

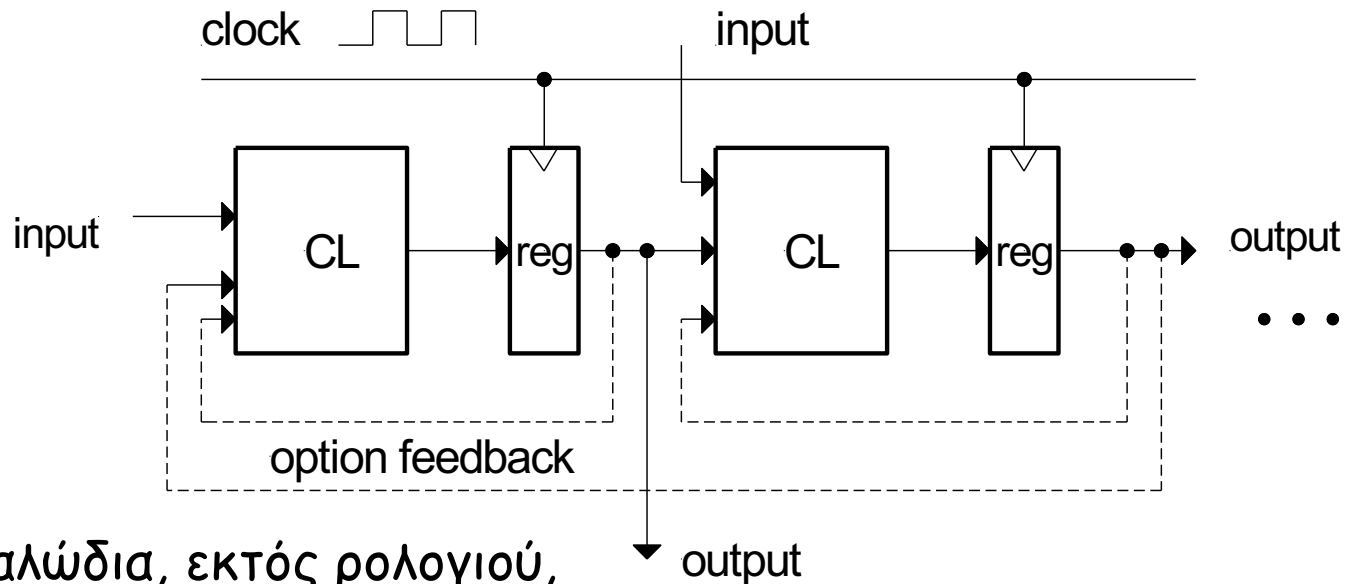
HY220

Εργαστήριο Ψηφιακών Κυκλωμάτων

Χειμερινό Εξάμηνο
2018-2019

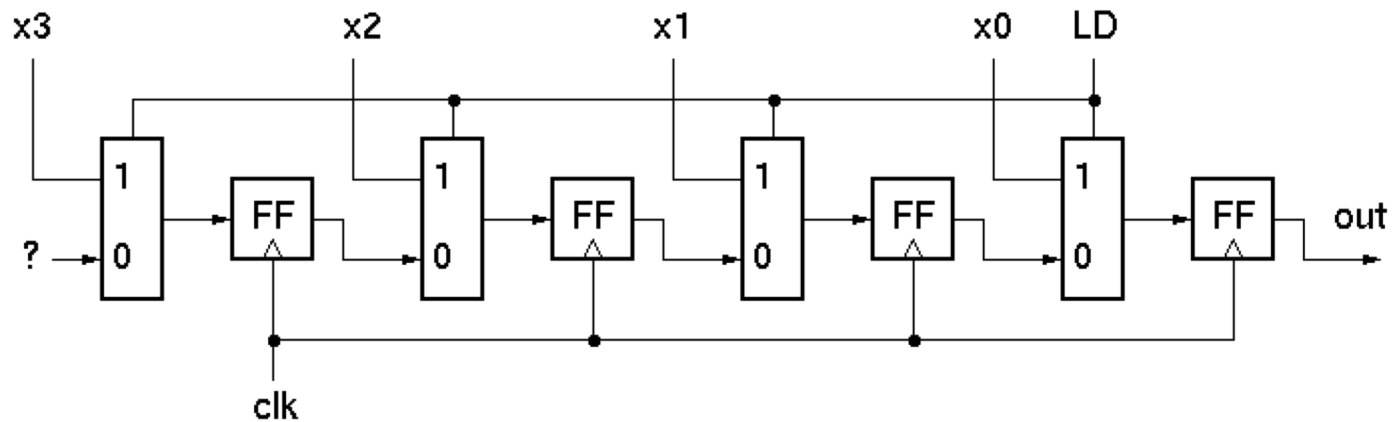
Χρονισμός Σύγχρονων Κυκλωμάτων,
Καταχωρητές και Μανταλωτές

Γενικό Μοντέλο Σύγχρονων Κυκλωμάτων



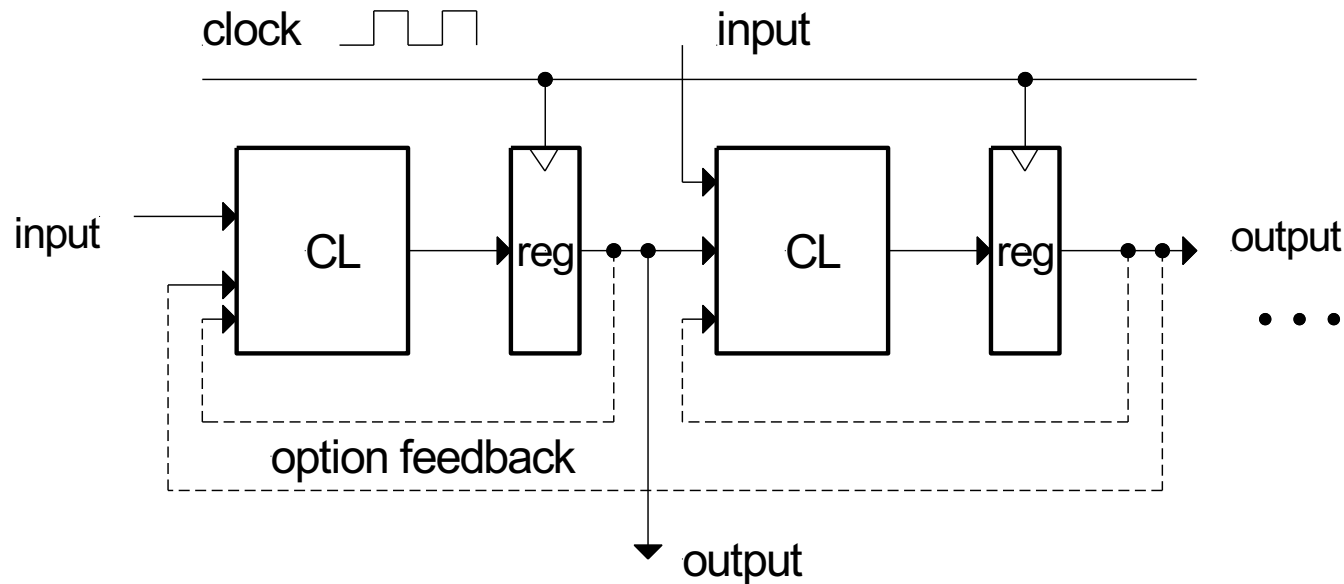
- Τα καλώδια, εκτός ρολογιού, μπορούν να έχουν πλάτος πολλά bits.
- Καταχωρητές (registers)
 - συλλογή από flip-flops
- Ρολόι
 - Διανέμεται στα flip-flops
- Συνδυαστική Λογική (Combinational Logic - CL)
 - Δεν έχουν εσωτερική κατάσταση
 - Έξοδοι είναι συναρτήσεις των εισόδων
- Προαιρετικά feedbacks

Παράδειγμα κυκλώματος



- Parallel to Serial Converter
- Όλα τα μονοπάτια είναι ενός bit
- Οι καταχωρητές είναι απλά flip-flops
- Η συνδυαστική λογική είναι οι πολυπλέκτες
- Δεν υπάρχει feedback

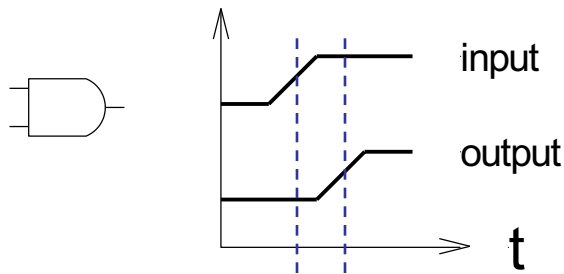
Γενικό Μοντέλο Σύγχρονων Κυκλωμάτων



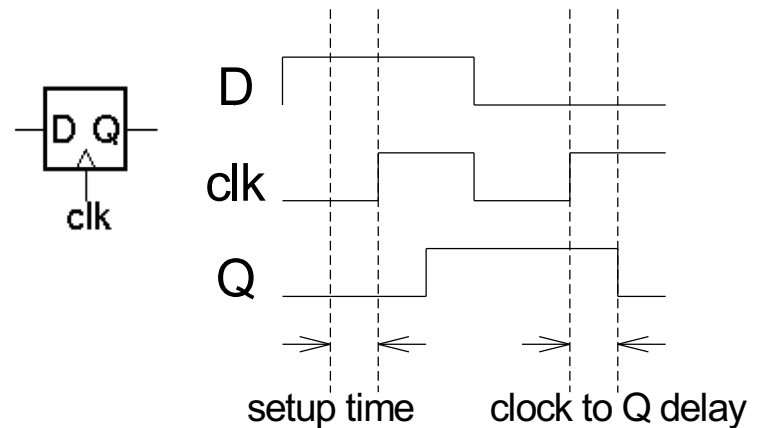
- Πώς μετράμε επιδόσεις ;
 - Λειτουργίες / *sec* ;
 - Κύκλοι / *sec* ;
- Τι περιορίζει τον κύκλο ρολογιού ;
- Τι συμβαίνει αν αυξήσουμε τη συχνότητα του ρολογιού;

Περιορισμοί στη συχνότητα του ρολογιού

1 Καθυστερήσεις πυλών

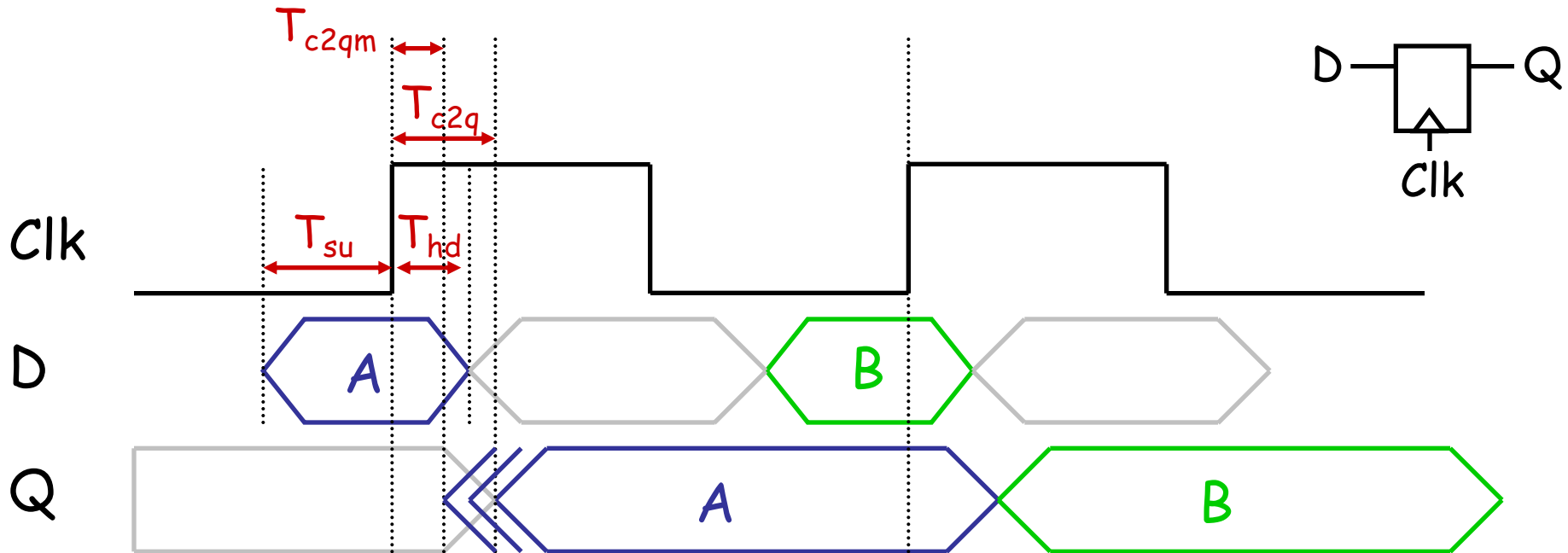


2 Καθυστερήσεις flip-flops



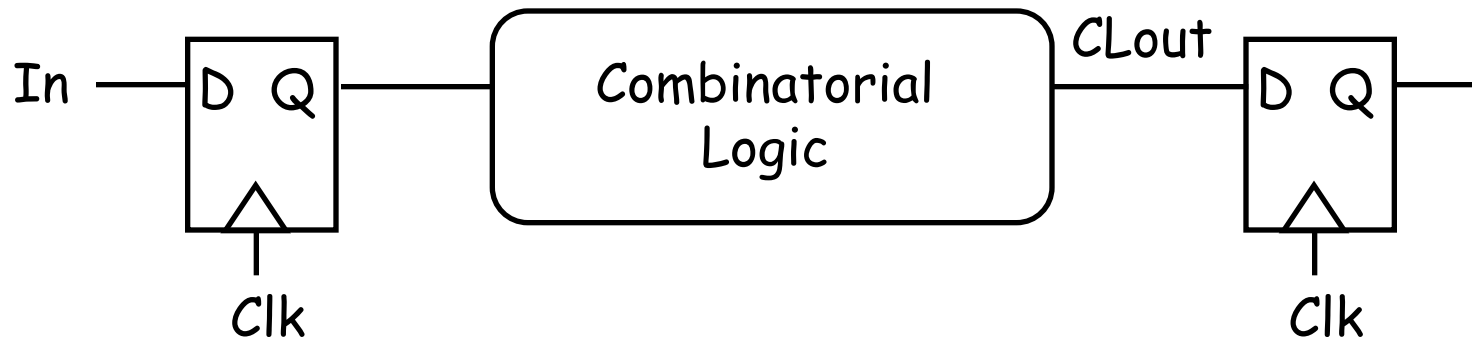
- Τι πρέπει να συμβεί σε ένα κύκλο του ρολογιού για να έχουμε σωστή λειτουργία ;
- Θεωρώντας ότι το ρολόι διανέμεται τέλεια (όλα τα flip-flops βλέπουν την ακμή ταυτόχρονα):
 - Όλα τα σήματα πρέπει να είναι έτοιμα (setup) πριν την θετική ακμή του ρολογιού

Flip-Flop: Χρονικές Παράμετροι



- Η είσοδος D πρέπει να μείνει σταθερή τουλάχιστον για χρόνο T_{su} (setup time) πριν την ακμή του ρολογιού και τουλάχιστον T_{hd} (hold time) μετά την ακμή.
 - Ένα παράθυρο χρόνου γύρω από την ακμή του ρολογιού για το οποίο η είσοδος πρέπει να μείνει σταθερή
- Η έξοδος Q αλλάζει λίγο μετά την ακμή του ρολογιού
 - T_{c2q} είναι ο χρόνος καθυστέρησης από την ακμή στην έξοδο (propagation delay)
 - T_{c2qm} είναι ο ελάχιστος χρόνος καθυστέρησης από την ακμή στην έξοδο (αρχίζουν να αλλάζουν τα δεδομένα - contamination delay)

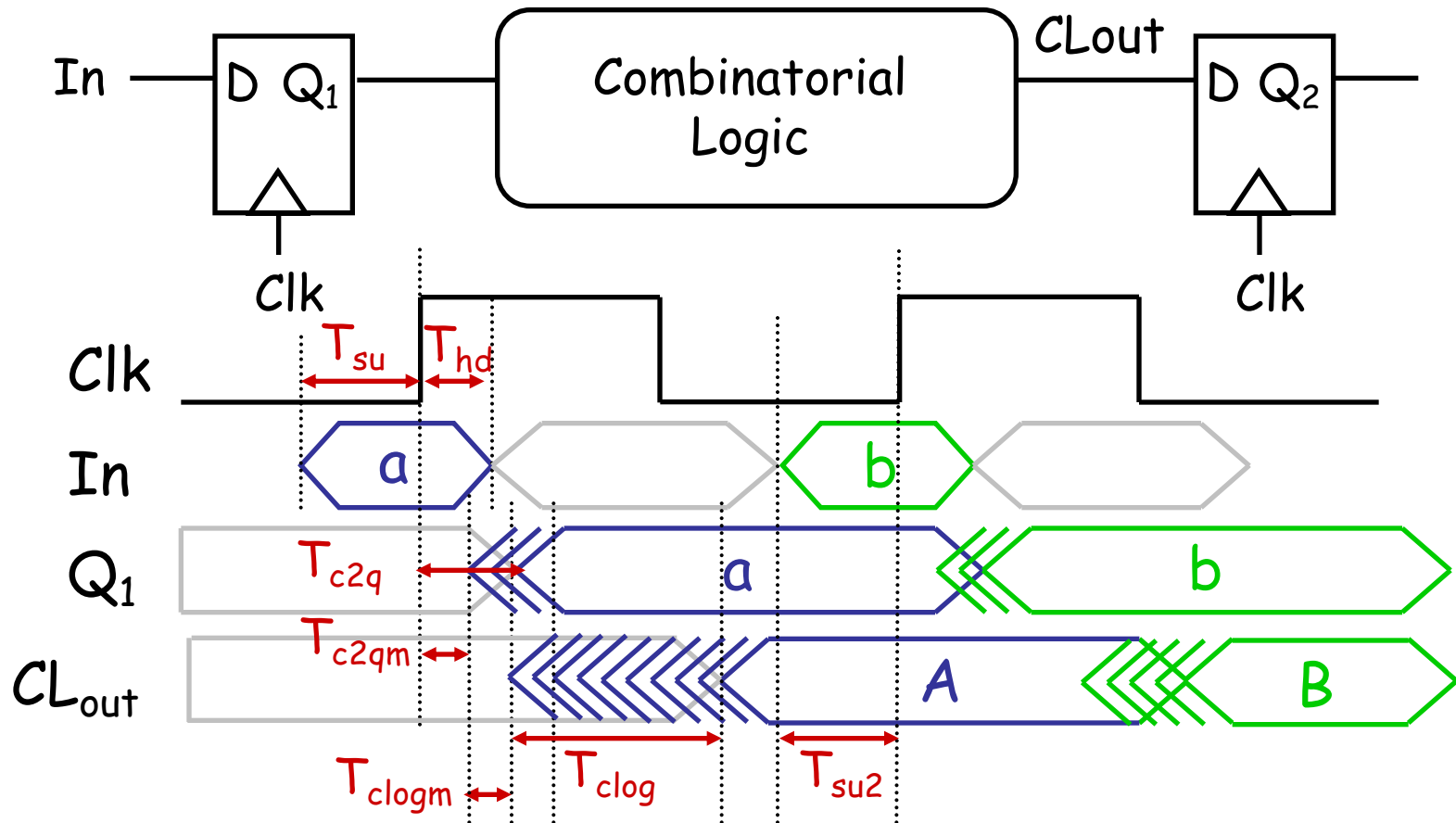
Σύγχρονο Κύκλωμα: Χρονικές Παράμετροι



- Χρονικές παράμετροι καταχωρητών
 - T_{clk} : Περίοδος Ρολογιού
 - T_{su} : Setup time
 - T_{hd} : Hold time
 - T_{c2q} : Clock to Q (worst)
 - T_{c2qm} : Clock to Q (min)

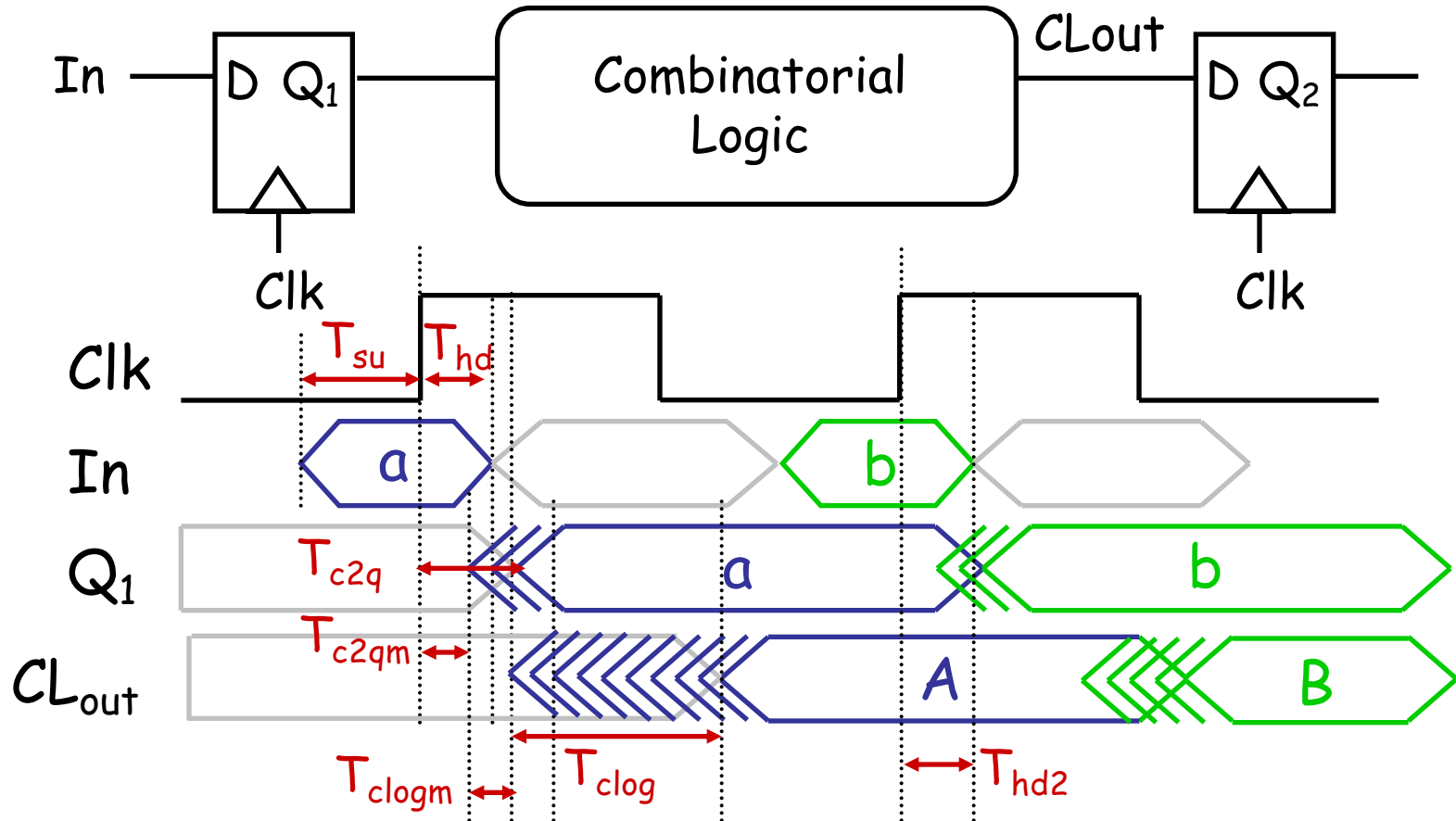
- Χρονικές παράμετροι συνδυαστικής λογικής
 - T_{clog} : Καθυστέρηση συνδυαστικής λογικής (max - propagation delay)
 - T_{clogm} : Ελάχιστη καθυστέρηση συνδυαστικής λογικής (min-contamination)

Χρονισμός Σύγχρονων Κυκλωμάτων: Ελάχιστη περίοδος



$$T_{clk} \geq T_{c2q} + T_{clog} + T_{su}$$

Χρονισμός Σύγχρονων Κυκλωμάτων: Ελάχιστη καθυστέρηση

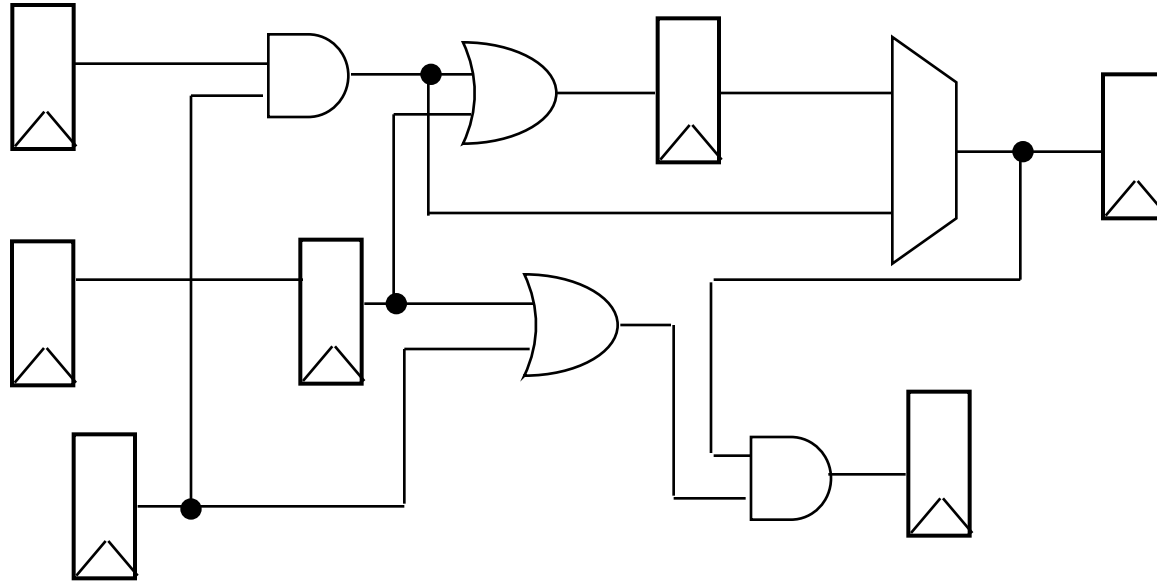


$$T_{clogm} \geq T_{hd} - T_{c2qm}$$

Χρονισμός Σύγχρονων Κυκλωμάτων

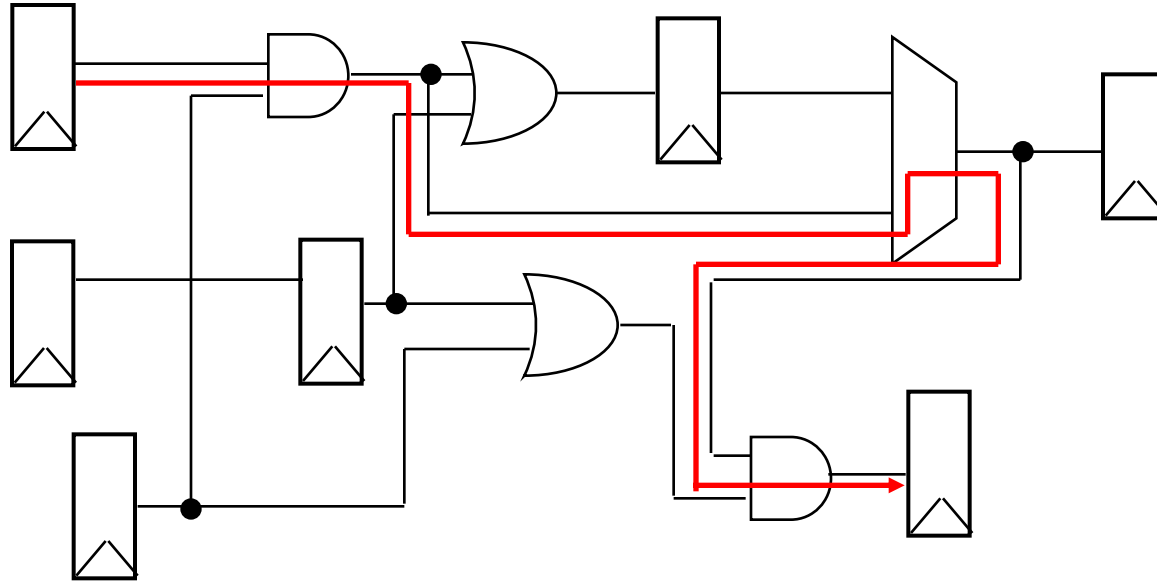
- Γενικά , για σωστή λειτουργία πρέπει για όλα τα μονοπάτια να ισχύει:
 - $T_{clk} \geq T_{c2q} + T_{clog} + T_{su}$
 - $T_{clogm} \geq T_{hd} - T_{c2qm}$ (ή $T_{hd} \leq T_{c2qm} + T_{clogm}$)
- Πώς βρίσκουμε όλα τα μονοπάτια ;
 - Από κάθε είσοδο ή έξοδο καταχωρητή σε κάθε είσοδο καταχωρητή ή έξοδο του κυκλώματος
 - Το πιο αργό μονοπάτι συνδυαστικής λογικής είναι αυτό που καθορίζει το T_{clog} (οπότε και την ελάχιστη περίοδο) και λέγεται **critical path**.
 - Ο εντοπισμός του **critical path** μας δίνει τη δυνατότητα να προσπαθήσουμε να απλοποιήσουμε την λογική του μονοπατιού και να πετύχουμε υψηλότερη συχνότητα λειτουργίας του κυκλώματος.

Παράδειγμα (1/6)



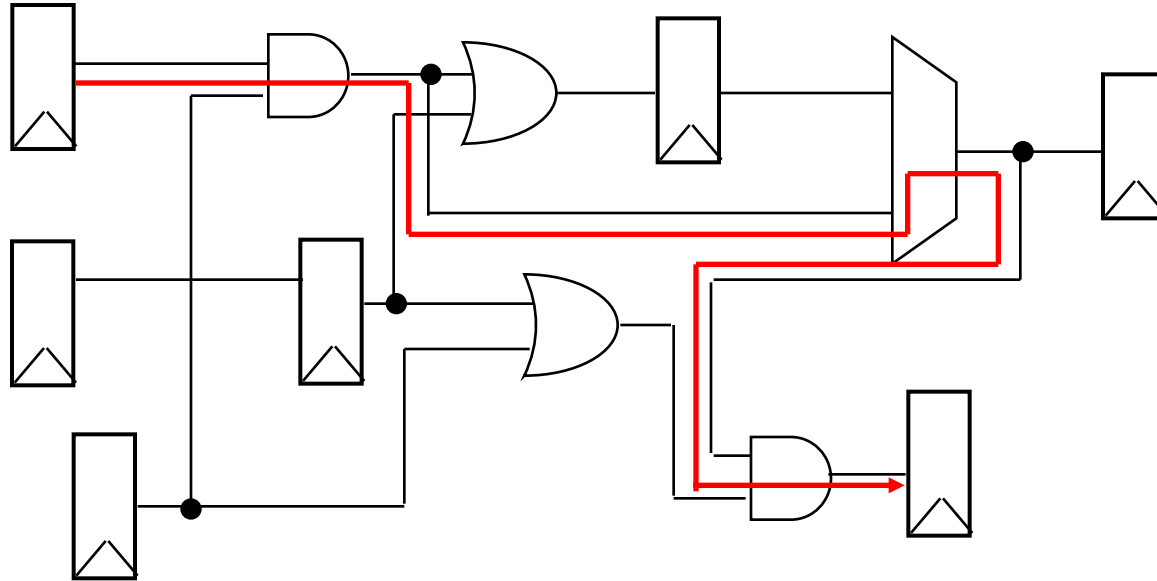
- $T_{\text{and}} = 2\text{ns}$, $T_{\text{or}} = 1\text{ns}$, $T_{\text{mux}} = 3\text{ns}$,
- $T_{c2q} = 0.5\text{ns}$, $T_{c2qm} = 0.2\text{ns}$, $T_{\text{su}} = 0.4\text{ns}$, $T_{\text{hd}} = 0.3\text{ns}$
- Ποιό είναι το **critical path**;
- Πόση είναι η ελάχιστη περίοδος ρολογιού;
- Καλύπτονται όλες οι συνθήκες χρονισμού;

Παράδειγμα (2/6)



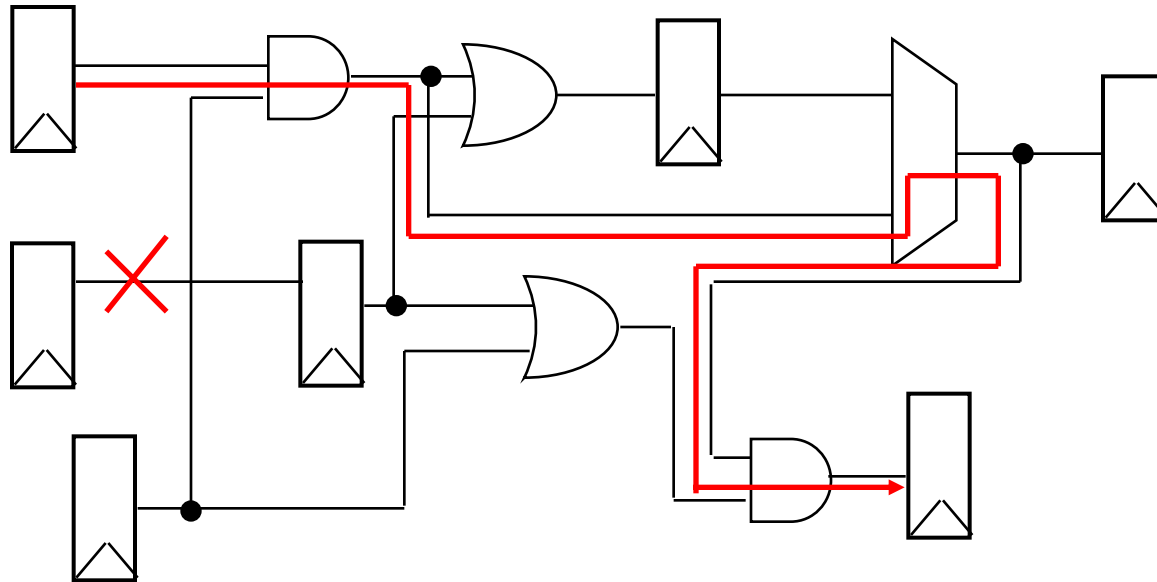
- $T_{\text{and}} = 2\text{ns}$, $T_{\text{or}} = 1\text{ns}$, $T_{\text{mux}} = 3\text{ns}$,
- $T_{c2q} = 0.5\text{ns}$, $T_{c2qm} = 0.2\text{ns}$, $T_{su} = 0.4\text{ns}$, $T_{hd} = 0.3\text{ns}$
- Ποιό είναι το **critical path**;

Παράδειγμα (3/6)



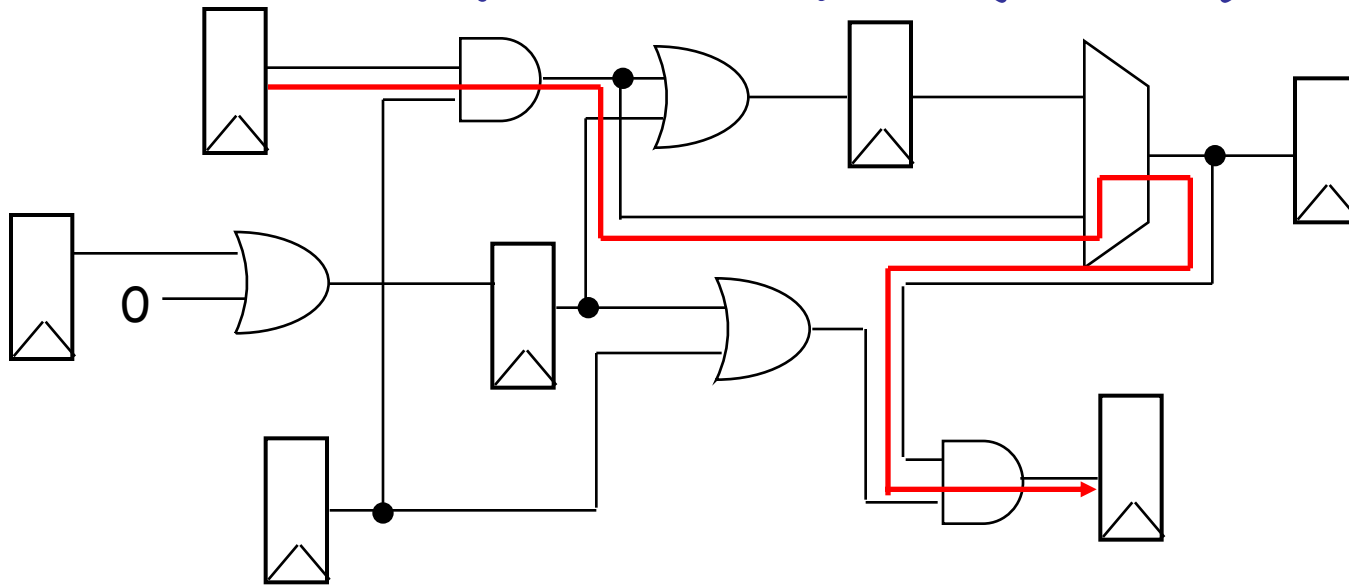
- $T_{\text{and}} = 2\text{ns}$, $T_{\text{or}} = 1\text{ns}$, $T_{\text{mux}} = 3\text{ns}$,
- $T_{c2q} = 0.5\text{ns}$, $T_{c2qm} = 0.2\text{ns}$, $T_{\text{su}} = 0.4\text{ns}$, $T_{\text{hd}} = 0.3\text{ns}$
- Πόση είναι η ελάχιστη περίοδος ρολογιού;
 - $T_{\text{min}} = T_{c2q} + T_{\text{and}} + T_{\text{mux}} + T_{\text{and}} + T_{\text{su}} = 7.9\text{ns}$

Παράδειγμα (4/6)



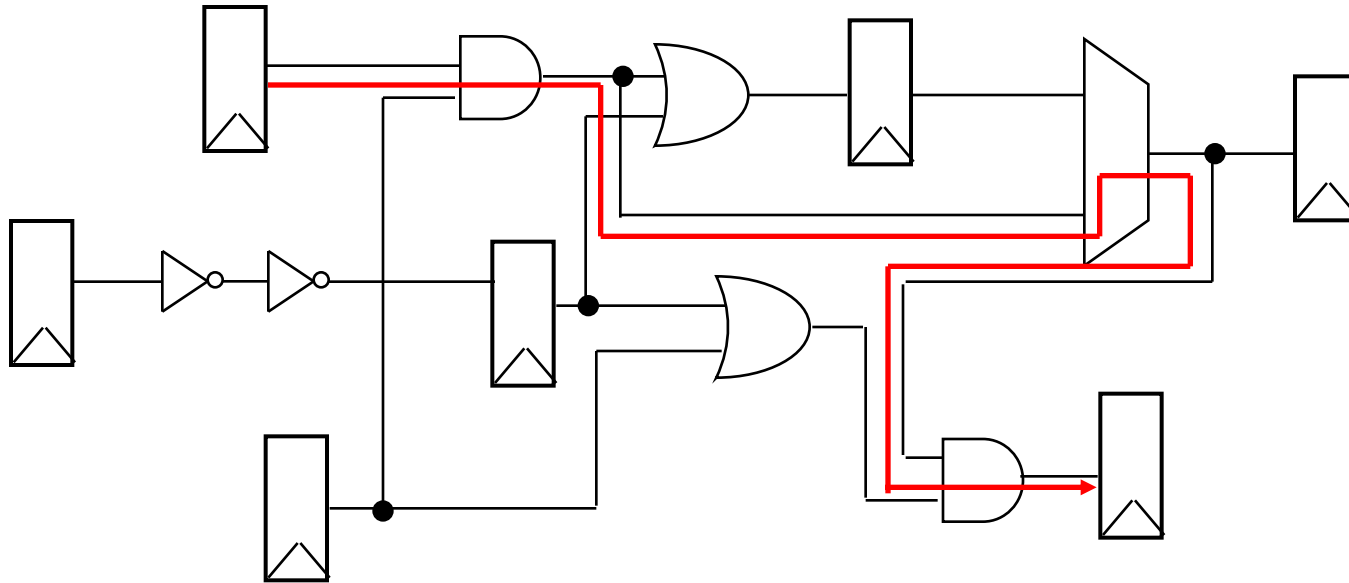
- $T_{\text{and}} = 2\text{ns}$, $T_{\text{or}} = 1\text{ns}$, $T_{\text{mux}} = 3\text{ns}$,
- $T_{c2q} = 0.5\text{ns}$, $T_{c2qm} = 0.2\text{ns}$, $T_{\text{su}} = 0.4\text{ns}$, $T_{\text{hd}} = 0.3\text{ns}$
- Καλύπτονται όλες οι συνθήκες χρονισμού;
 - **ΟΧΙ !!!** Έχουμε $T_{\text{clogm}} = 0\text{ns}$ και $T_{c2qm} = 0.2\text{ns}$
 - Πρέπει $T_{\text{hd}} \leq T_{c2qm} + T_{\text{clogm}}$
 - Και τώρα τι κάνουμε ;

Παράδειγμα (5/6)



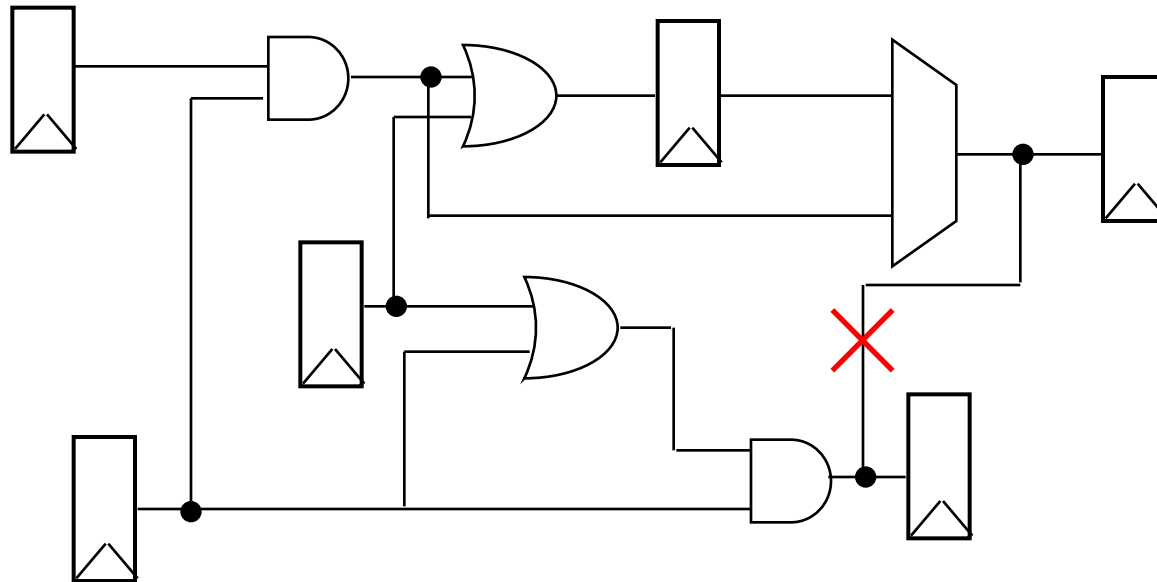
- $T_{\text{and}} = 2\text{ns}$, $T_{\text{or}} = 1\text{ns}$, $T_{\text{mux}} = 3\text{ns}$,
- $T_{\text{c2q}} = 0.5\text{ns}$, $T_{\text{c2qm}} = 0.2\text{ns}$, $T_{\text{su}} = 0.4\text{ns}$, $T_{\text{hd}} = 0.3\text{ns}$
- Καλύπτονται όλες οι συνθήκες χρονισμού:
 - Πρέπει $T_{\text{hd}} \leq T_{\text{c2qm}} + T_{\text{clogm}}$
 - Προσθέτουμε μια πύλη με $T_{\text{or}} = 1\text{ns}$ (αρκεί;)
 - Γενικά όχι! T_{orm} (min-contam.);
 - Έστω $T_{\text{orm}} = T_{\text{or}}$ και γενικά για όλες τις πύλες! (απλοποίηση)
 - Τώρα OK!!! Έχουμε $T_{\text{clogm}} = 1\text{ns}$ και $T_{\text{c2qm}} = 0.2\text{ns}$

Παράδειγμα (6/6)



- $T_{\text{and}} = 2\text{ns}$, $T_{\text{or}} = 1\text{ns}$, $T_{\text{mux}} = 3\text{ns}$,
- $T_{c2q} = 0.5\text{ns}$, $T_{c2qm} = 0.2\text{ns}$, $T_{\text{su}} = 0.4\text{ns}$, $T_{\text{hd}} = 0.3\text{ns}$
- Καλύπτονται όλες οι συνθήκες χρονισμού;
 - Πρέπει $T_{\text{hd}} \leq T_{c2qm} + T_{\text{clogm}}$
 - Συνήθως βάζουμε 2 αντιστροφείς (έστω $T_{\text{invm}} = 0.3\text{ns}$)
 - **OK !!!** Έχουμε $T_{\text{clogm}} = 0.6\text{ns}$ και $T_{c2qm} = 0.2\text{ns}$

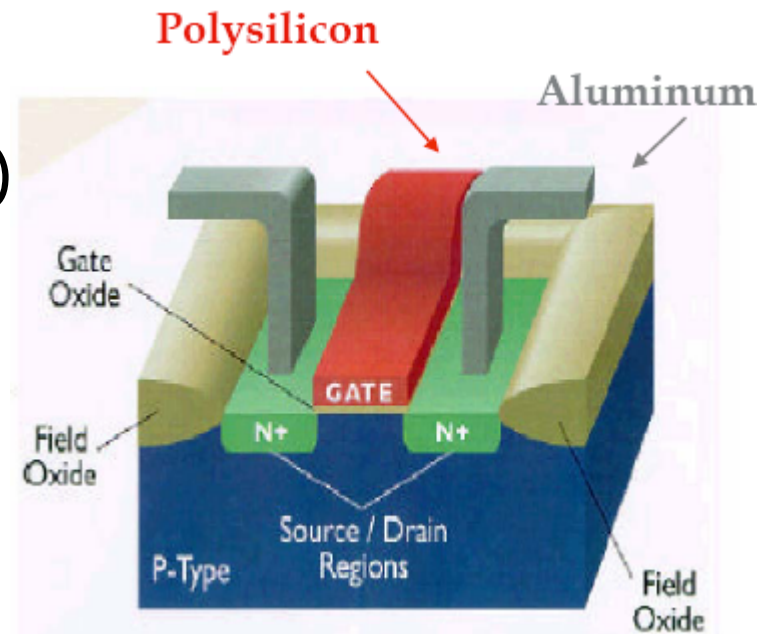
Παράδειγμα κυκλώματος με λάθος



Πού είναι το λάθος ;

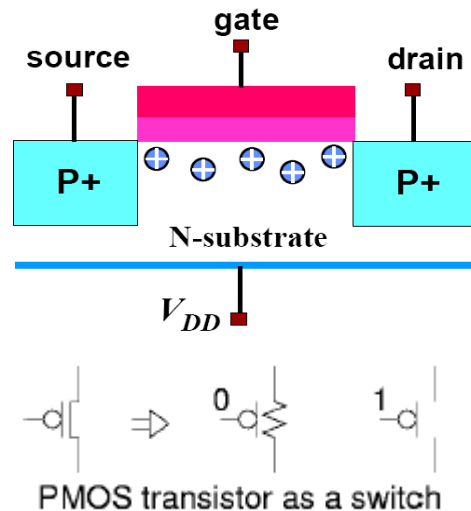
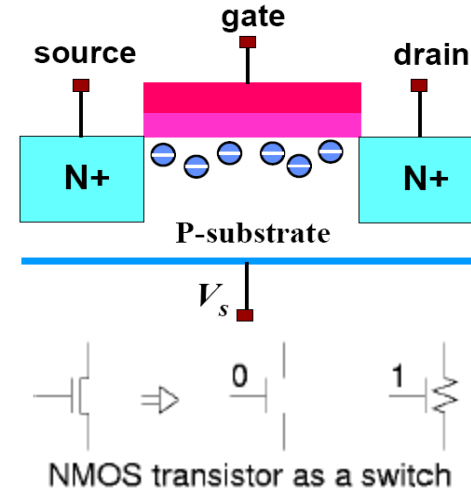
Πύλες και τεχνολογία (1/6)

- Οι πύλες στα ολοκληρωμένα κυκλώματα υλοποιούνται σε τεχνολογία CMOS (Complementary MOS)
 - Βάση της τεχνολογίας τα transistors τύπου MOSFET (metal oxide semiconductor field effect transistors - transistor επίδρασης πεδίου τύπου μέταλλο - οξειδίο - ημιαγωγός)
 - Gate (του transistor)
 - Source
 - Drain
 - Channel



Πύλες και τεχνολογία (2/6)

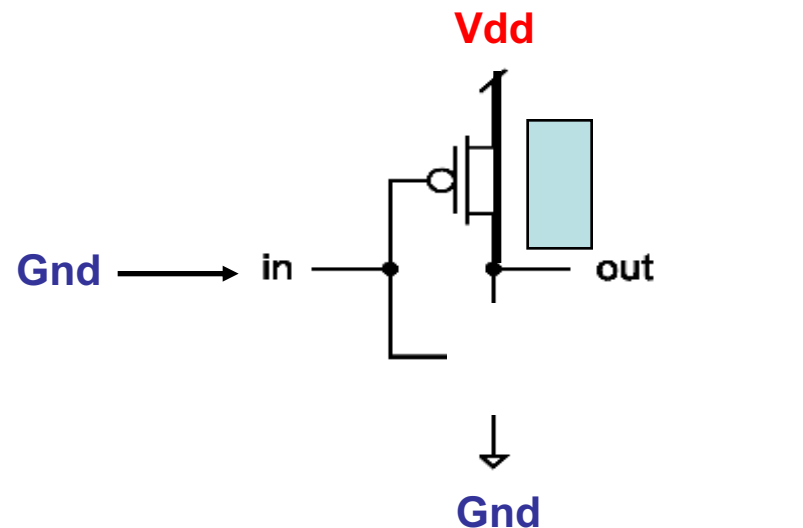
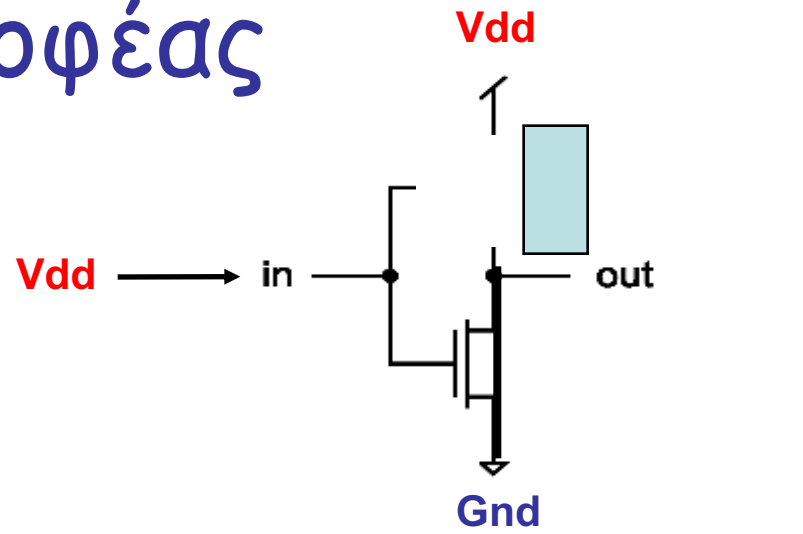
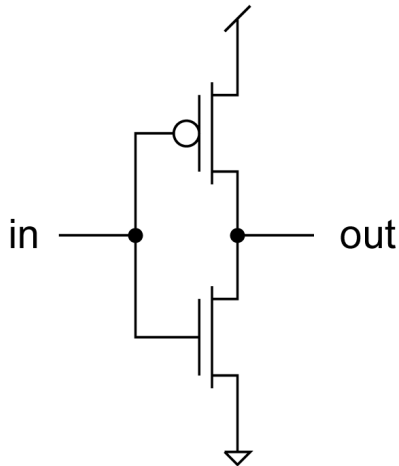
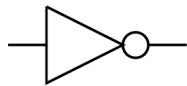
- 2 συμπληρωματικά είδη transistors
 - NMOS
(negative channel)
 - PMOS
(positive channel)
- Τα transistors συμπεριφέρονται σαν διακόπτες



Πύλες και τεχνολογία (3/6)

Ο αντιστροφέας

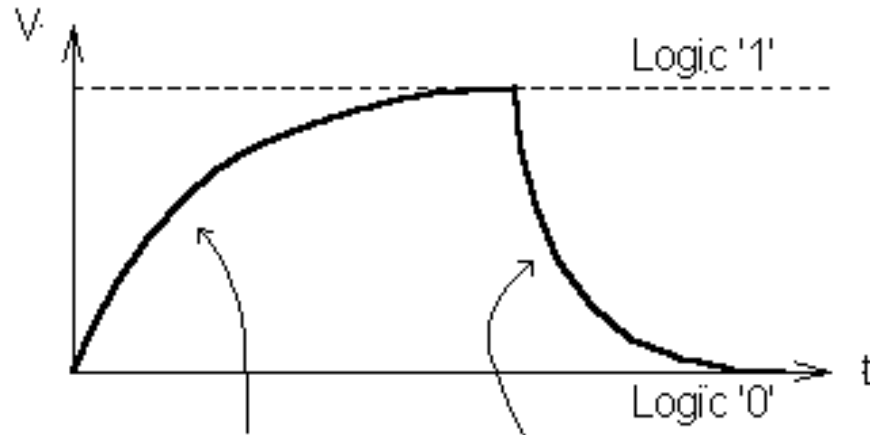
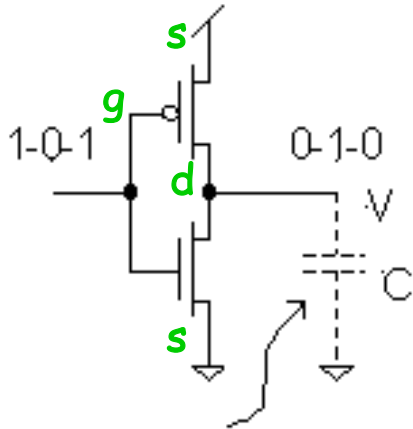
- Αντιστροφέας (NOT gate):



Πύλες και τεχνολογία (4/6)

Συμπεριφορά πυλών

Ο αντιστροφέας



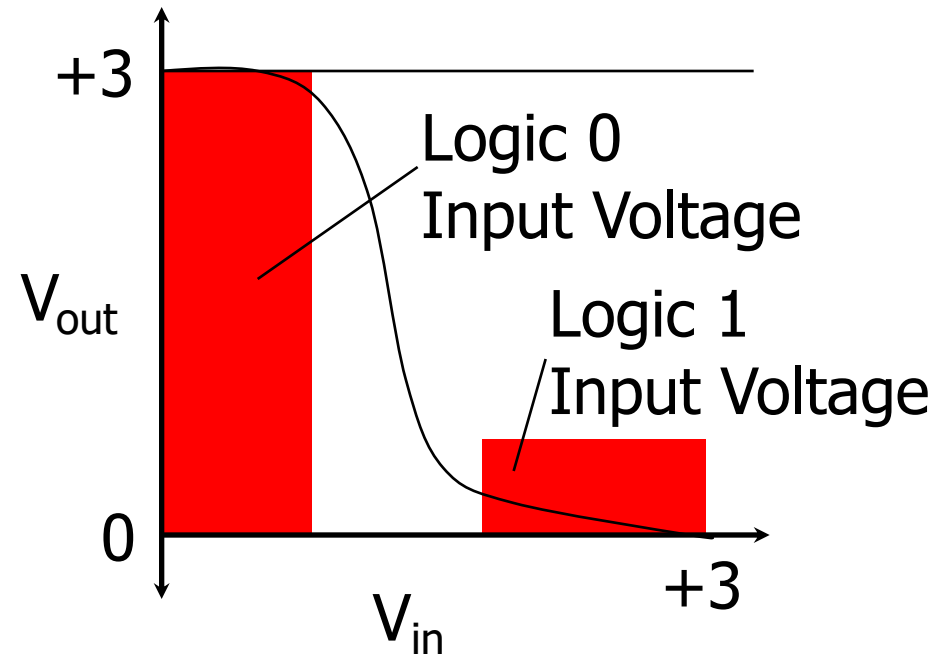
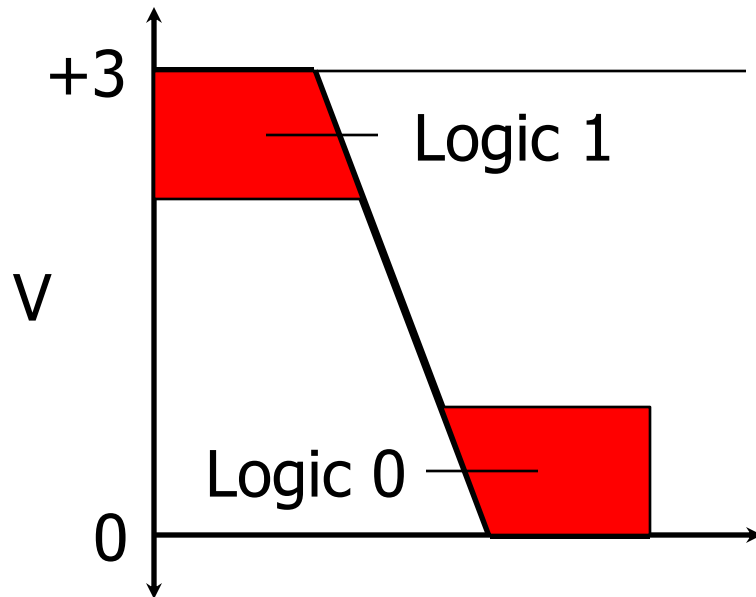
Μοντελοποιεί την είσοδο άλλων πυλών και την χωρητικότητα του καλωδίου

Ο ρυθμός εξαρτάται από την δύναμη του δικτύου ανέλκυσης και την χωρητικότητα C

Ο ρυθμός εξαρτάται από την δύναμη του δικτύου καθέλκυσης και την χωρητικότητα C

Πύλες και τεχνολογία (5/6)

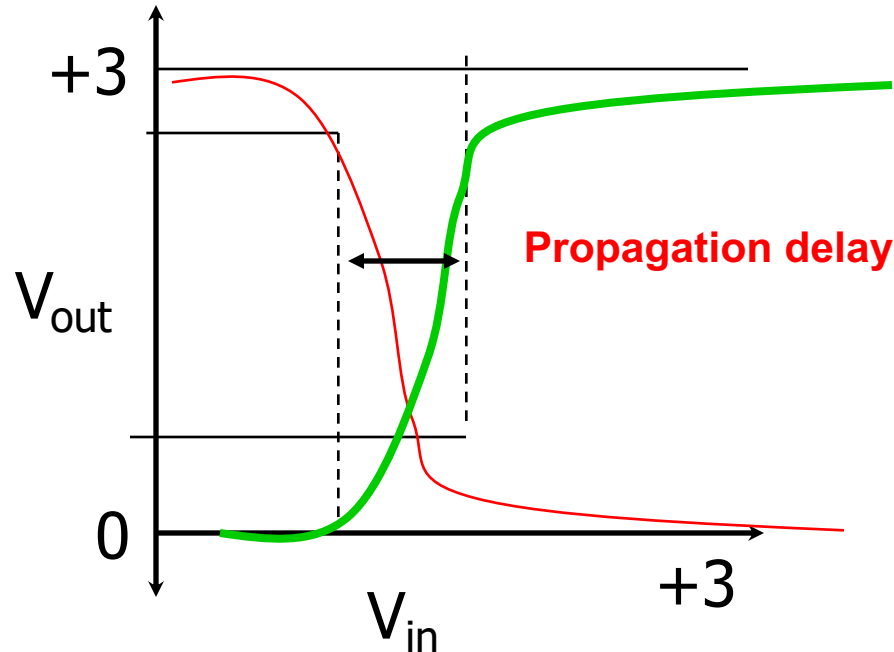
Λογικές τιμές



- Κατώφλι - Threshold
 - Λογικό 1 (true) : $V > V_{dd} - V_{th}$
 - Λογικό 0 (false) : $V < V_{th}$

Πύλες και τεχνολογία (6/6)

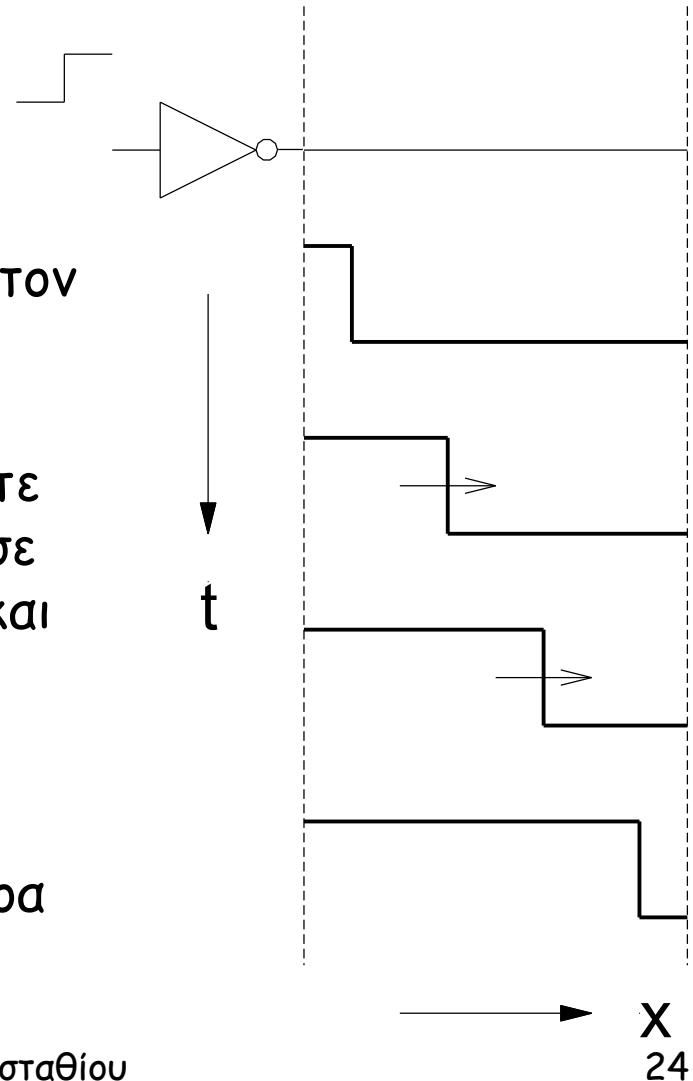
Το στοιχείο του χρόνου



- Οι αλλαγές στις εξόδους δεν είναι ακαριαίες !!!

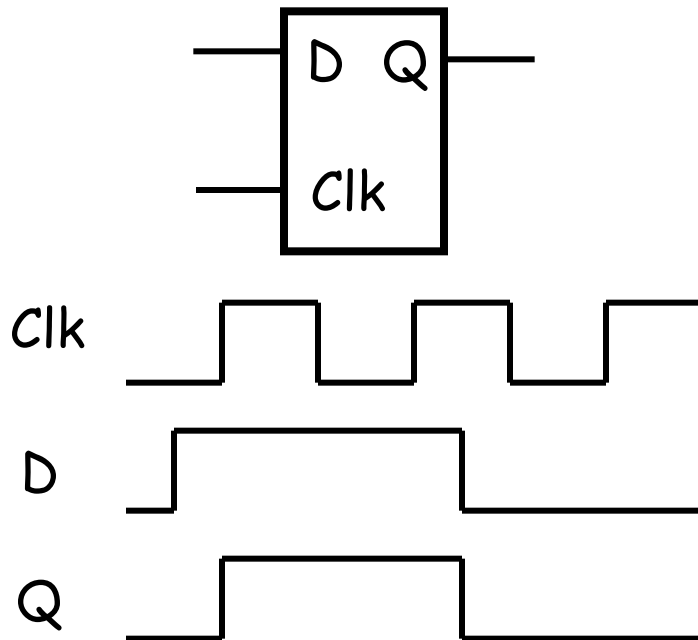
Καθυστερήσεις Καλωδίων

- Τα καλώδια έχουν καθυστέρηση!!!
 - Τα σήματα κινούνται περίπου με την ταχύτητα του φωτός ($\sim 30 \text{ cm/ns}$)
 - Ο χρόνος των σημάτων από την πηγή στον προορισμό είναι ο χρόνος μεταφοράς (transit time)
 - Στα ICs τα καλώδια είναι «κοντά» οπότε οι χρόνοι μεταφοράς είναι πολύ μικροί σε σύγκριση με την περίοδο του ρολογιού και συνήθως τις αγνοούμε!
 - Έχουν μεγάλη σημασία όμως στις τυπωμένες πλακέτες (PCBs)
 - Επίσης είναι πολύ σημαντικές σε γρήγορα chips με μακριά καλώδια
 - Π.χ. Busses, clocks

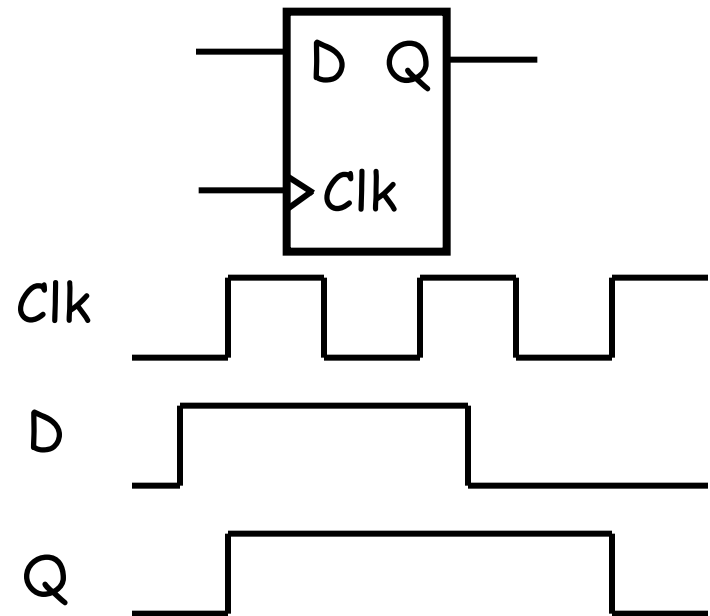


Στοιχεία Μνήμης: Latch vs Register

- Latch - Μανταλωτής:
- **Level triggered!**
- Αποθηκεύει τα δεδομένα όταν το ρολόι είναι 0.



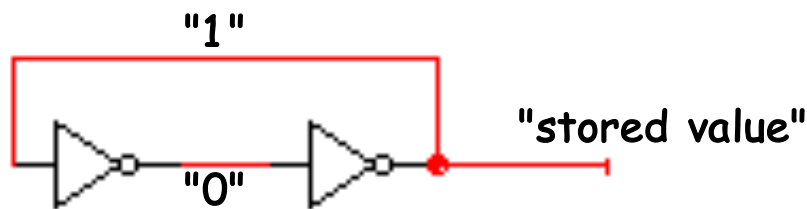
- Register - Καταχωρητής:
- **Edge triggered!**
- Αποθηκεύει τα δεδομένα στην ακμή του ρολογιού



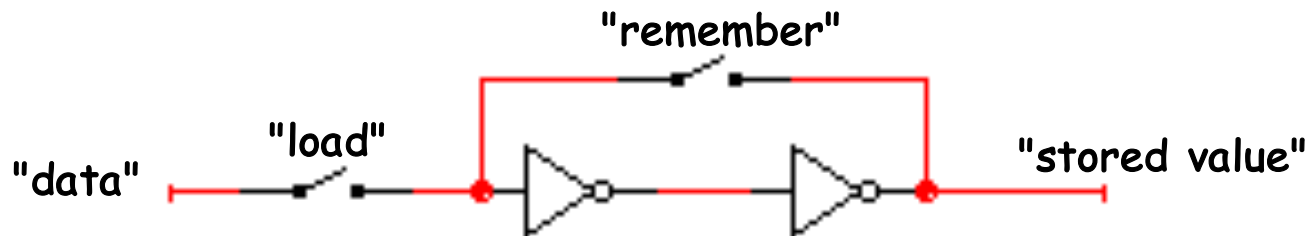
Υλοποιήσεις:

Στοιχεία μνήμης μέσω ανάδρασης

- Δύο αντιστροφείς σχηματίζουν ένα στατικό κύτταρο μνήμης (memory cell) - το πιο απλό με βρόγχο ανάδρασης
 - Θα κρατήσει την τιμή όσο τροφοδοτείται με ηλεκτρισμό

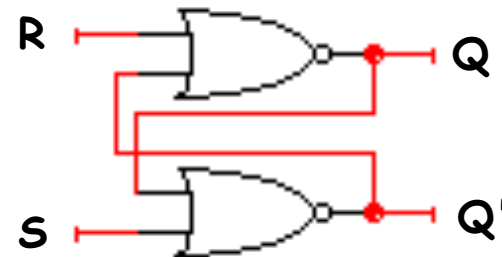
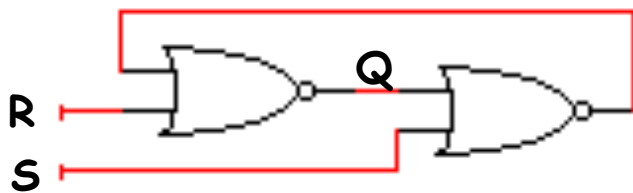


- Πώς μπορούμε να εισάγουμε νέα τιμή στο memory cell ;
 - Σπάμε το μονοπάτι της ανάδρασης (feedback)
 - Φορτώνουμε νέα τιμή

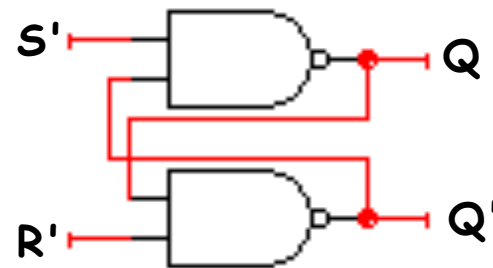
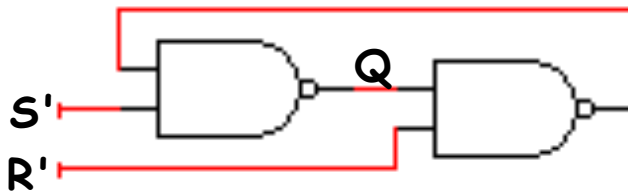


Υλοποιήσεις: Cross-coupled Gates - RS Latches

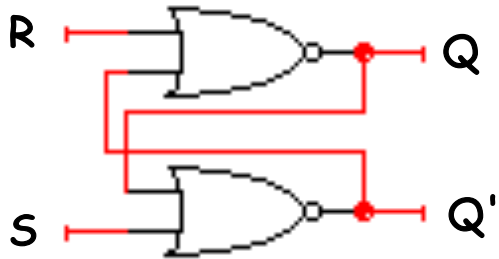
- Βρόγχος ανάδρασης με NOR για υλοποίηση RS latch
 - Παρόμοιο με το ζευγάρι των αντιστροφένων αλλά με τη δυνατότητα να θέσουμε την έξοδο στο 0 (reset=1) ή στο 1 (set=1)



- Βρόγχος ανάδρασης με NAND για υλοποίηση RS latch
 - Παρόμοιο με το ζευγάρι των αντιστροφένων αλλά με τη δυνατότητα να θέσουμε την έξοδο στο 0 (reset=0) ή στο 1 (set=0)

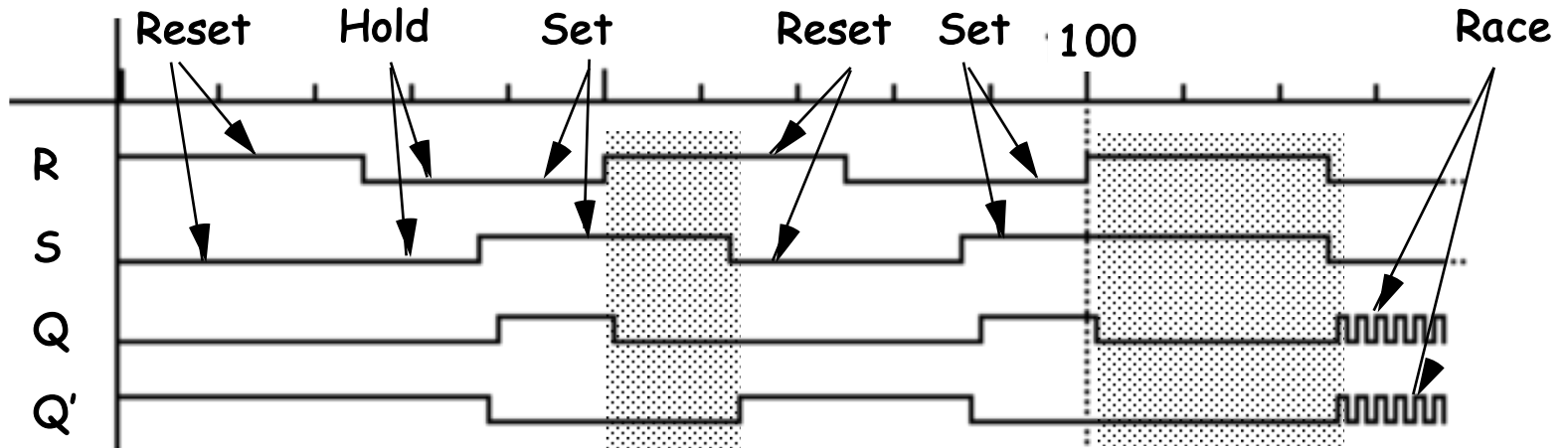


Χρονική Συμπεριφορά: NOR-based RS Latch



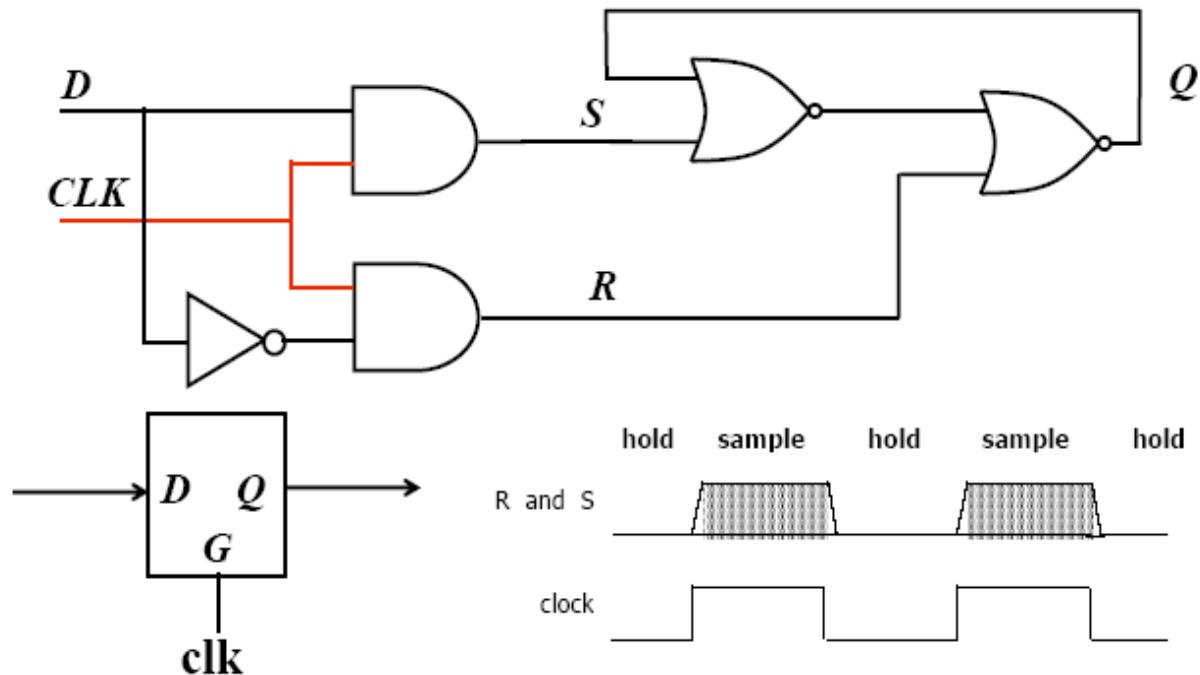
S	R	Q	\bar{Q}
0	0	Q	\bar{Q}
1	0	1	0
0	1	0	1
1	1	0	0

Απαγορευμένο



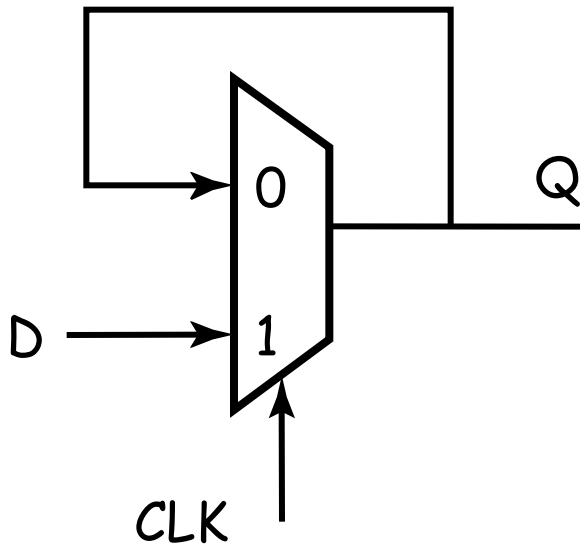
Υλοποιήσεις: D Latch

- D-Latch με ρολόι και data:
 - Φόρτωση δεδομένων εισόδου με το ρολόι
 - Υλοποίηση με gated NOR-based RS Latch



Υλοποιήσεις: Mux-Based Positive D Latches

Positive latch : «διαφανής» όταν $CLK=1$



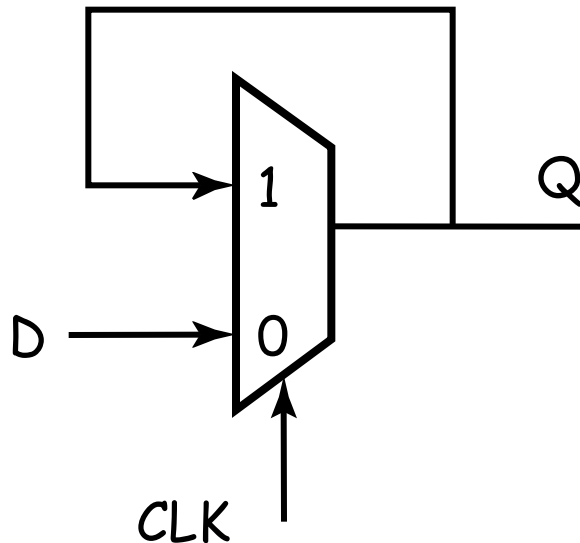
```
module pos_latch (clk, d, q)
input clk, d;
output q;
reg q;

always @(clk or d)
    if (clk) q <= d;

endmodule
```

Υλοποιήσεις: Mux-Based Negative D Latches

Negative latch : «διαφανής» όταν $CLK=0$

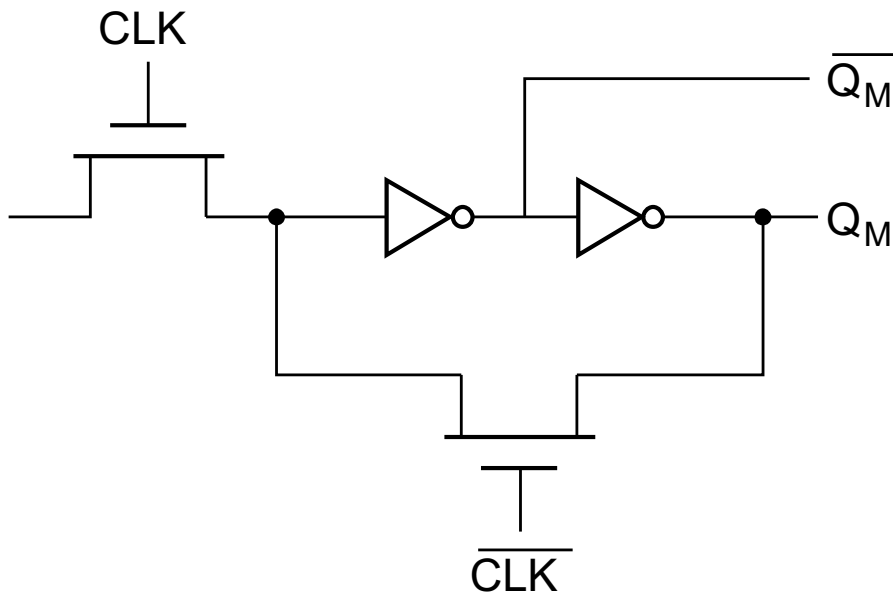


```
module neg_latch (clk, d, q)
input clk, d;
output q;
reg q;

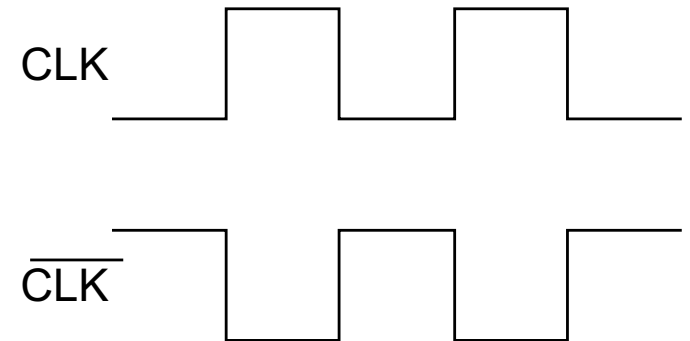
always @(clk or d)
    if (~clk) q <= d;

endmodule
```

Υλοποιήσεις: Latch με Transistors



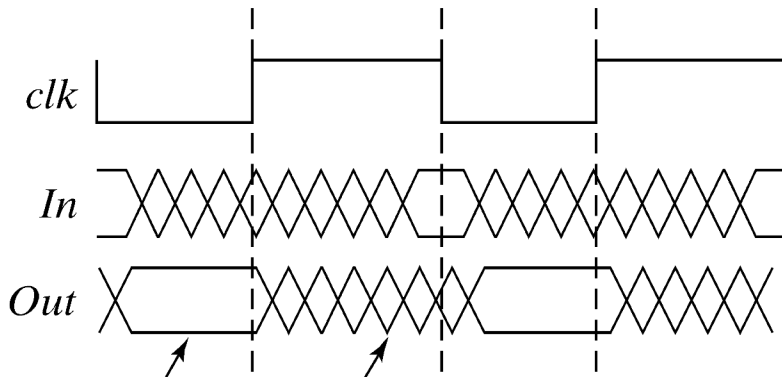
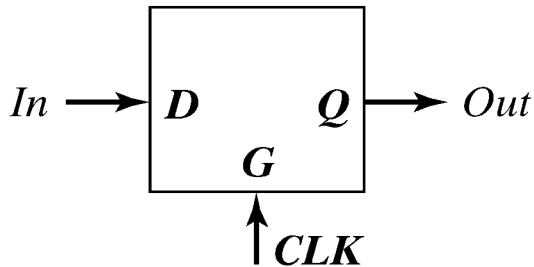
NMOS transistors



Δέχεται μη επικαλυπτόμενα
(non-overlapping) clocks

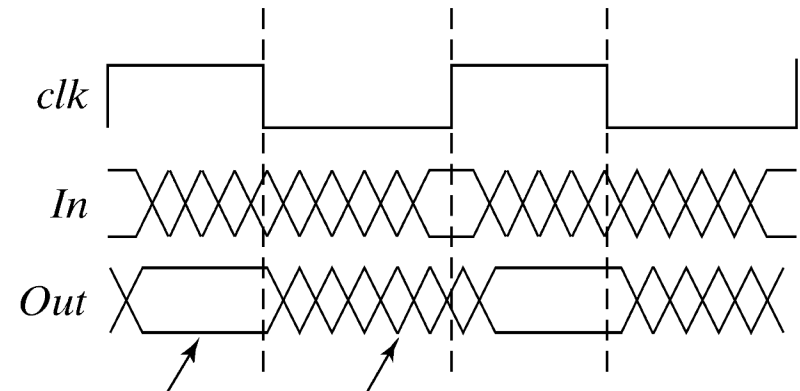
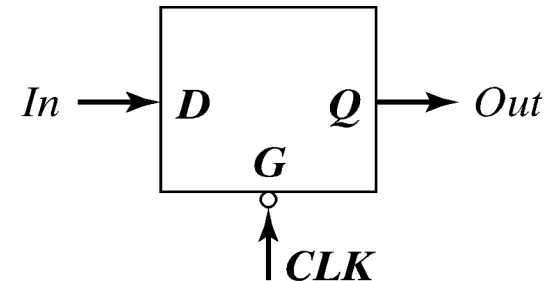
Latches

Positive Latch



Έξοδος σταθερή
Έξοδος ακολουθεί την είσοδο

Negative Latch

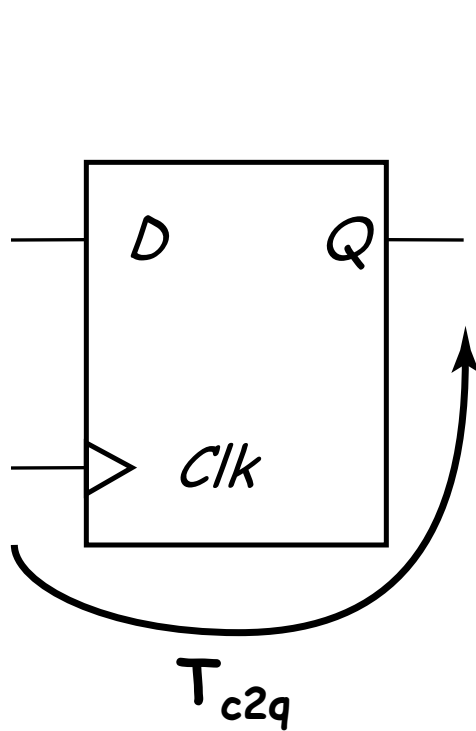


Έξοδος σταθερή
Έξοδος ακολουθεί την είσοδο

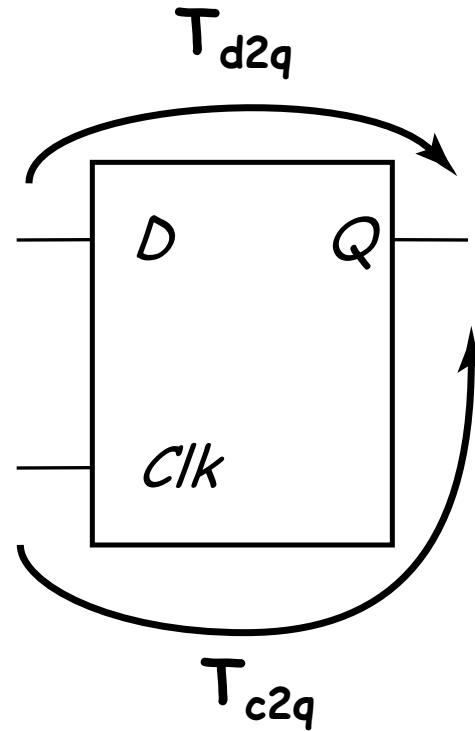
Χρονικές παράμετροι:

- T_{d2q} : Χρόνος από την είσοδο την έξοδο όταν το ρολόι θεωρείται ενεργό
- T_{c2q} : Χρόνος για την αλλαγή της εξόδου μετά την ενεργοποίηση του ρολογιού

Καθυστερήσεις Καταχωρητή - Μανταλωτή



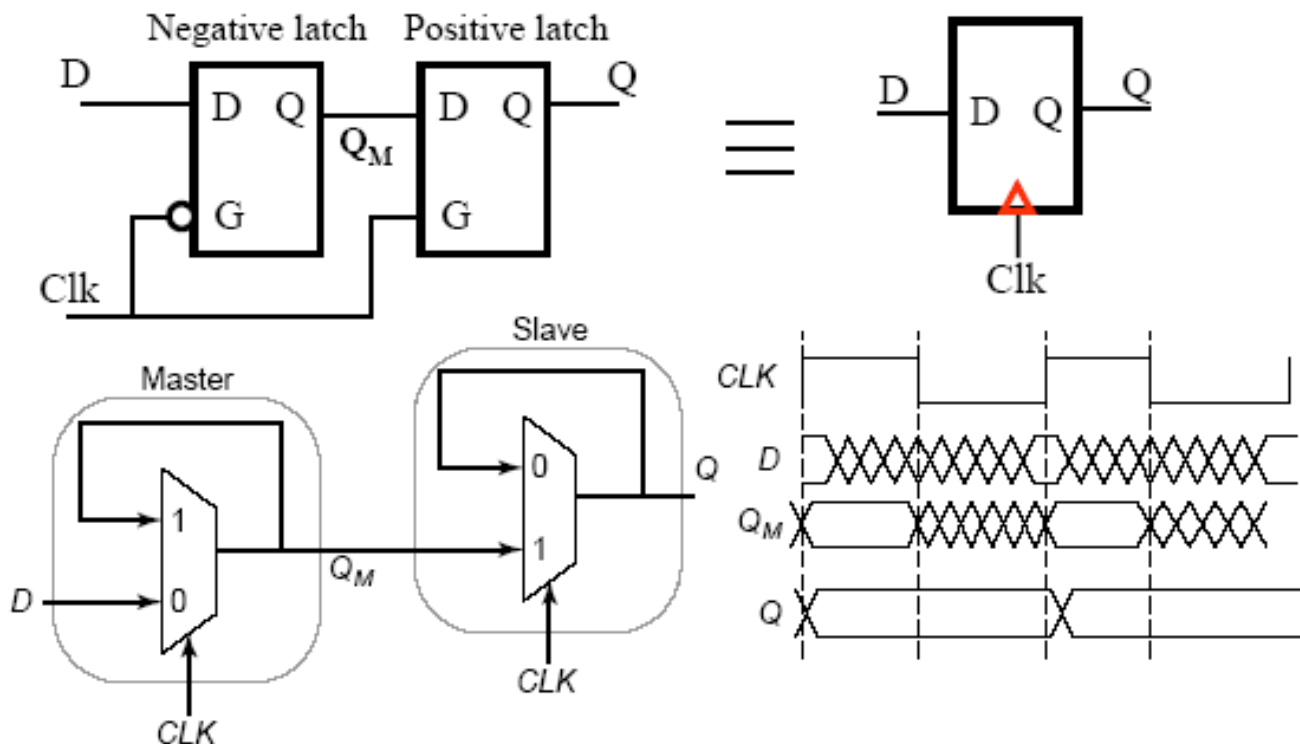
Register



Latch

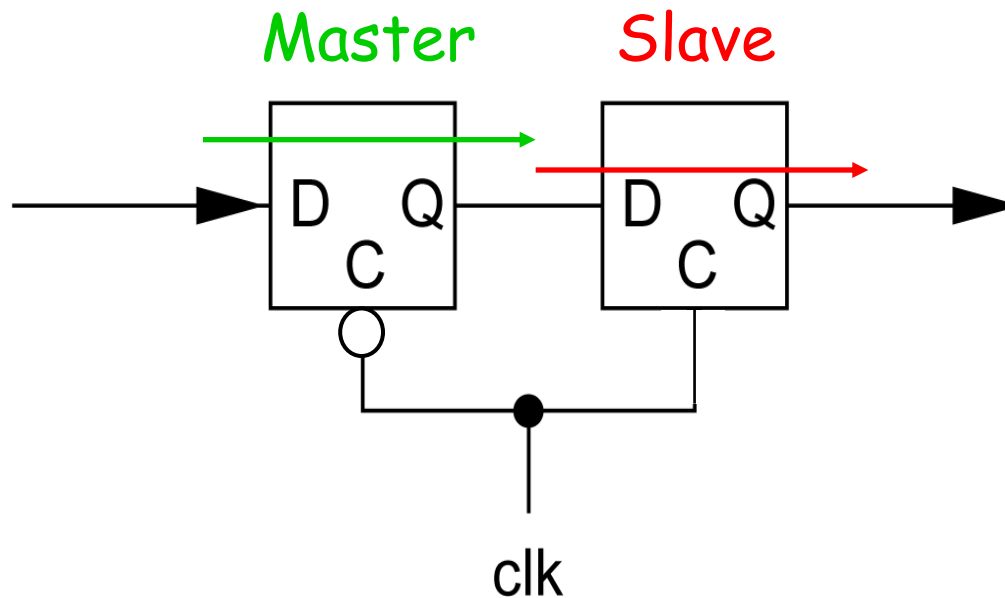
Ακμοπυροδότητος Καταχωρητής Αφέντη - Σκλάβου (1/2)

- Master-Slave καταχωρητής - Edge-triggered D Flip-Flop
 - Κατά την αρνητική φάση του ρολογιού αποθηκεύονται τα data στον master latch
 - Κατά την θετική φάση του ρολογιού αλλάζουν οι έξοδοι του slave latch



Ακμοπυροδότητος Καταχωρητής Αφέντη - Σκλάβου (2/2)

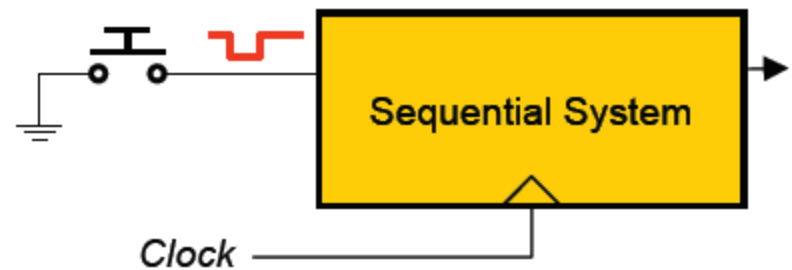
- Master-Slave καταχωρητής - Edge-triggered D Flip-Flop
 - Το setup-time προκύπτει από την καθυστέρηση του **master latch**
 - Το c2q-time προκύπτει από την καθυστέρηση του **slave latch**



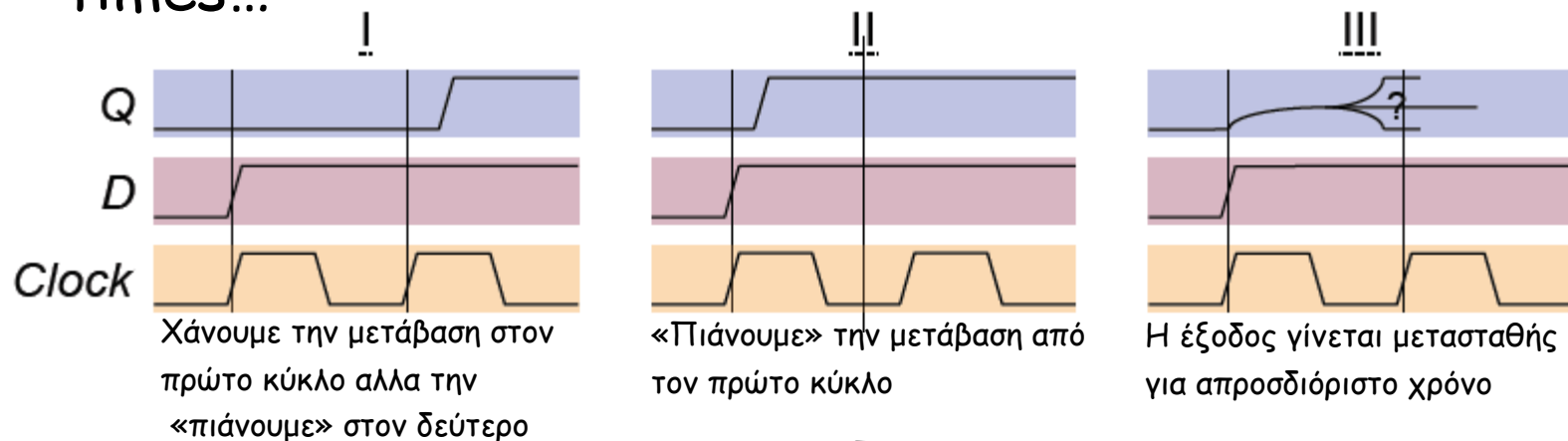
Ασύγχρονες είσοδοι σε σύγχρονα κυκλώματα (1/2)

- Τι γίνεται με τα εξωτερικά σήματα ; (π.χ. buttons)

- Δεν μπορούμε να εγγυηθούμε ότι οι χρόνοι setup και hold θα τηρούνται!!!

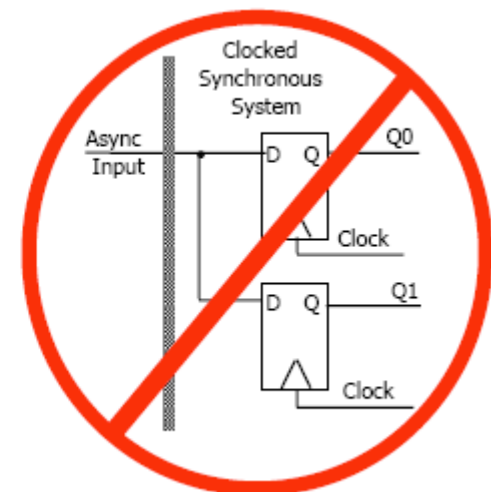
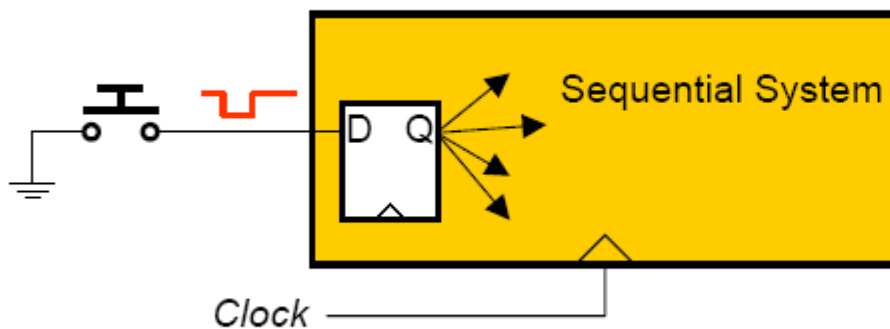


- Όταν ένα ασύγχρονο σήμα παραβιάζει setup και hold times...



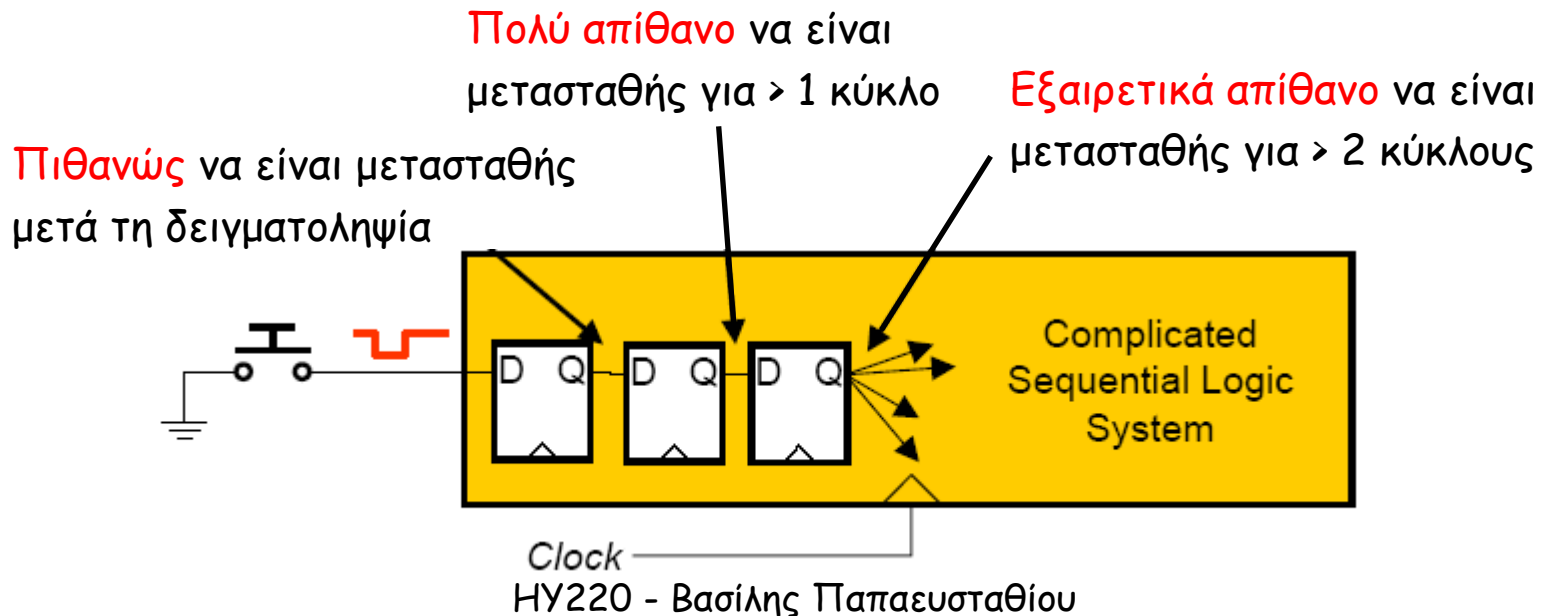
Ασύγχρονες εισοδοι σε σύγχρονα κυκλώματα (2/2)

- Σιγουρευτείτε ότι οι εξωτερικές εισοδοι πηγαίνουν σε ένα ακριβώς flip-flop!!!
 - Οι περιπτώσεις I και II μπορούν να προκαλέσουν λάθος στο κύκλωμα αν από την ίδια εισοδο σε ένα flip-flop συμβεί το φαινόμενο I ενώ σε ένα άλλο το II.



Χειρισμός Μεταστάθειας (Metastability)

- Περίπτωση III - Αδύνατον να προληφθεί !
- Τα μοντέρνα ψηφιακά κυκλώματα βγαίνουν σχετικά γρήγορα από καταστάσεις μεταστάθειας.
- Λύση: Περιμένουμε τα σήματα να σταθεροποιηθούν
 - Συγχρονισμός με 2-3 flips-flops (synchronization)



Η αρχή του Pipelining με ένα παράδειγμα (1/2)

- Ανάλογο πλύσης ρούχων:

- βήμα 1: wash (20 minutes)
- βήμα 2: dry (20 minutes)
- βήμα 3: fold (20 minutes)

60 minutes x 4 loads \Rightarrow 4 hours

- Και αν επικαλύψουμε τα βήματα - στάδια:

20mins 20mins 20mins 20mins 20mins 20mins

wash	load1	load2	load3	load4		
dry		load1	load2	load3	load4	
fold			load1	load2	load3	load4

overlapped \Rightarrow 2 hours

Η αρχή του Pipelining με ένα παράδειγμα (2/2)

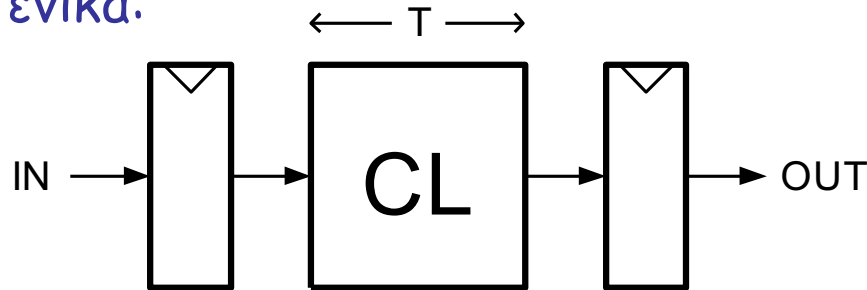
20mins 20mins 20mins 20mins 20mins 20mins

wash	load1	load2	load3	load4		
dry		load1	load2	load3	load4	
fold			load1	load2	load3	load4

- Αν αύξησουμε τον αριθμό των loads, ο μέσος χρόνος ανα load πλησιάζει τα 20 minutes
- **Καθυστέρηση - Latency** (ο χρόνος από την αρχή μέχρι το τέλος) για ένα load = 60 min
- **Παροχή - Throughput** = 3 loads/hour
- Pipelined throughput \approx # of pipe stages x un-pipelined throughput.

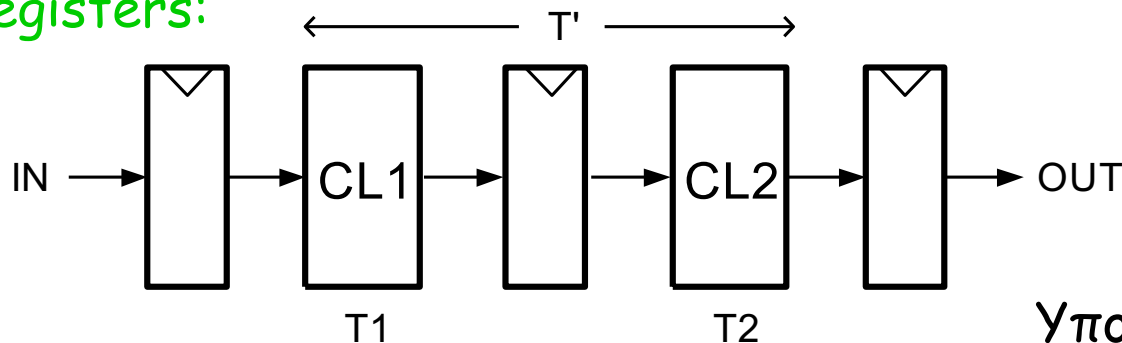
Pipelining

- Γενικά:



Υποθέστε $T = 8 \text{ ns}$
 $T_{FF}(\text{setup} + \text{clk} \rightarrow \text{q}) = 1 \text{ ns}$
 $F = 1/9 \text{ ns} = 111 \text{ MHz}$

- Κόβουμε το CL block σε κομμάτια (stages) και τα χωρίζουμε με registers:



Υποθέστε $T1 = T2 = 4 \text{ ns}$

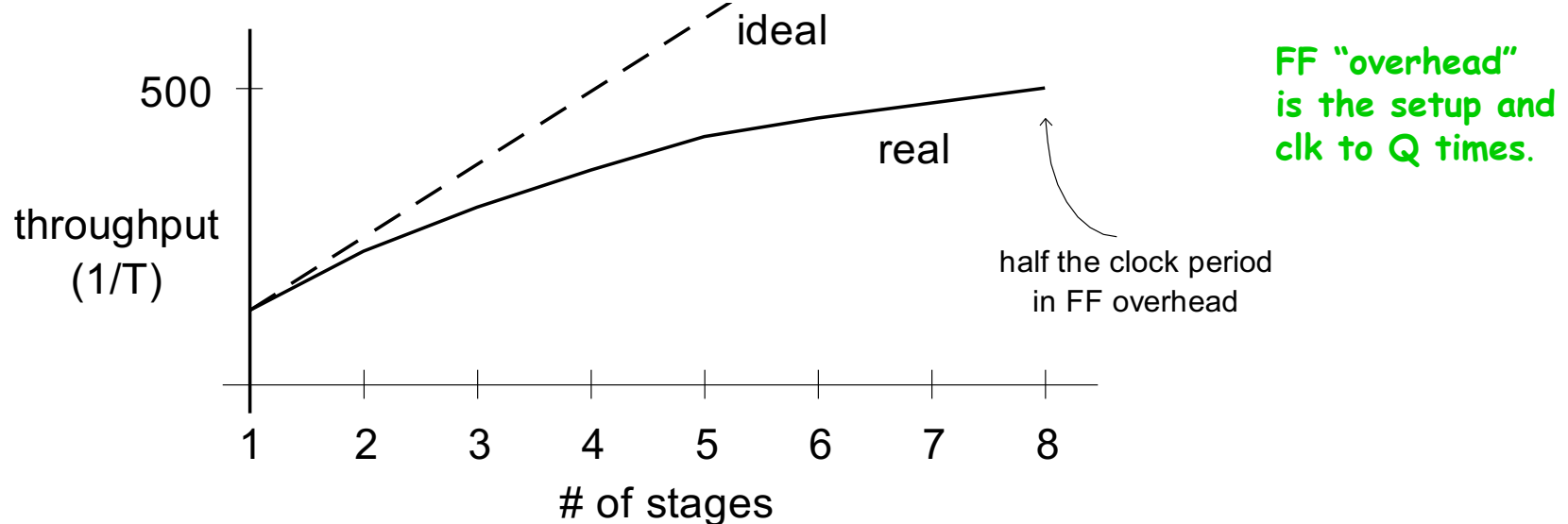
$$T' = 4 \text{ ns} + 1 \text{ ns} + 4 \text{ ns} + 1 \text{ ns} = 10 \text{ ns}$$

$$F = 1/(4 \text{ ns} + 1 \text{ ns}) = 200 \text{ MHz}$$

- CL block παράγει νέο αποτέλεσμα κάθε 5 ns αντί για κάθε 9 ns

Όρια στο Pipelining

- Χωρίς το χρονικό κόστος (overhead) των FF, η βελτίωση στο throughput θα ήταν ανάλογη του αριθμού των σταδίων(stages) του pipeline
 - Αν προσθέσουμε πολλά στάδια, το overhead των FF αρχίζει να κυριαρχεί!



- Άλλοι περιοριστικοί παράγοντες για πιο αποδοτικό pipelining:
 - Οι καθυστερήσεις/αβεβαιότητες του ρολογιού (clock skew) συνεισφέρουν στο overhead
 - Μη ισορροπημένα στάδια
 - Το κόστος των FFs κυριαρχεί
 - Κατανάλωση ισχύος για την διανομή του ρολογιού(clock distribution power consumption)
 - Αναδράσεις στις λογικές - *feedbacks (dependencies between loop iterations)*