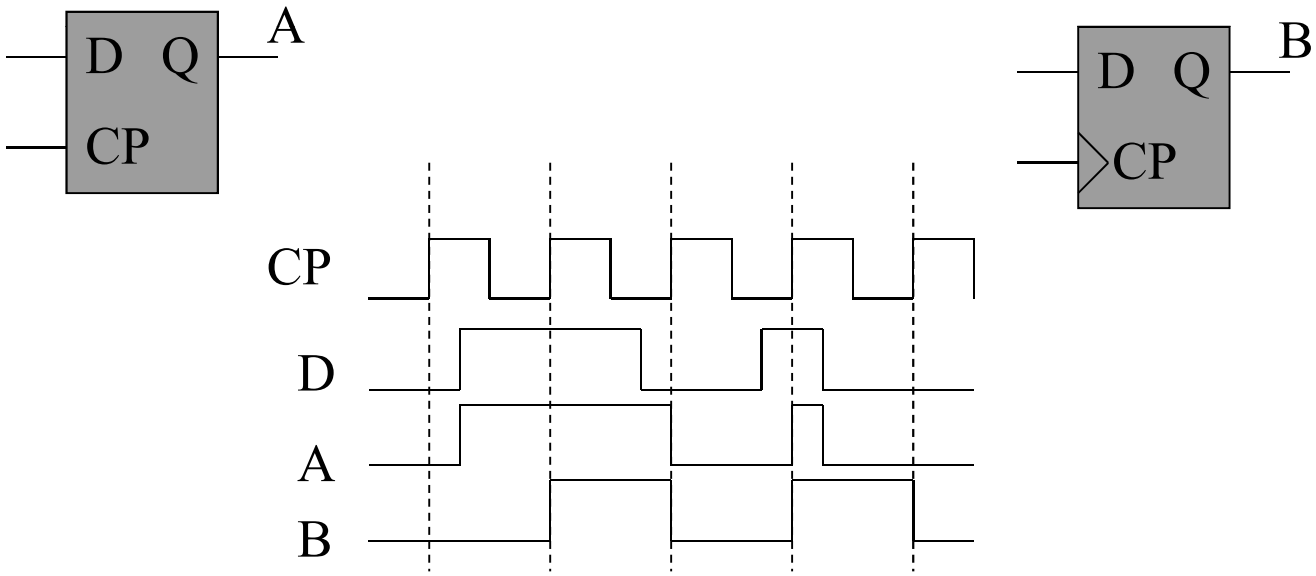
Αρχιτεκτονική Συστήματος

Διαδικασία σύνθεσης σε επίπεδο συστήματος:

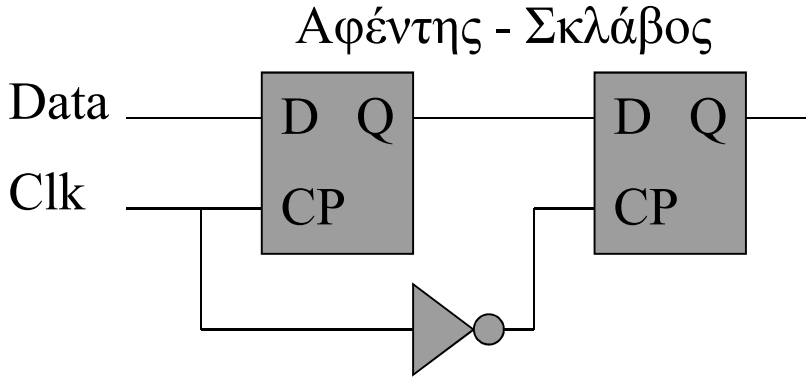
- Compilation (μετατροπή της περιγραφής σε Control Data Flow Graph – CDFG που δείχνει εξαρτήσεις δεδομένων και ελέγχου).
- Partitioning (διαίρεση σε chips και υποομάδες/chip, κάθε υποομάδα υλοποιείται με μια FSMD δομή)
- Interface Synthesis (καθορισμός επικοινωνίας)
- Scheduling (διαιρεί το CDFG σε καταστάσεις ή βήματα ελέγχου)
- FSMD Synthesis

Συγχρονισμός Συστήματος (clocking)

- Στα σύγχρονα συστήματα απαιτείται ρολόϊ.
- Το ρολόϊ συγχρονίζει την αποθήκευση των δεδομένων στα στοιχεία μνήμης (flip flops / latches).
- Έχουμε ακμοπυροδότητα flip flops και ευαίσθητα σε ακμή latches.

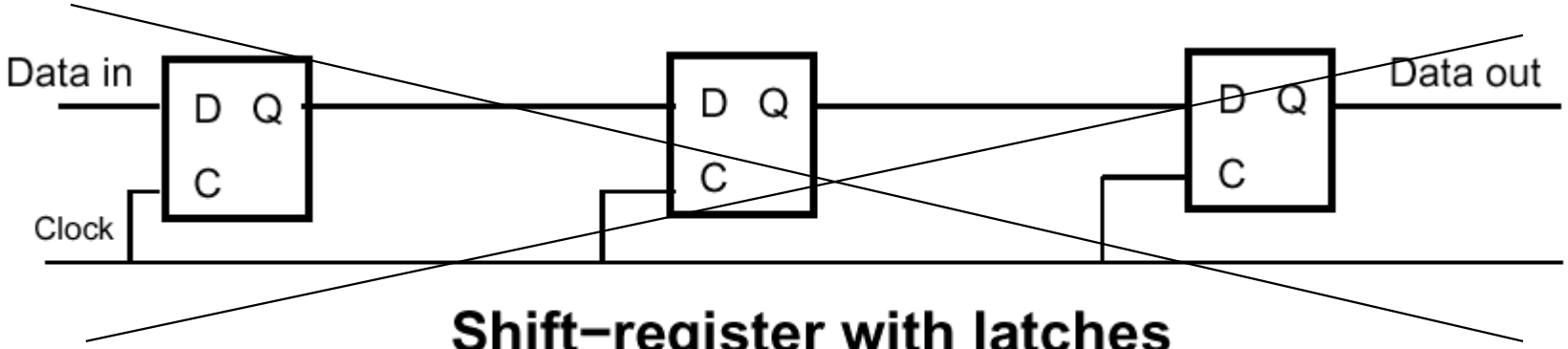


Συγχρονισμός Συστήματος (clocking)

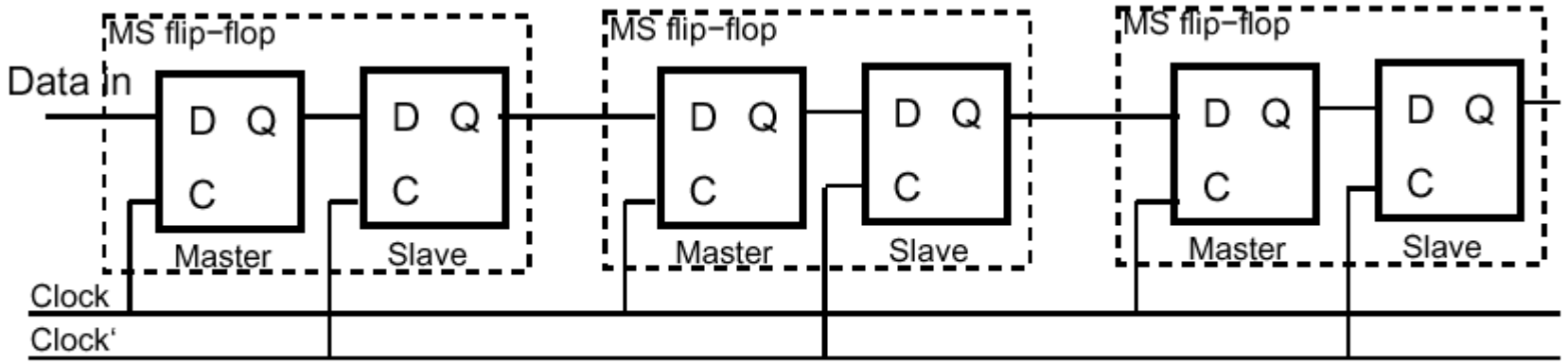


- Η αποθήκευση σε όλα τα στοιχεία γίνεται ταυτόχρονα σε ένα σύστημα (συγχρονισμός).
- Υπάρχουν περιπτώσεις που προτιμάται το ένα είδος συγχρονισμού έναντι του άλλου.
- Υπάρχουν περιπτώσεις που δεν μπορεί να χρησιμοποιηθεί ο συγχρονισμός ευαισθησίας σε ακμές (καταχωρητές ολίσθησης).

Συγχρονισμός Συστήματος (clocking)



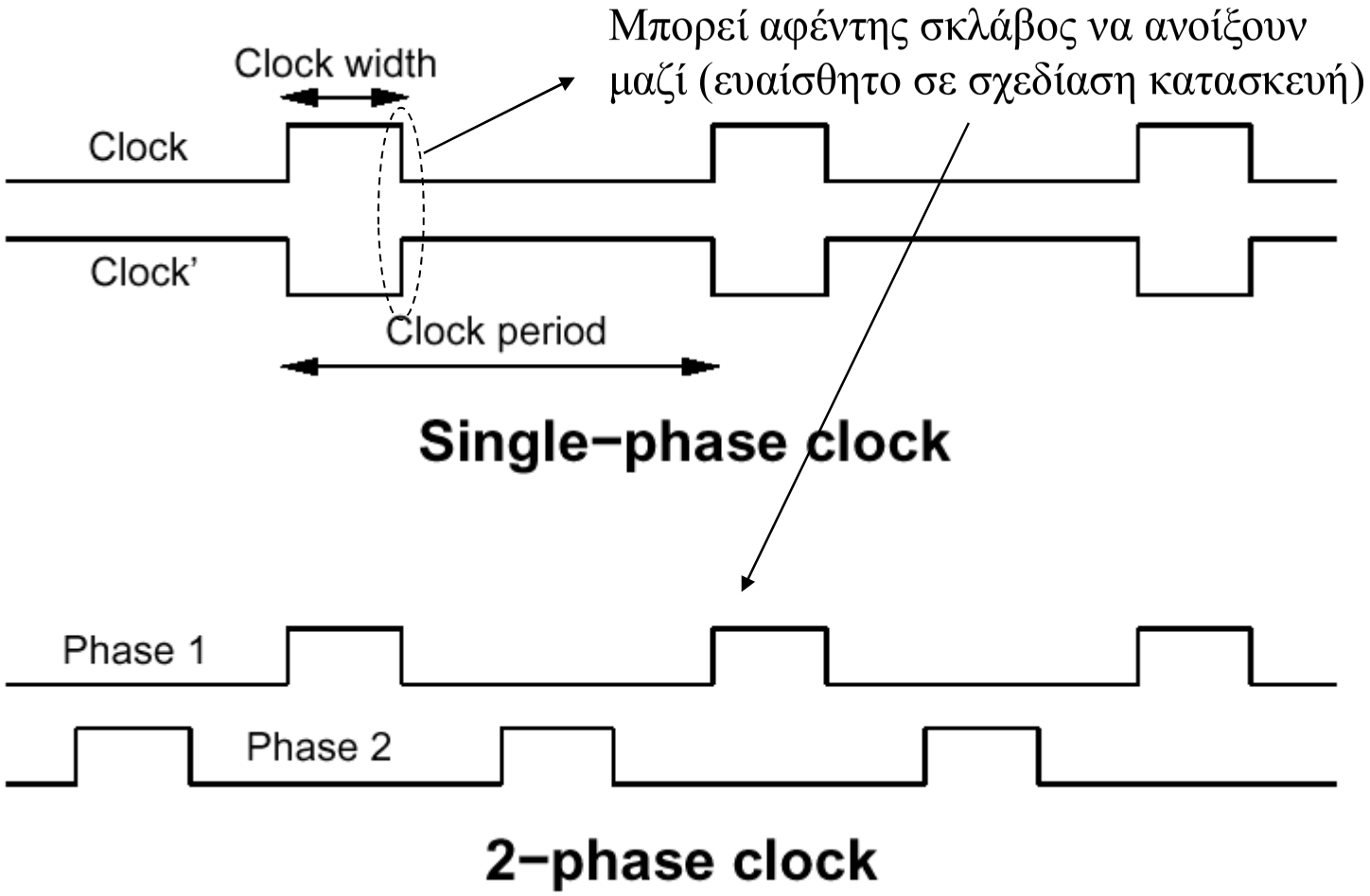
Shift−register with latches



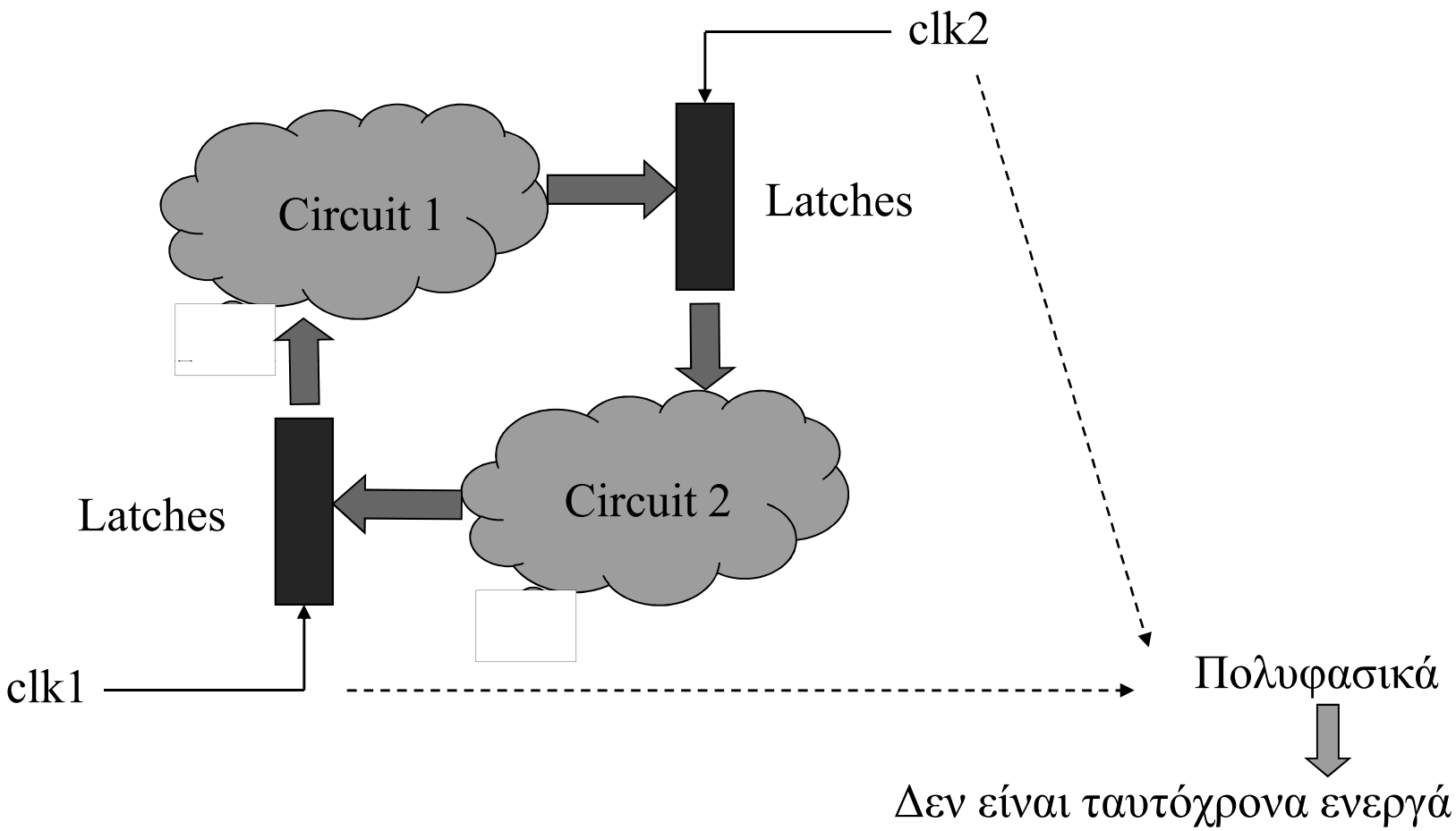
Shift−register with MS flip−flops

Συγχρονισμός Συστήματος (clocking)

<u>Πρόβλημα</u>:

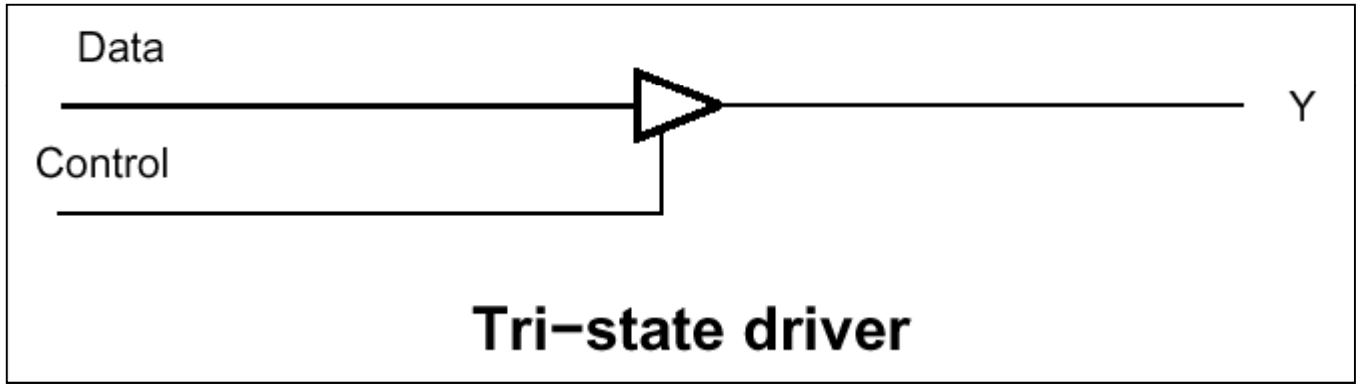


Πολυφασικά ρολόγια



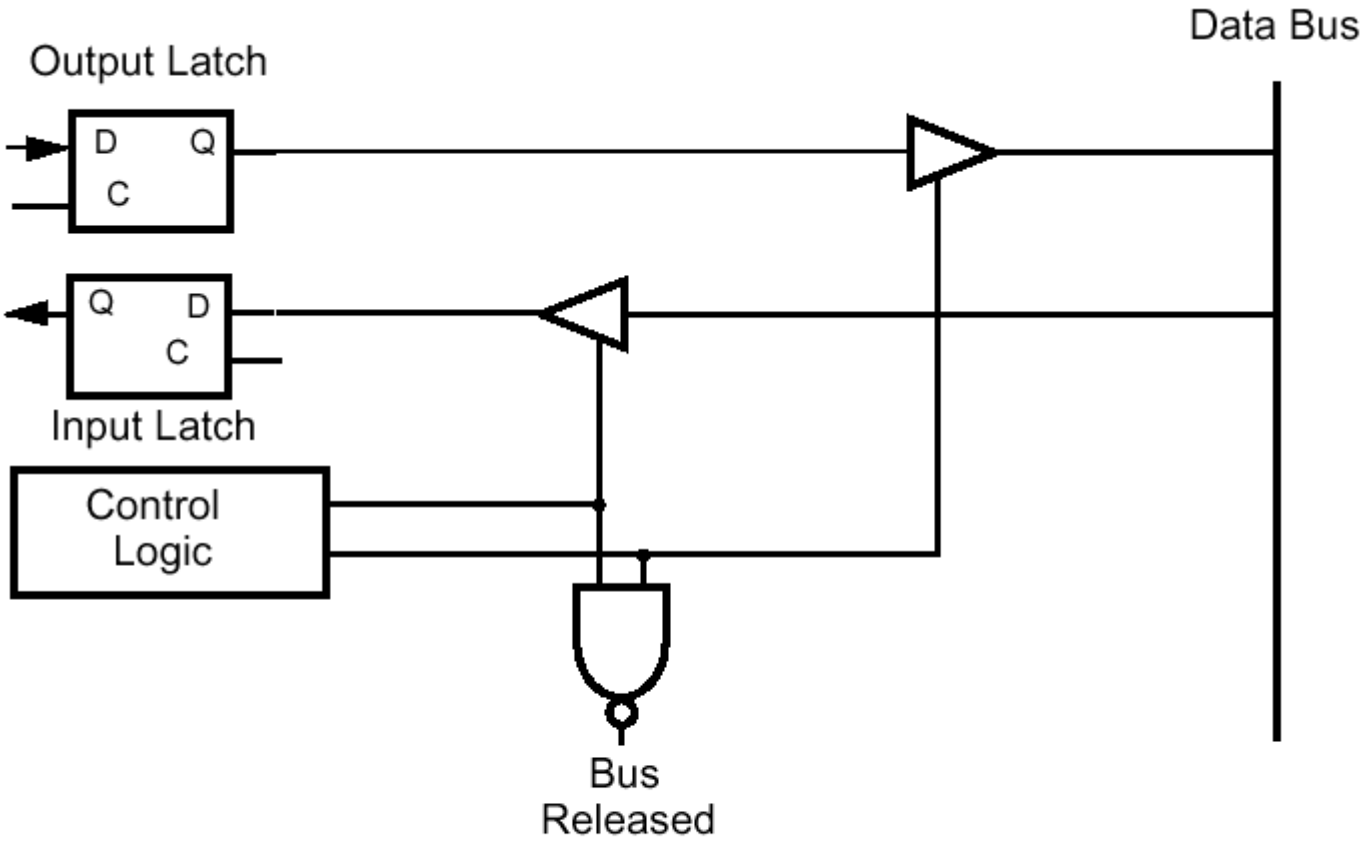
Δίαυλοι

- Είναι η πιο διαδεδομένη μονάδα διασύνδεσης.
- Έχουν πολύ απλό Layout
- Μπορούμε να διασυνδέουμε μονάδες πολύ εύκολα.
- Το βασικό δομικό στοιχείο είναι ο οδηγός τριών καταστάσεων.



Ελεγκτής Διαύλου: FSMD
- Τηρεί προτεραιότητες
- Παρέχει την χρήση του Bus
- Δέχεται αιτήσεις χρήσης

Δίαυλοι

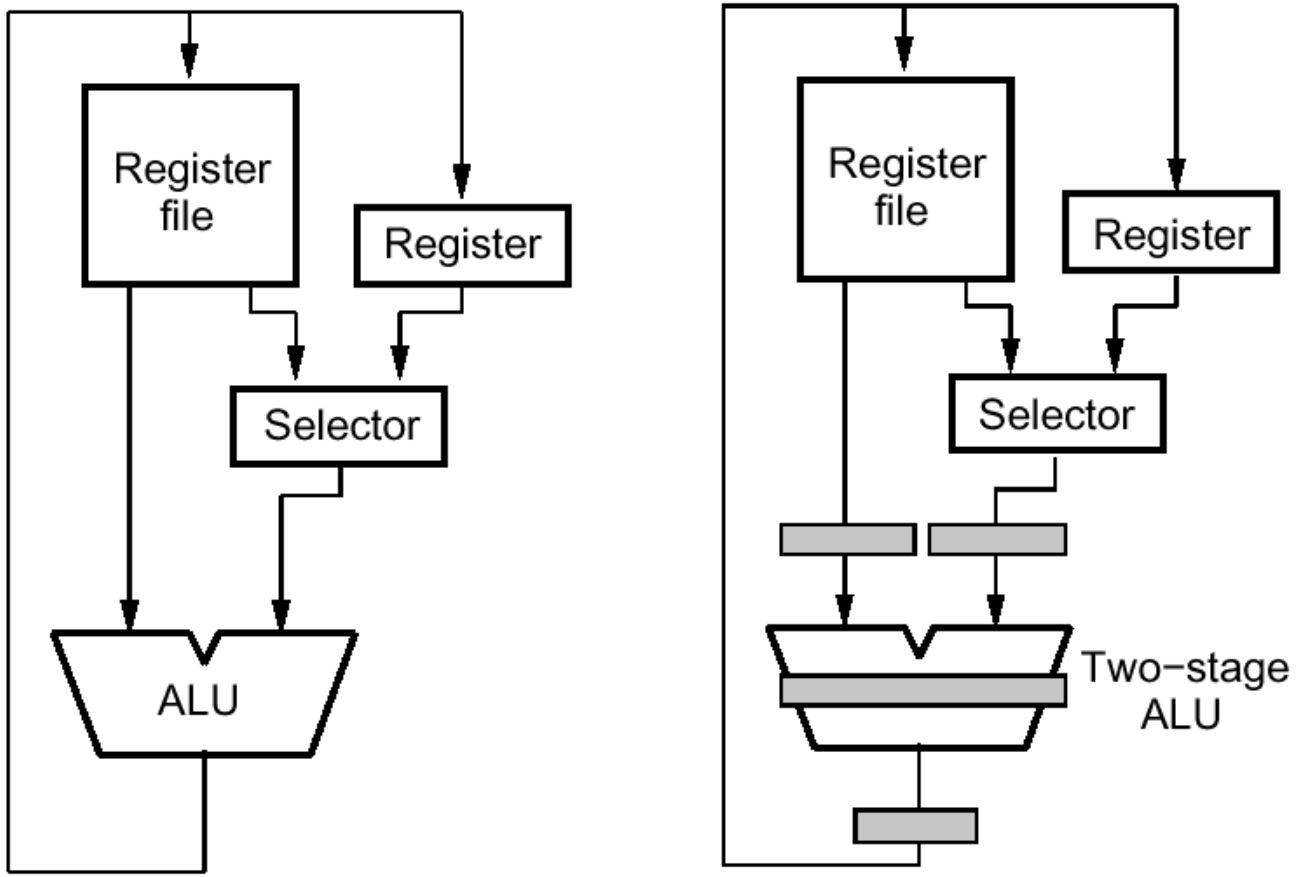


Bus interface

Pipelining

- Αυξάνει το ποσοστό χρήσης λειτουργικών μονάδων.
- Μειώνει τον χρόνο κύκλου.
- Εισάγει περιορισμούς καθώς το αποτέλεσμα μίας πράξης δεν μπορεί να είναι διαθέσιμο για τις επόμενες πράξεις (μέχρι να βγει από το pipeline).
- Η μονάδα ελέγχου είναι πιο περίπλοκη.
- Διαιρείται η μονάδα σε βαθμίδες και πριν από κάθε βαθμίδα τοποθετείται ένα latch.
- Η πιο αργή βαθμίδα καθορίζει την συχνότητα ρολογιού.

Pipelined Functional Unit

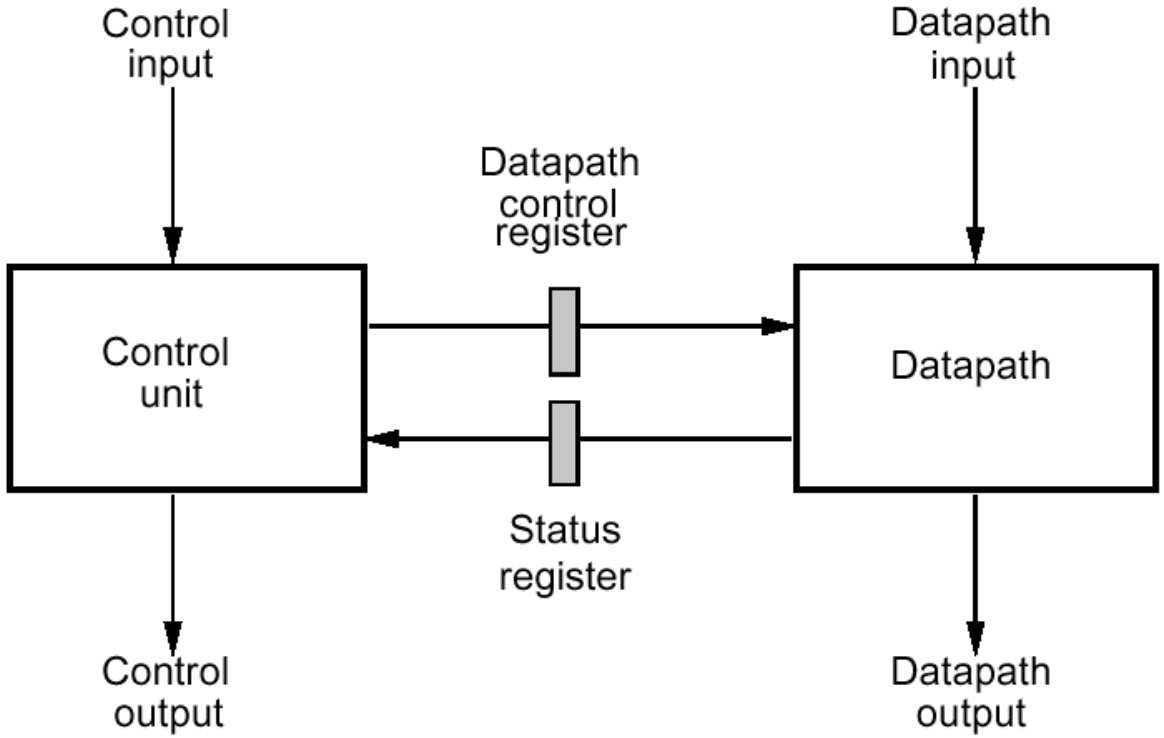


Non−pipelined datapath

Pipelined datapath with 2−stage adder

Pipelined Control Unit

Στις pipelined μονάδες ελέγχου διαιρείται σε βαθμίδες α) η λογική επόμενης κατάστασης και β) η λογική εξόδου



Pipelined control unit

Συνήθως μπαίνουν latches μόνο στις εισόδους και εξόδους της λογικής ελέγχου καθώς δεν υπάρχουν πολλά επίπεδα πυλών σε μονάδες ελέγχου.