

# 3<sup>η</sup> Θεματική Ενότητα

## *Reuse Methodology*

Μεθοδολογία Σχεδιασμού για  
Επαναχρησιμοποίηση

1

## Μέρος I

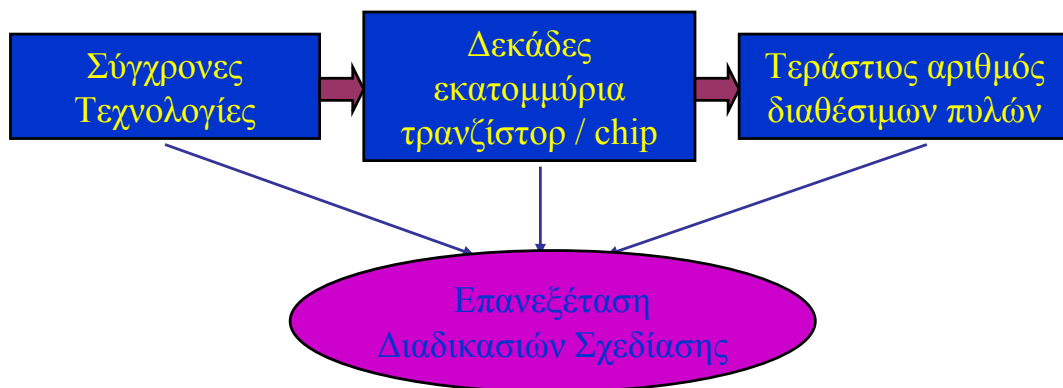
### *Εισαγωγή*

Μεθοδολογία Σχεδιασμού για  
Επαναχρησιμοποίηση

2

## Εισαγωγικές Έννοιες

---



Η δημιουργία ενός ASIC με τέτοιο αριθμό πυλών είναι ιδιαίτερα δύσκολη με τα υπάρχοντα εργαλεία, ενώ ο **χρόνος σχεδίασης** πρέπει να είναι μικρός, και η **πολυπλοκότητα** χαμηλή.

Επαναχρησιμοποίηση σχεδιασμού = χρήση  $\left\{ \begin{array}{l} \text{Προ-σχεδιασμένων} \\ \text{Προ-επιβεβαιωμένων} \end{array} \right.$  Cores

## Εισαγωγικές Έννοιες

---

Η μεθοδολογία ανάπτυξης διαφέρει ανάλογα με τους σχεδιαστές. Όμως όλοι αντιμετωπίζουν ένα κοινό σύνολο προβλημάτων:

- **Time-to-market** pressure  $\Rightarrow$  Ταχεία ανάπτυξη προϊόντος
- **Ποιότητα** προϊόντος σε **απόδοση, επιφάνεια, κατανάλωση**
- Η **αυξημένη πολυπλοκότητα** κάνει την **επιβεβαίωση ορθής λειτουργίας** πιο δύσκολη
- Η ομάδα ανάπτυξης έχει διαφορετικά επίπεδα και περιοχές ειδίκευσης και συχνά είναι διασκορπισμένη σε διαφορετικά γεωγραφικά σημεία.
- Παλαιότεροι σχεδιασμοί δεν μπορούν να χρησιμοποιηθούν γιατί έχει αλλάξει η **σχεδιαστική ροή**, τα εργαλεία και οι γενικές οδηγίες.
- Οι SoC σχεδιασμοί εμπεριέχουν επεξεργαστικούς πυρήνες και σημαντικά κομμάτια λογισμικού που οδηγούν σε **σχεδιαστικές προκλήσεις**.

## Εισαγωγικές Έννοιες

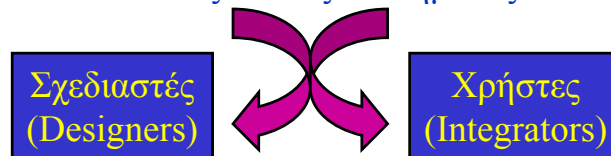
---

**Αποτέλεσμα:** πολλές σχεδιαστικές ομάδες υιοθετούν μια **βασισμένη σε block σχεδιαστική προσέγγιση** που δίνει έμφαση στην **επαναχρησιμοποίηση (reuse)**. Αν δεν ακολουθηθεί αυτή η προσέγγιση, τότε απαιτείται **reverse engineering** για να χρησιμοποιηθεί ένας ξένος σχεδιασμός ⇒ μεγάλο κόστος σε χρόνο.

Τι θα περιγράψουμε:

- ▣ Πως τα reusable macros ταιριάζουν σε ένα περιβάλλον σχεδιασμού SoC.
- ▣ Πως να σχεδιάσουμε reusable soft/hard macros.
- ▣ Πως να ενσωματώνουμε soft/hard macros σε ένα SoC.
- ▣ Πως να επιβεβαιώσουμε χρονισμούς και λειτουργικότητα σε ένα μεγάλο Soc

### Οπτικές Γωνίες Μαθήματος



---

Μεθοδολογία Σχεδιασμού για  
Επαναχρησιμοποίηση

5

## Εισαγωγικές Έννοιες

---

**Ορισμοί:**

1. Macro, Core, Block: είναι σχεδιαστικές μονάδες αυθύπαρκτες.
2. Subblock: ένα τμήμα ενός Macro, Core, Block πολύ μικρό για να μπορεί να είναι αυθύπαρκτο.
3. Hard macro: ένας σχεδιασμός που δίδεται αφού περάσει από το στάδιο placement-routing.
4. Soft macro: ένας σχεδιασμός που δίδεται σαν κώδικας RTL προς σύνθεση.

### Virtual Socket Interface Alliance

Είναι μια βιομηχανική ομάδα που εργάζεται με στόχο την υιοθέτηση **standards** για εργαλεία και σχεδιαστικές πρακτικές

---

Μεθοδολογία Σχεδιασμού για  
Επαναχρησιμοποίηση

6

## Εισαγωγικές Έννοιες

---

Η πρόκληση:

Μια αποδοτική μεθοδολογία απαιτεί μία εκτεταμένη βιβλιοθήκη από reusable blocks ή macros. Η διαδικασία σχεδιασμού reusable macros ακολουθεί τις εξής αρχές:

- Κατασκευή όλων των επιπέδων σχεδίασης με την λογική ότι το block θα χρησιμοποιηθεί ξανά σε άλλες σχεδιάσεις από άλλους σχεδιαστές.
- Χρήση εργαλείων και διαδικασιών που συλλαμβάνουν την πληροφορία του σχεδιασμού συνεχώς.
- Χρήση εργαλείων και διαδικασιών που κάνουν εύκολη την σύνδεση των μονάδων χωρίς την παρουσία του σχεδιαστή τους.

*Πολλές Τεχνικές Reusability  
είναι απλά καλές τεχνικές  
σχεδιασμού*

*Καλογραμμένος κώδικας  
Καλό documentation  
Εκτενής σχολιασμός  
Καλοσχεδιασμένο περιβάλλον επιβεβαίωσης  
Ισχυρά scripts*

## Εισαγωγικές Έννοιες

---

Επιπλέον για ένα hardware macro έχουμε τις εξής απαιτήσεις:

- Σχεδιασμός για την λύση **γενικού προβλήματος** (διαμορφούμενα blocks).
- Σχεδιασμός για χρήση σε **πολλές τεχνολογίες**. Για τα soft macros σημαίνει ότι τα scripts πρέπει να έχουν ικανοποιητική ποιότητα αποτελεσμάτων για πολλές βιβλιοθήκες. Για τα hard macros απαιτείται μια αποτελεσματική στρατηγική αντιστοίχισης σε άλλες τεχνολογίες.
- Σχεδιασμός για εξομοίωση με **πολλούς εξομοιωτές**.
- Επιβεβαίωση λειτουργίας **ανεξάρτητη από το chip** στο οποίο θα χρησιμοποιηθεί.
- Επιβεβαίωση σε **υψηλό επίπεδο εμπιστοσύνης**.
- **Πλήρως documented** για διαμορφώσεις, παραμέτρους, περιορισμούς, απαιτήσεις επικοινωνίας κλπ.

## Εισαγωγικές Έννοιες

---

Συνηθισμένα προβλήματα με κώδικα μη-επαναχρησιμοποιήσιμο:

- Η αναπαράσταση του σχεδιασμού δεν είναι κατάλληλη (πχ για μετατροπή από Verilog σε VHDL).
- Ο σχεδιασμός έρχεται με ελλιπείς πληροφορίες σχεδίασης, χωρίς περιγραφή λειτουργικής συμπεριφοράς και με μη αναγνώσιμο ασχολίαστο κώδικα.
- Δεν παρέχονται scripts ή αυτά είναι αδύνατον να χρησιμοποιηθούν.
- Τα εργαλεία που χρησιμοποιήθηκαν για την δημιουργία του σχεδιασμού δεν υπάρχουν πλέον.
- Ένα διαθέσιμο hard macro μπορεί να έχει τόσο αργό μοντέλο εξομοίωσης που η εξομοίωση σε επίπεδο συστήματος δεν είναι πρακτική.

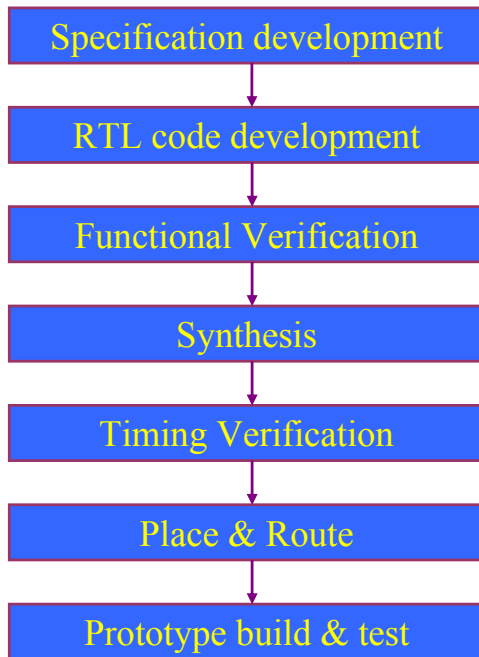
## Κανονικός σχεδιασμός SOC

---

Τα πραγματικά SoC είναι αρκετά πιο περίπλοκα από αυτό το κανονικό παράδειγμα. Περιλαμβάνουν πολλά σύνολα από IP interfaces, μετασχηματισμούς δεδομένων, επεξεργαστές, DSPs. Επίσης το σύστημα μνήμης είναι πιο περίπλοκο με πολλαπλά επίπεδα διαχείρισης, διαμοιραζόμενα τμήματα κλπ.

## Ροή σχεδιασμού συστήματος

---



Υπάρχουν οι παρακάτω επιλογές σε σχεδιαστικές ροές:

- Waterfall ή Spiral μοντέλο,
- Μεθοδολογία Top-down, bottom-up ή συνδυασμός

Το project περνά από τη μία φάση στην άλλη **χωρίς να ξαναγυρνά πίσω** (ελάχιστο interaction μεταξύ των σχεδιαστικών ομάδων). Δουλεύει καλά μέχρι 100k πύλες και έως .5μ

## Ροή σχεδιασμού συστήματος

---

Με την αύξηση της πολυπλοκότητας και την συρρίκνωση της τεχνολογίας οι σχεδιαστές μετακινούνται προς την ροή τύπου spiral, όπου η ομάδα σχεδιασμού εργάζεται ταυτόχρονα σε πολλές όψεις του σχεδιασμού.

Σχ 2-3

## Ροή σχεδιασμού συστήματος

---

Η ροή **spiral** χαρακτηρίζεται από:

- α. Παράλληλη ταυτόχρονη ανάπτυξη υλικού και λογισμικού,
- β. Παράλληλη επιβεβαίωση και σύνθεση των τμημάτων,
- γ. Το Floorplanning και place-route συμπεριλαμβάνονται στην διαδικασία σύνθεσης.
- δ. Τα τμήματα σχεδιάζονται μόνο αν δεν υπάρχει κάποιο προσχεδιασμένο hard ή soft macro.

## Ροή σχεδιασμού συστήματος

---

Η κλασσική διαδικασία top/down είναι μια αναδρομική διαδικασία που ξεκινά από το **specification** και το **decomposition** και καταλήγει με **integration** και **verification**:

- α. Συγγραφή πλήρους specification για το σύστημα ή το υποσύστημα που σχεδιάζεται.
- β. Εκλέπτυνση αρχιτεκτονικής και αλγορίθμων,
- γ. Αποσύνθεση της αρχιτεκτονικής σε καλά καθορισμένα macros.
- δ. Σχεδίαση ή επιλογή έτοιμων macros.
- ε. Σύνδεση των macros στο υψηλότερο επίπεδο. Επιβεβαίωση λειτουργίας και χρονισμών.
- στ. Επιβεβαίωσε την λειτουργία του συστήματος σε όλες τις παραμέτρους τους.

# Ροή σχεδιασμού συστήματος

---

Η top-down μεθοδολογία θεωρεί ότι τα blocks στο κατώτερο επίπεδο είναι σχεδιασμένα και λειτουργούν. Αν αυτό δεν συμβεί έχουμε επανάληψη όλης της διαδικασίας ↓ γίνεται χρήση μίας **μίξης των μεθοδολογιών top-down, bottom-up** με επιβεβαίωση των blocks στο κατώτερο επίπεδο ταυτόχρονα με τον καθορισμό των προδιαγραφών στο ανώτερο επίπεδο.



Οι βιβλιοθήκες των reusable cores που είναι σχεδιασμένα και επιβεβαιωμένα από πριν **επιταχύνουν** την διαδικασία σχεδιασμού.

## Correct by corection

Μεθοδολογία της Microsystems για την κατασκευή του επεξεργαστή UltraSPARC όπου οι σχεδιαστές σχεδίασαν σε όλα τα βήματα της διαδικασίας για να δουν πως ο σχεδιασμός τους επηρέαζε τα υπόλοιπα βήματα που ανήκαν συνήθως σε άλλες ομάδες.

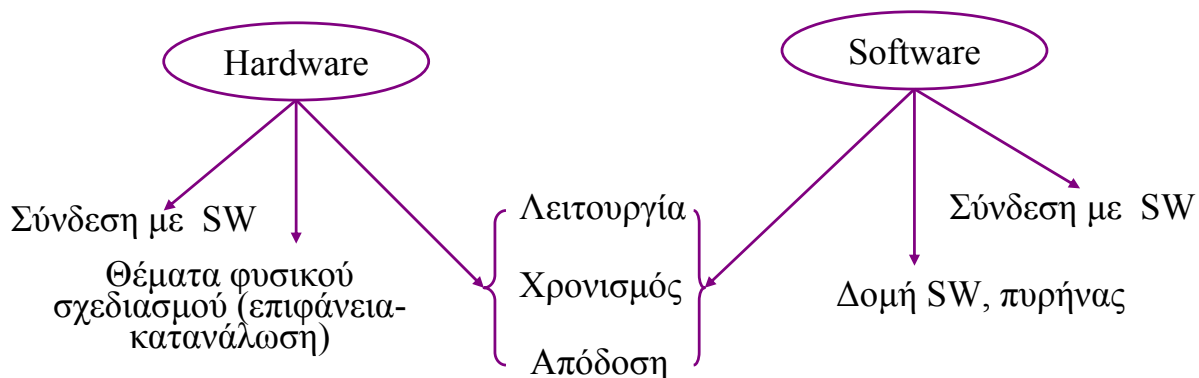
# Specification Problem

---

Στο πρώτο τμήμα της διαδικασίας σχεδιασμού έχουμε ανάπτυξη, επιβεβαίωση και εκλέπτυνση ενός συνόλου λεπτομερειών λειτουργίας (specifications) έως ότου επιτρέπουν την κωδικοποίηση σε RTL. Είναι **η πιο προκλητική φάση** ενός project.

Αποτυχία της ⇒ λάθη που ανακαλύπτονται σε άλλες φάσεις αργότερα (στοιχίζουν). Επίσης σημαντικό το **documentation** από νωρίς.

## Απαιτήσεις των Specifications



# Specification Problem

---

Συνήθως τα specifications γράφονται σε φυσική γλώσσα (πχ Αγγλικά) με αποτέλεσμα διάφορα προβλήματα  $\Rightarrow$  χρήση **εκτελέσιμων** specifications.

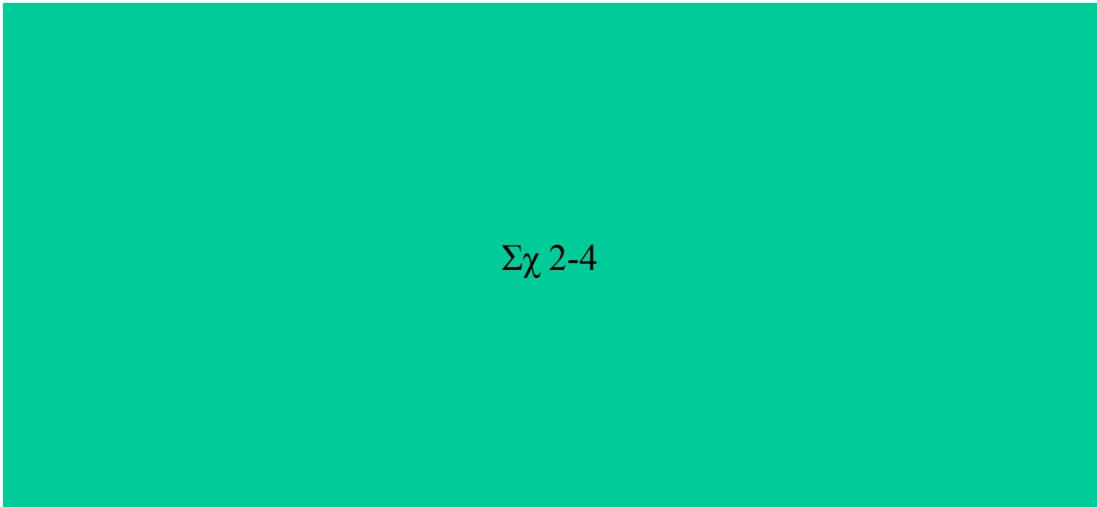
**Είδη Specifications:**

**Τυπικά (formal).** Τα απαιτούμενα χαρακτηριστικά του σχεδιασμού ορίζονται ανεξάρτητα από κάθε υλοποίηση. Υπάρχουν γλώσσες όπως οι VHDL, VSPEC που κάνουν αυτή τη δουλειά, και παρέχουν specifications και για χαρακτηριστικά επιφάνειας και κατανάλωσης. *Κυρίως για ερευνητικούς σκοπούς.*

**Εκτελέσιμα (executable).** Περιγραφή λειτουργίας για τους σχεδιασμούς. Είναι τυπικά αφηρημένα μοντέλα για το υλικό ή/και το λογισμικό που καθορίζεται. Στα υψηλότερα επίπεδα χρησιμοποιείται C, C++ ή SDL ενώ στα χαμηλότερα Verilog ή VHDL. Ο γρήγορος προσδιορισμός τους επιτρέπει στην ομάδα σχεδιασμού να επιβεβαιώσει την βασική λειτουργικότητα και το interface HW/SW πολύ πριν καθοριστούν οι λεπτομέρειες

## Η διαδικασία σχεδιασμού του συστήματος

---



Σχ 2-4

# Η διαδικασία σχεδιασμού του συστήματος

---

Η διαδικασία συμπεριλαμβάνει τα ακόλουθα βήματα:

**Καθορισμός Συστήματος.** Καθορίζονται οι απαιτήσεις (λειτουργίες, απόδοση, κόστος, χρόνος ανάπτυξης).

**Εκλέπτυνση του μοντέλου και έλεγχος.** Δημιουργείται ένα περιβάλλον επιβεβαίωσης για το μοντέλο υψηλού επιπέδου, για εκλέπτυνση και έλεγχο (λειτουργικότητας και απόδοσης) του αλγορίθμου. Αν σχεδιαστεί σωστά μπορεί να χρησιμοποιηθεί αργότερα για επιβεβαίωση μοντέλων υλικού και λογισμικού όπως το μοντέλο RTL. Στόχος εδώ είναι ο αλγόριθμος και όχι η υλοποίηση.

**Διαμέριση υλικού/λογισμικού.** Είναι διαδικασία που βασίζεται κυρίως σε προσωπική εργασία του σχεδιαστή και απαιτεί εμπειρία, καλή γνώση των trade-offs κόστους/απόδοσης για διάφορες αρχιτεκτονικές. Μια πλούσια βιβλιοθήκη επαναχρησιμοποιήσιμων και προ-επιβεβαιωμένων macros μπορεί να βοηθήσει αρκετά. Τέλος πρέπει να καθοριστεί το interface μεταξύ λογισμικού/υλικού.

# Η διαδικασία σχεδιασμού του συστήματος

---

**Block specification.** Το αποτέλεσμα της διαμέρισης είναι ένα specification για το υλικό και ένα για το λογισμικό. Το specification του υλικού περιλαμβάνει περιγραφή των βασικών συναρτήσεων, χρονισμούς, επιφάνεια, απαιτήσεις κατανάλωσης και την διασύνδεση υλικού και λογισμικού με λεπτομερή περιγραφές των I/O pins και τον χάρτη των καταχωρητών.

**Μοντέλο συμπεριφοράς (behavioral) και συνεξομοίωση (cosimulation).** Παράλληλη ανάπτυξη του μοντέλου συμπεριφοράς του υλικού και μιας πρωτότυπης έκδοσης του λογισμικού. Μετά με συνεξομοίωση έχουμε εκλέπτυνση των παραπάνω μοντέλων και παρασκευή ισχυρών specifications. Η συνεξομοίωση συνεχίζεται σε όλα τα στάδια του σχεδιασμού.

## Μέρος II

### *Σχεδιασμός σε επίπεδο συστήματος: Κανόνες και Εργαλεία*

Μεθοδολογία Σχεδιασμού για  
Επαναχρησιμοποίηση

21

---

### Ενδομηματικά θέματα.

*Πριν την επιλογή ή τον σχεδιασμό των macros πρέπει η ομάδα σχεδιασμού να θέσει έναν αριθμό σχεδιαστικών κανόνων που αφορούν θέματα όπως clocking, reset, interface architecture, dft κλπ.*

---

Μεθοδολογία Σχεδιασμού για  
Επαναχρησιμοποίηση

22

## Θέματα χρονισμού και σύνθεσης.

---

### Σύγχρονο ή Ασύγχρονο σχεδιαστικό στυλ?

**Κανόνας.** Το σύστημα θα πρέπει να είναι σύγχρονο και βασισμένο σε καταχωρητές. Latches χρησιμοποιούνται μόνο για FIFOs ή μικρές μνήμες οι οποίες όμως πρέπει να είναι σύγχρονες προς τον έξω κόσμο.

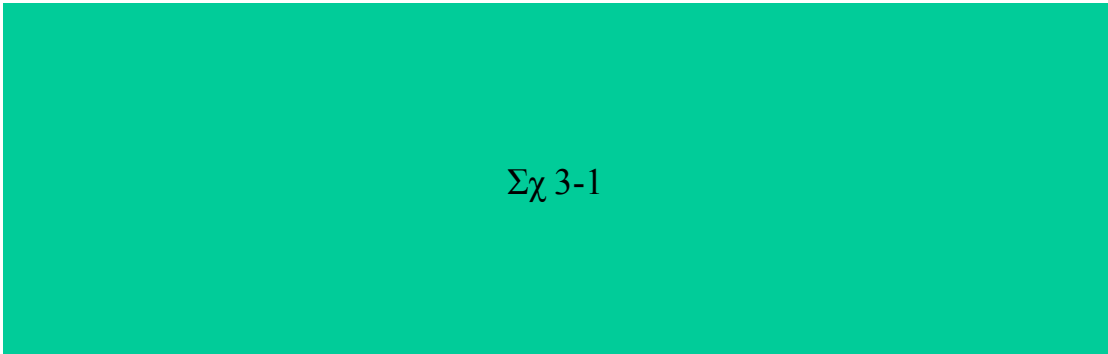
Παλιές μεθοδολογίες χρησιμοποιούσαν εκτενώς latches (multiphase, non-overlapping clocks →clocking pipeline stages) αφού τα κέρδη (πυκνότητα, απόδοση) υπερτερούσαν σε σχέση με την πρόσθετη πολυπλοκότητα.

Τα σημερινά **tradeoffs** είναι διαφορετικά:

- Deep submicron techn. ⇒ τεράστιος αριθμός πυλών ⇒ το μέγεθος της μνήμης on-chip κάνει το μέγεθος του pipeline να δείχνει πολύ μικρό.
- Οι καθυστερήσεις πλέον οφείλονται σε καθυστερήσεις διασύνδεσης οπότε το κέρδος σε ταχύτητα που έχουμε με τα latches εκμηδενίζεται.
- Το κόστος της αυξημένης πολυπλοκότητας των σχεδιασμών με latches αυξάνει με επιδράσεις στο μέγεθος και τον χρόνο του σχεδιασμού καθώς και στην ανάγκη για επαναχρησιμοποίηση.

## Θέματα χρονισμού και σύνθεσης.

---



Σχ 3-1

Ο χρονισμός ενός latch δημιουργεί προβλήματα στην χρονική ανάλυση του κυκλώματος.



Η χρήση των latches δεν είναι κατάλληλη για τα SoCs.

## Θέματα χρονισμού και σύνθεσης.

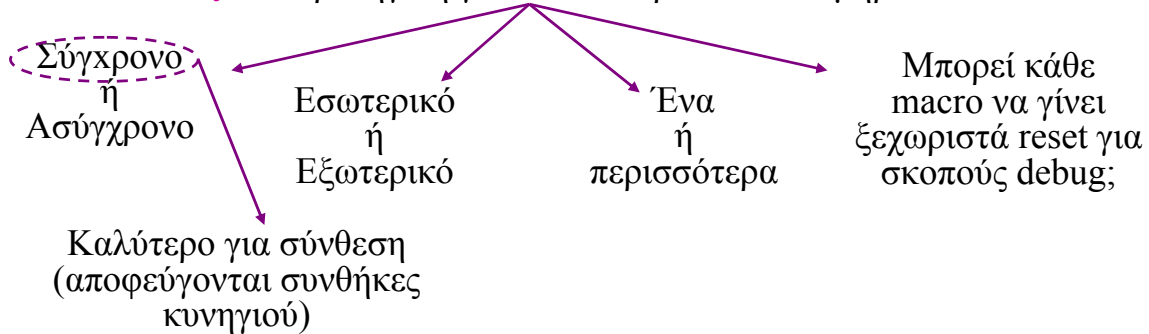
---

### *Clocking*

**Κανόνας.** Κάθε διαφορετικό domain ρολογιού που χρησιμοποιείται και κάθε συχνότητα θα πρέπει να τεκμηριώνεται πλήρως. Ειδικά πρέπει να δίνονται χρονισμοί όπως setup/hold.

### *Reset*

**Κανόνας.** Η στρατηγική για το reset πρέπει να τεκμηριωθεί.



## Θέματα χρονισμού και σύνθεσης.

---

### *Στρατηγική σύνθεσης και προϋπολογισμός χρονισμών*

**Κανόνας.** Οι γενικοί στόχοι για χρόνους, επιφάνεια και κατανάλωση θα πρέπει να τεκμηριωθούν πριν το macro επιλεγθεί ή σχεδιαστεί. Η μέθοδος σύνθεσης θα πρέπει να σχεδιαστεί αρκετά νωρίς.

### *Προτάσεις*

- Είναι προτιμότερη η στρατηγική bottom-up.
- Κάθε macro πρέπει να έχει το δικό του script σύνθεσης προκειμένου να επιτύχει τους απαιτούμενους εσωτερικούς χρονισμούς.
- Η σύνθεση σε επίπεδο chip είναι απλή σύνδεση των macros και καθορισμός των μεγεθών των απομονωτών εισόδου/εξόδου για οδήγηση κυκλώματος.

## Θέματα σχεδιασμού λειτουργίας.

---

### *Ενδοδιασύνδεση συστήματος και On-Chip Buses*

**Κανόνας.** Ο σχεδιασμός του σχήματος busing που θα διασυνδέσει τα διάφορα τμήματα του σχεδιασμού πρέπει να αποτελεί αναπόσπαστο τμήμα της επιλογής των macros και της διαδικασίας σχεδιασμού, και πρέπει να γίνει νωρίς για να μην προκύψουν συγκρούσεις που θα θέσουν σε κίνδυνο τον σχεδιασμό.

**Οδηγία.** Υπάρχουν πολλές διαφορετικές στρατηγικές για διαφορετικά είδη blocks ενώ οι μικροεπεξεργαστές και οι μικροελεγκτές έχουν συγκεκριμένα interfaces.

Είναι προτιμότερα τα macros με παραμετροποιημένα interfaces. Ιδιαίτερα ευέλικτα είναι τα interfaces των μνημών FIFO.

## Θέματα σχεδιασμού λειτουργίας.

---

### *Δομές εκσφαλμάτωσης*

**Κανόνας.** Η στρατηγική εκσφαλμάτωσης πρέπει να αποφασιστεί νωρίς κατά την διαδικασία σχεδιασμού. Η πιο αποτελεσματική στρατηγική περιλαμβάνει την εισαγωγή συγκεκριμένων δομών που αν γίνει νωρίς τότε μειώνεται το κόστος και ο χρόνος σχεδιασμού τους.

**Οδηγία.** Βελτίωση των ιδιοτήτων Controllability-Observability.

## Θέματα φυσικού σχεδιασμού

---

### *Hard Macros*

**Κανόνας.** Η στρατηγική οργάνωσης, τοποθέτησης και διαδρόμησης συνδυασμών hard και soft macros πρέπει να αποφασιστεί πριν την επιλογή ή τον σχεδιασμό των macros (Τα hard macros δημιουργούν προβλήματα).

### *Clock Distribution*

**Κανόνας.** Η αρχιτεκτονική διαμοίρασης του ρολογιού θα πρέπει να αποφασιστεί νωρίς και εξαρτάται από το μέγεθος του chip, την συχνότητα ρολογιού και την χρησιμοποιούμενη βιβλιοθήκη.

**Οδηγία.** Για μειωμένη κατανάλωση χρησιμοποιείται bus χαμηλότερης ταχύτητας για την επικοινωνία τμημάτων. Μικρότερος αριθμός σημείων διαμοιρασμού του σήματος ρολογιού. Κάθε macro συγχρονίζει το ρολόι του με το bus.

## Στρατηγική επιβεβαίωσης

---

**Κανόνας.** Η στρατηγική επιβεβαίωσης σε επίπεδο συστήματος θα πρέπει να αναπτυχθεί και επιβεβαιωθεί πριν την επιλογή ή τον σχεδιασμό των macros. Η στρατηγική καθορίζει ποια εργαλεία επιβεβαίωσης θα χρησιμοποιηθούν και περιλαμβάνουν εξομοίωση γεγονότων ή κύκλου ρολογιού και/ή προσομοίωση. Επίσης καθορίζει το είδος των testbenches που απαιτούνται και που θα **αντανακλούν το περιβάλλον του συστήματος** όπου θα τοποθετηθεί τελικά το chip ώστε να δουλεύει σωστά όταν τοποθετηθεί.

# Στρατηγικές ελέγχου κατασκευής

---

## Έλεγχος σε επίπεδο συστήματος

**Κανόνας.** Η στρατηγική ελέγχου κατασκευής θα πρέπει να τεκμηριωθεί. Προτιμούνται δομές ελέγχου on-chip για όλα τα blocks. Η ανάπτυξη παράλληλων διανυσμάτων ελέγχου για μεγάλα chips δεν είναι εφικτή.

## Έλεγχος μνήμης

Κάποια μορφή BIST είναι απαραίτητη για RAM μαζί με διαδικασίες άμεσης προσπέλασης.

## Έλεγχος μικροεπεξεργαστών

Scan chain controller, full/partial scan, test vectors, boundary scan

## Διάφορα macros

Full scan (υψηλή κάλυψη με μικρή σχεδιαστική προσπάθεια)

Logic BIST  $\Rightarrow$  pattern generation and check inside macro

---

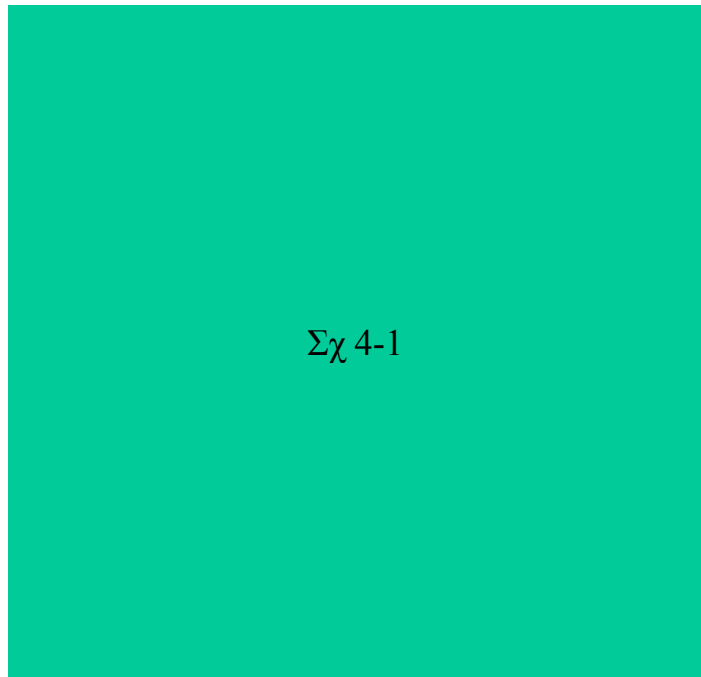
## Μέρος III

### *Η διαδικασία σχεδιασμού Macro*

## Περίληψη διαδικασίας σχεδιασμού.

---

Μετά την ανάπτυξη του συνόλου των προσδιορισμών (specifications) για τα διάφορα macros θα πρέπει αυτά να επιλεγούν από μια υπάρχουσα βιβλιοθήκη ή να δημιουργηθούν από την αρχή.

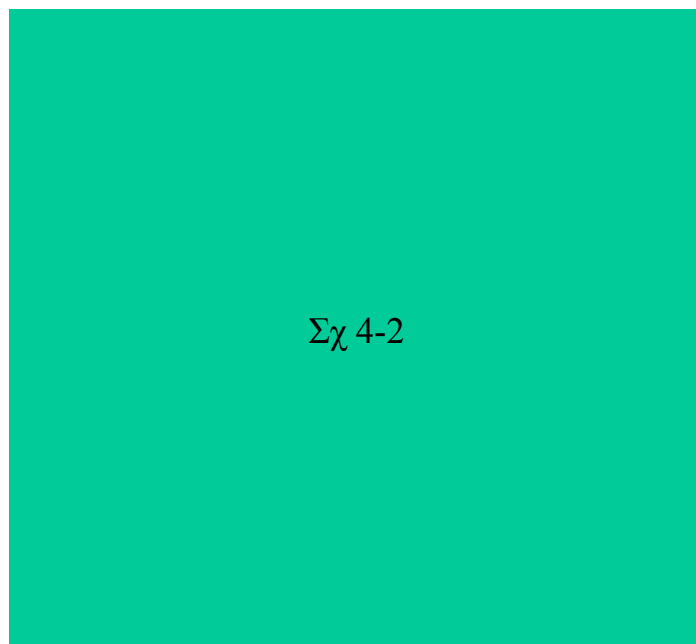


Σχ 4-1

## Περίληψη διαδικασίας σχεδιασμού.

---

Subblock integration



Σχ 4-2

## Περίληψη διαδικασίας σχεδιασμού.

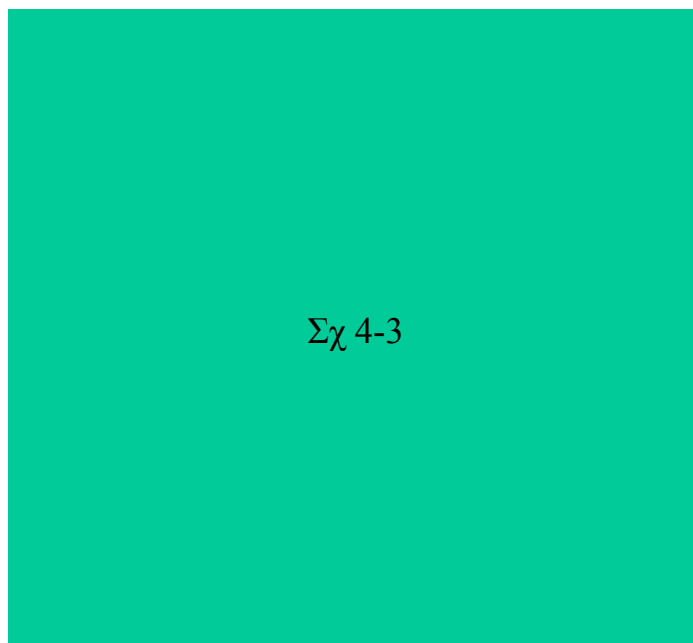
---

Τα κύρια βήματα της διαδικασίας σχεδιασμού των macros είναι:

1. **Πλήρης κατανόηση των αρχικών προσδιορισμών.** Εκλέπτυνσή τους (ανάπτυξη μοντέλου συμπεριφοράς και έλεγχος) και διαίρεση του σχεδιασμού σε μικρότερα τμήματα.
2. **Προσδιορισμός των τμημάτων και σχεδιασμός.** Ανάπτυξη λειτουργικών προσδιορισμών για κάθε τμήμα δίνοντας έμφαση στους χρονισμούς και την λειτουργικότητα του interface με άλλα τμήματα. Ανάπτυξη του κώδικα RTL, των λεπτομερών χρονικών περιορισμών, των scripts σύνθεσης και των testbenches.
3. **Έλεγχος χρονισμού.** Η ομάδα πρέπει διαρκώς να ελέγχει αν το κάθε τμήμα πληροί τους χρονικούς περιορισμούς.
4. **Integration των τμημάτων.** Παραγωγή του netlist για λειτουργικό έλεγχο και σύνθεση. Ικανοποίηση των απαιτήσεων για κατασκευαστικό έλεγχο συνήθως με ATPG και scan insertion.
5. **Παραγωγιμοποίηση.** Προετοιμασία του macro για χρήση.

## Περίληψη διαδικασίας σχεδιασμού.

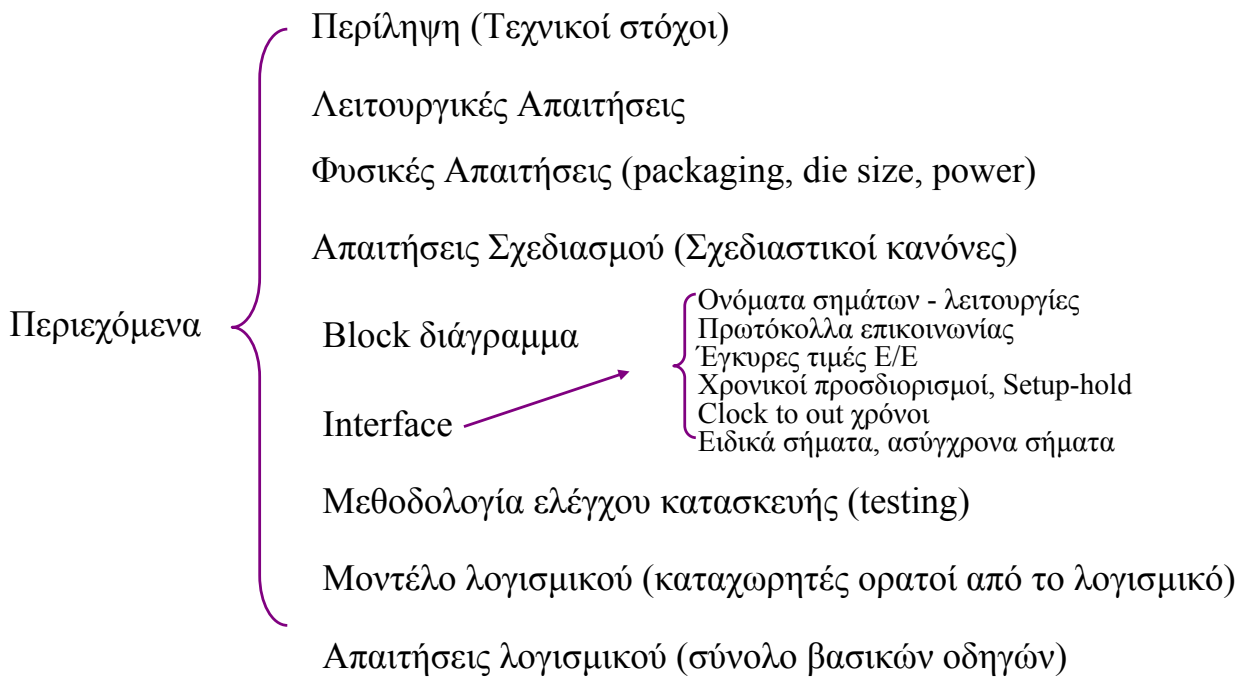
---



Σχ 4-3

## Περιεχόμενα ενός καλού προσδιορισμού.

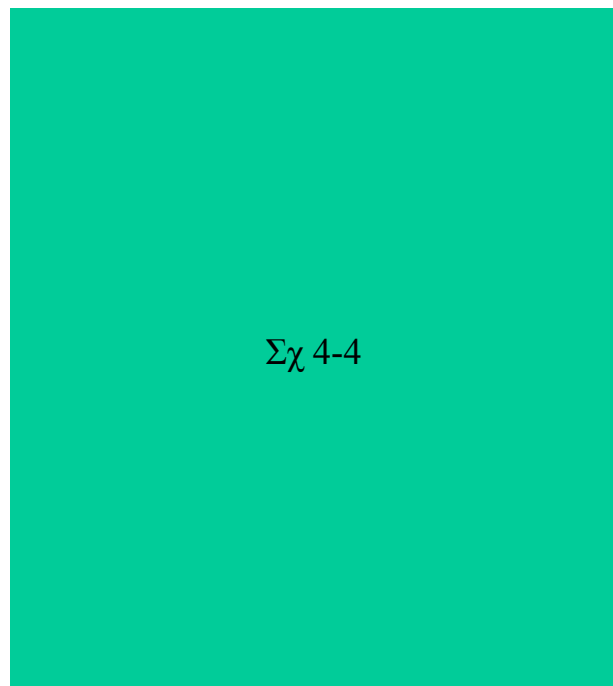
---



## Top-Level Macro Design.

---

Η πρώτη φάση σχεδίασης του macro αποτελείται από την **εκλέπτυνση των λειτουργικών προσδιορισμών** έως το σημείο που ο σχεδιασμός μπορεί να διαιρεθεί σε τμήματα αρκετά μικρά ώστε κάθε ένα να μπορεί να σχεδιαστεί, κωδικοποιηθεί και ελεγχθεί από ένα άτομο. Σημαντικό στοιχείο είναι ο σωστός προσδιορισμός των **interfaces** των τμημάτων ώστε να μπορούν μετά να ενσωματωθούν εύκολα στον σχεδιασμό.



## Top-Level Macro Design.

---

Η φάση είναι πλήρης όταν η ομάδα σχεδιασμού έχει δημιουργήσει και ελέγξει τα ακόλουθα στοιχεία:

1. Updated macro hardware specification. Ενημέρωση του προσδιορισμού του macro με την διαίρεση του στα τμήματα που επιλέχθηκαν.
2. Executable specification/behavioral model.
3. Testbench.
4. Subblock preliminary specification.

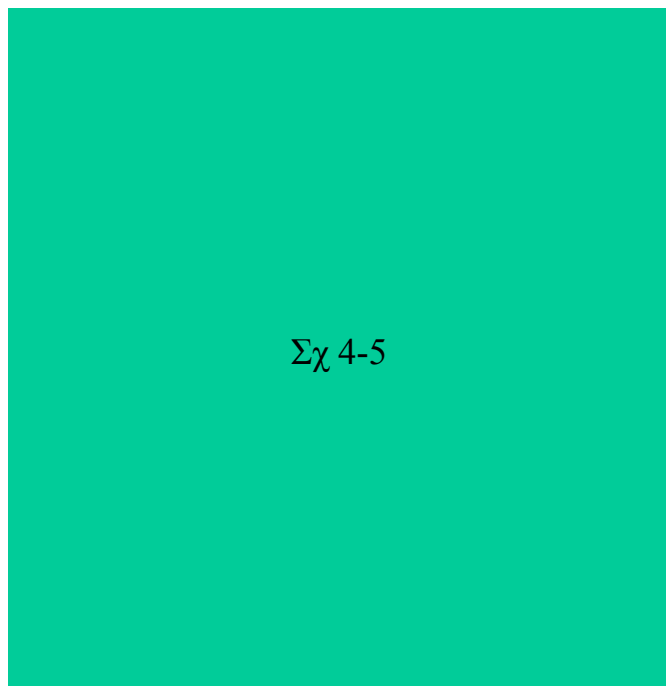
Υπάρχουν τα ακόλουθα εργαλεία και διαδικασίες:

1. Το μοντέλο συμπεριφοράς στους περισσότερους σχεδιασμούς αναπτύσσεται σε C/C++, Verilog ή VHDL. Η C/C++ είναι ιδιαίτερα χρήσιμη όταν απαιτείται σημαντικό hw/sw co-simulation πχ. επεξεργαστές.
2. COSSAP, SPW (εργαλεία stream-driven). Για σχεδιασμούς με μεγάλες απαιτήσεις σε datapath.

## Subblock Design.

---

Η δεύτερη φάση του σχεδιασμού του macro αποτελείται από τον σχεδιασμό, την κωδικοποίηση σε RTL και τον έλεγχο των subblocks του macro. Το κλειδί στην επιτυχία είναι να υπάρχει πλήρης και καθαρός προσδιορισμός για κάθε subblock πριν ξεκινήσει η κωδικοποίηση.



## Subblock Design.

---

Ο σχεδιασμός των subblocks ξεκινά όταν υπάρχει ένας προκαταρκτικός hw προσδιορισμός για το subblock και ένα σύνολο οδηγιών για το συνολικό project. Η φάση τελειώνει όταν η ομάδα σχεδιασμού έχει παράγει και ελέγξει τα ακόλουθα στοιχεία:

- Έναν ενημερωμένο hw προσδιορισμό για το subblock.
- Ένα script σύνθεσης.
- Ένα testbench για το subblock καθώς και test process με 100% fc.
- Τελικό RTL κώδικα.

Η διαδικασία περιέχει τις ακόλουθες ενέργειες και εργαλεία:

- Ανάπτυξη των λειτουργικών και τεχνικών προσδιορισμών. Οι λειτουργικοί προσδιορισμοί περιγράφουν τα σημεία του subblock που είναι ορατά στο υπόλοιπο macro: λειτουργικότητα, I/O, χρονισμός, επιφάνεια και κατανάλωση. Οι τεχνικοί προσδιορισμοί περιγράφουν το εσωτερικό του subblock.

## Subblock Design.

---

- Ανάπτυξη του κώδικα RTL.
- Ανάπτυξη του testbench. Κύριοι στόχοι είναι η αναγνωσιμότητα και η ευκολία μετατροπής.
- Ανάπτυξη των scripts σύνθεσης και σύνθεση.
- Εκτέλεση ενός εργαλείου lint για έλεγχο του RTL για παραβιάσεις κωδικοποίησης και άλλα λάθη.
- Μέτρηση της κάλυψης λαθών που παρέχει το testbench. VeriSure, VHDL Cover παρέχουν ποσοστό κάλυψης.
- Μέτρηση της ενέργειας που καταναλώνεται και επιβεβαίωση ότι είναι μέσα στα πλαίσια των προδιαγραφών.

## Macro Integration.

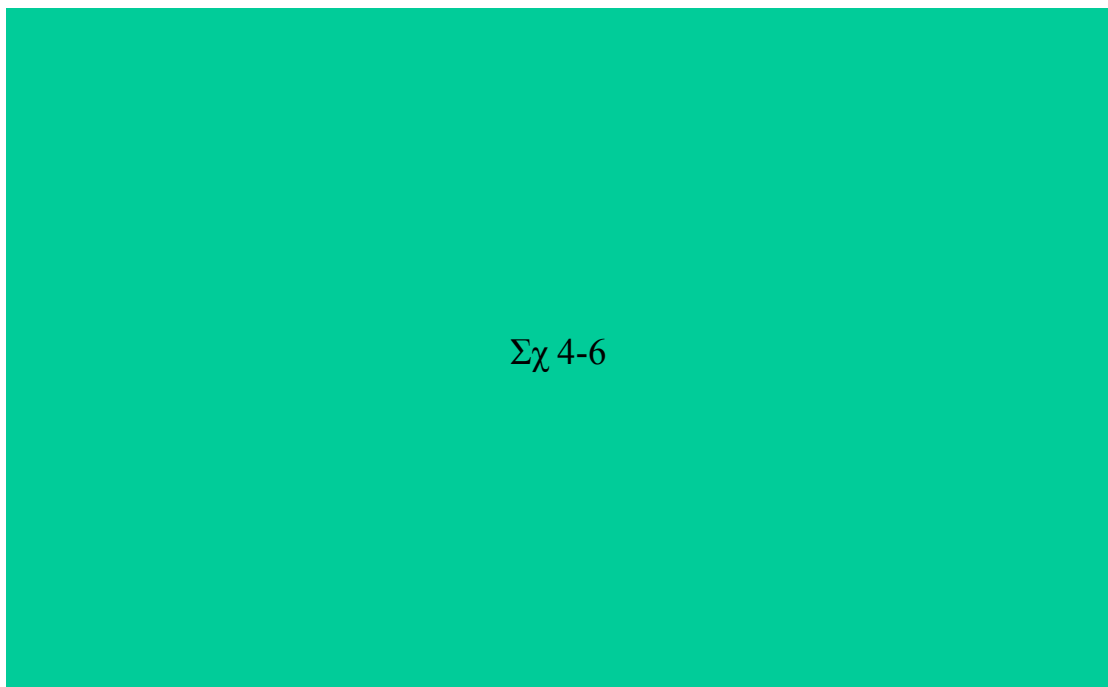
---

Η τρίτη φάση αποτελείται από την ενσωμάτωση των subblocks στο macro και εφαρμογή ενός τελικού συνόλου από ελέγχους. Απαιτεί να είναι πλήρως καθορισμένη η χρονική και λειτουργική συμπεριφορά των interfaces. Η διαδικασία ενσωμάτωσης είναι πλήρης όταν:

- Η ανάπτυξη του top-level RTL έχει ολοκληρωθεί.
- Το RTL του macro περνά επιτυχώς όλα τα test.
- Το macro μετά την σύνθεση με την δεδομένη βιβλιοθήκη ικανοποιεί όλα τα κριτήρια χρονισμού, επιφάνειας και κατανάλωσης.
- Το RTL έχει ικανοποιητική κάλυψη λαθών (lint) και σφαλμάτων κατασκευής (>95%).

## Macro Integration.

---



Σχ 4-6

## Macro Integration.

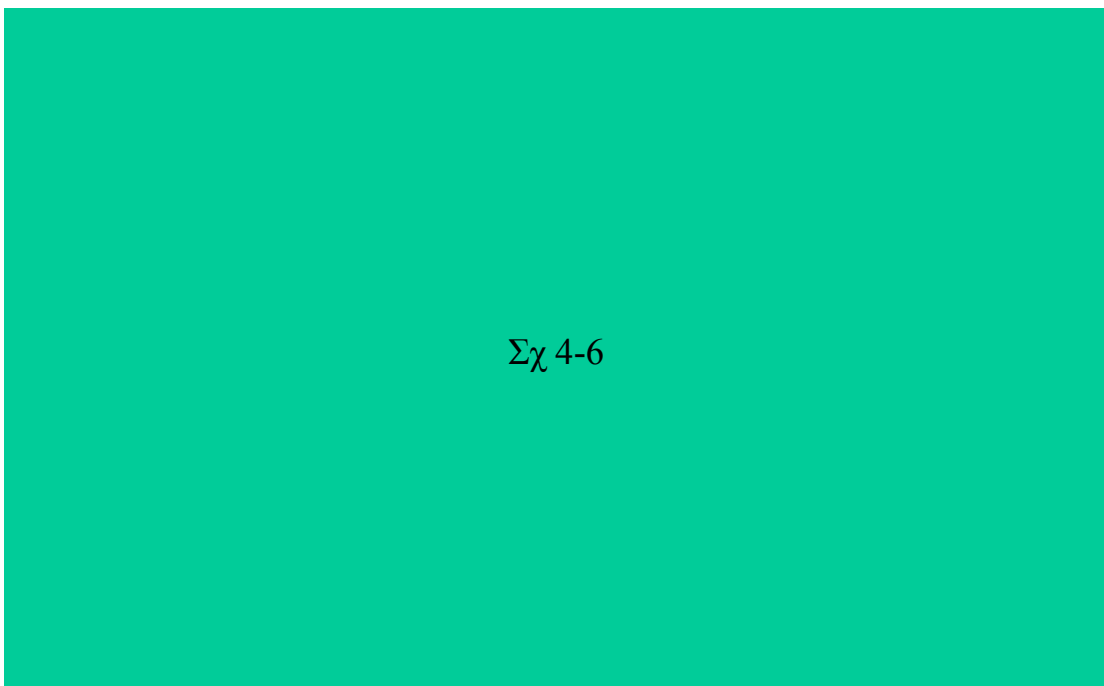
---

Η διαδικασία ενσωμάτωσης των subblocks εμπεριέχει τις ακόλουθες λειτουργίες και εργαλεία:

- Ανάπτυξη του top-level RTL. Σχηματισμός και σύνδεση των subblocks. Η παραμετροποίηση των macros όταν ο αριθμός των subblocks δεν είναι σταθερός είναι μία πρόκληση σχεδιαστική.
- Εκτέλεση λειτουργικών ελέγχων.
- Ανάπτυξη των scripts για την σύνθεση.
- Σύνθεση για όλα τα configurations.
- Scan insertion.
- Ανάλυση κατανάλωσης.

## Macro Productization.

---



## Macro Productization.

---

Είναι η τελευταία φάση σχεδιασμού και περιέχει την κατασκευή όλων των απαραίτητων στοιχείων για την επαναχρησιμοποίηση του macro (soft):

- Εκδόσεις Verilog και VHDL του κώδικα, testbenches και tests.
- Installation-Synthesis scripts που είναι απαραίτητα για την δημιουργία διαφορετικών configurations του macro.
- Τεκμηρίωση.

Δραστηριότητες και εργαλεία:

- Ανάπτυξη του πρωτότυπου chip.
- Παροχή του macro και του testbench σε Verilog και VHDL.
- Εκτέλεση σε διάφορους εξομοιωτές.
- Σύνθεση σε πολλαπλές τεχνολογίες.
- Εξομοίωση σε gate-level.