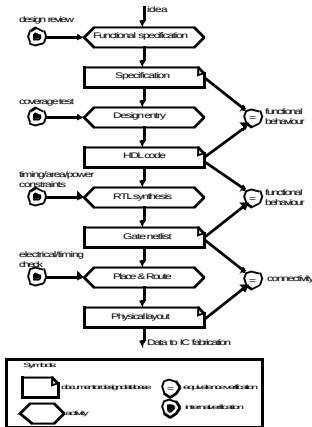


## F13: Konstruktionsflöde för VLSI chip



1 ( 4 2 )

- Målsättning
  - Presentera hela flödet från specifikation till färdig krets tillsammans med de verktyg som stödjer de ingående aktiviteterna
- Innehåll
  - Inmatning av konstruktion
  - Verifiering av konstruktion
  - Ekonomi
  - Datablad

2 ( 4 2 )

## Introduktion

- En god designstrategi är en strategi som effektivt tar konstruktionsidén till en färdig layout
- Ett bra konstruktionssystem ska sörja för att konstruktionen har en konsistent beskrivning i alla domäner (beteende, struktur, fysisk) på alla abstraktionsnivåer
- I konstruktionsprocessen hanteras stora mängder information, leder till ett stort beroende av CAD-verktygen och dess kvalitet
- Hur bra konstruktionsarbetet har genomförts kan mätas i termer av
  - Prestanda (hastighet, effektförbrukning, funktion, flexibilitet)
  - Storlek på chippet
  - konstruktionstid
  - Testbarhet
- Konstruktion innebär att en rad mål ska uppnås. En del måste uppfyllas medans andra kan kompromissas mot varandra
  - Prestanda specifikationen måste uppfyllas
  - andra mål kan vara de som är en funktion av ekonomi

3 ( 4 2 )

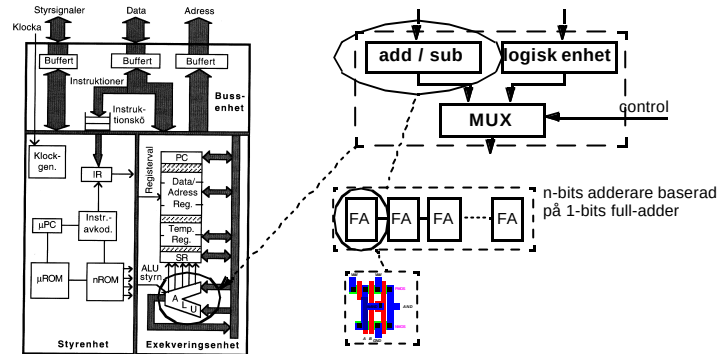
## Strukturerad konstruktion

- Strukturerande arbetsmetoder ett *måste* för att hantera komplexa konstruktioner
- Detta görs bl.a genom att vara uppmärksam på klassiska tekniker som:
  - Hierarkier
  - Regelbundenhet
  - Modularitet
  - Lokaltitet

4 ( 4 2 )

## Hierarki

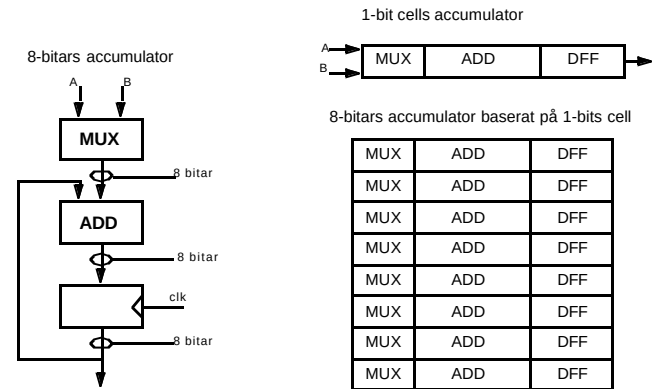
- "Divide and Conquer" - strukturering m.a.p. funktion
  - Submoduler bryts ner i submoduler tills komplexiteten blir hanterlig



5 (42)

## Hierarki

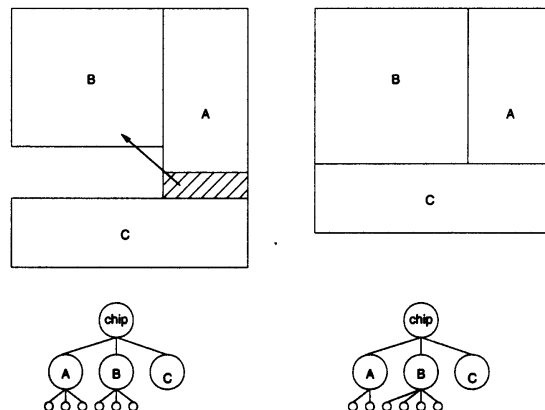
- strukturering m.a.p. regelbundenhet



6 (42)

## Hierarki

- strukturering m.a.p. fysisk layout



7 (42)

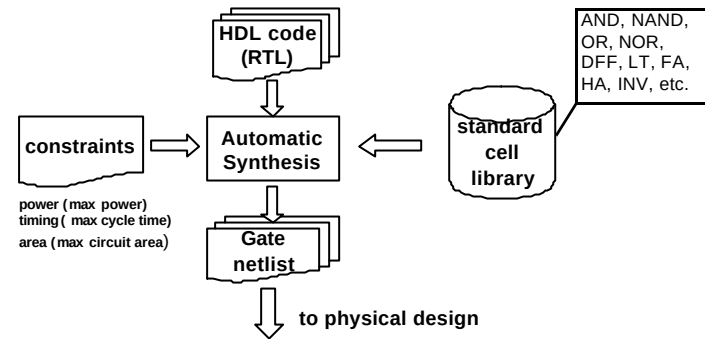
## Hierarkier

- En ren hierarkisk nedbrytning kan i slutet ge en stor mängd olika submoduler
- Med regelbundenhet i åtanke så sker nedbrytning så att submoduler kan återanvändas på olika ställen i konstruktionen
- Återanvändning är användbart på alla hierarkiska nivåer, t.ex:
  - krets-nivå - använd samma storlek på transistorerna
  - grind-nivå - använda ett litet antal logiska grindar för de kombinatoriska näten
  - arkitekt-nivå - använd identiska processelement
- Regelbundenhet syftar till att öka produktiviteten genom återanvändning

8 (42)

## Standard-cell konstruktion

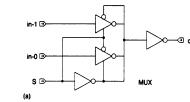
### • Typiskt konstruktionsflöde



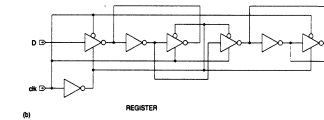
9 (42)

## exempel på återanvändning av celler

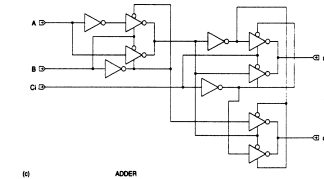
### MUX



### DFF



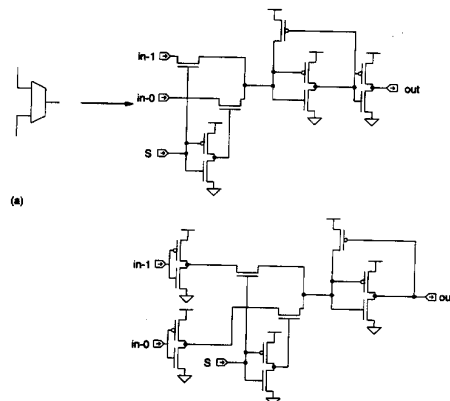
### Full-adder



10 (42)

## Modularitet

Signalvärde på en insignal påverkar kretsens karakteristik för andra ingångar.



11 (42)

### • Modularitet innebär att en submodul har

- väldefinierad funktion
  - definieras på ett otvetydigt sätt
- väldefinierat gränssnitt, exempel på fysiskt gränssnitt
  - position
  - lagertyp
  - storlek
  - signaltyp

### • Modularitet hjälper

- konstruktören att använda submoduler som tidigare har konstruerats utan att behöva sätta sig in i submodulens interna funktion (black-box)
- konstruktören att dokumentera sin problemlösning
- konstruktörer att kommunicera med varandra

12 (42)

## Lokalitet

- Lokalitet är att göra en modul så oberoende av andra moduler som det är möjligt
  - T.ex hålla variabler lokala för att slippa hämta värden från register långt borta.

## Summering

|                | Programvara                                            | Hårdvara                                                                  |
|----------------|--------------------------------------------------------|---------------------------------------------------------------------------|
| Hierarki       | sub-rutiner, bibliotek                                 | moduler                                                                   |
| Regelbundenhet | ireation, dela kod, objektorienterat tillvägagångssätt | datapaths, återanvändning av moduler, regelbunda arrayer, standard celler |
| Modularitet    | väldefinierade subrutin gränssnitt                     | väldefinierade modulgränssnitt, timing- och lastdata för celler           |
| Lokalitet      | lokal räckvidd för variabler (inga globala variabler)  | lokal kommunikation, register på in/ut-gångar                             |

## Inmatning av konstruktion

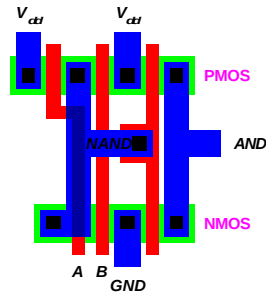
- HDL konstruktion (HDL=Hardware Description Language)
  - Beskriver: Bettende och struktur
  - Format: Text (grafiska verktyg kommer)
  - Exempel på HDL: VHDL, Verilog, ELLA
- HDL innehåller konstruktioner som finns i högnivå som C och Pascal (villkor, sekvens, repetition) + stöd för hårdvarumodellering

## Schemainmatning

- Beskriver: Struktur genom ett elektriskt schema
- Format: Grafiskt m.h.a en interaktiv grafisk editor
- Fördel:
  - en grafisk bild kan i vissa fall säga mer (snabbare) än HDL-kod, speciellt för strukturer
- Nackdel:
  - svårt att utläsa beteendet på en krets. T.ex en tillståndsmaskin med DFF och logiska grindar

## Inmatning av layout

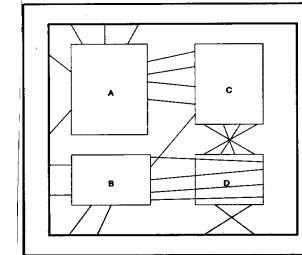
- Beskriver: fysisk layout
- Format: Grafiskt (interaktiv grafisk editor) eller kod (layout generatorer)
- Layout editorer är ofta kopplade till DRC program för interaktiv kontroll av design regler



17 (42)

## Floorplanner

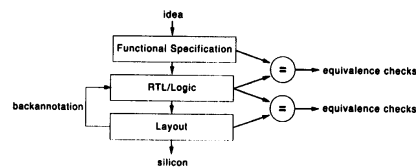
- Beskriver: placering och sammankoppling av moduler på top-nivå (fysisk struktur)
- Format: Grafiskt (interaktiv grafisk editor)
- Används för att få en uppfattning om ledningslängder på topp-nivå samt area för hela chippet. Själva ledningsdragningen görs inte, baserar sig på estimering.



18 (42)

## Verifiering av konstruktion

- Introduktion
- Mellan varje syntessteg måste konstruktionen verifieras



19 (42)

## Simulering

- Det vanligaste tekniken att verifiera konstruktionen
- Olika typer av simulatorer finns, vilken man väljer beror på
  - krav på noggrannhet
  - maximal simuleringstid
- Simulatorer
  - kretsnivå
  - timing
  - logisk (gate-level)
  - switch-level
  - mix-mode

20 (42)

## Kretsnivå-simulering

- Arbetar på kretsnivå (med modeller för resistorer, kapacitanser MOS-transistor etc)
- Noggrann simulering
- Lång simuleringstid
- Verifiering av små kretsars prestanda
- Begränsningar i noggrannhet pga:
  - onoggrannhet i MOS-modellens parametrar
  - användning av felaktig MOS-modell
  - onoggrannhet i parasit- kapacitanser och resistanser
- Exempel på kommersiell programvara: HSPICE

21 (42)

## Timing-simulering

- Genom att förenkla MOS-modellen kan simuleringstiderna minskas
- MOS-strömmarna tas fram genom 'look-up' tabeller
- Minskad noggrannhet i jämförelse med SPICE (10-20% fel)
- Exempel på kommersiell programvara: Powermill

22 (42)

## Gate-level simulator

- 'Digital' simulering
- Logiska primitiv modeller (som not, and, nand, or, nor, dff)
- Optimerad för korta simuleringstider eller komplexa konstruktioner
- Funktionell verifiering med enhets-fördröjning
- Timing verifiering baserat på:

$$T_{gate} = T_{intrinsic} + C_{load} \times T_{load}$$

- Tillräckligt noggrann för hörnsimuleringar
- Passar standard cell konstruktion med statisk logik
- Exempel på kommersiell programvara: Verilog-XL, QuickHDL

23 (42)

## Switch-Level simulator

- Beroende på logiskt värde modelleras en CMOS-grind som pull-up eller pull-down resistor till  $V_{DD}$  eller  $V_{SS}$
- Utifrån kanalresistans och kapacitanser beräknas stig- och falltider
- Icke standard CMOS kretsar kan vålla förvirring för simulatoren, t.ex TG-baserade lösningar

24 (42)

## Mixed-mode simulering

- En mixed-mode simulator kan samtidigt hantera
  - funktionell simulering
  - logisk simulering
  - switch-level simulering
  - krets nivå simulering
- Exempel på ett system med analog/digital elektronik
  - A/D omvandlare (krets nivå simuleras)
  - RAM (funktionell simulering)
  - "full-custom" CMOS logik (switch-level simulering)
  - standard cell logik (logisk simulering)
- Olika delar i systemet kräver olika noggrannhet

25 (42)

## Timing verifierare

- Verifiering av timing med simulering
  - funktionell verifiering med enhets-fördröjning
  - timing verifiering för grindar med 'worst-case' fördröjningar
  - Fördröjningen i den kritiska vägen fås fram med ett testmönster som kan visa denna (den *simulerade* kritiska vägen är beroende av testmönster)
- Timing verifierare
 

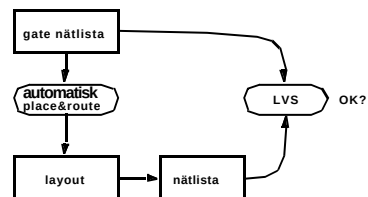
Med en timing verifierare undgår man problemet att ta fram testmönster som visar den kritiska vägen. Istället går den igenom alla signalvägar i konstruktionen och kontrollerar att den längsta vägen uppfyller timingkravet. Detta kallas **statisk timing analys**.

  - Exempel på kommersiell programvara: Design Compiler (Synopsys)

26 (42)

## Nätlist isomorfism (LVS=Layout versus schematic)

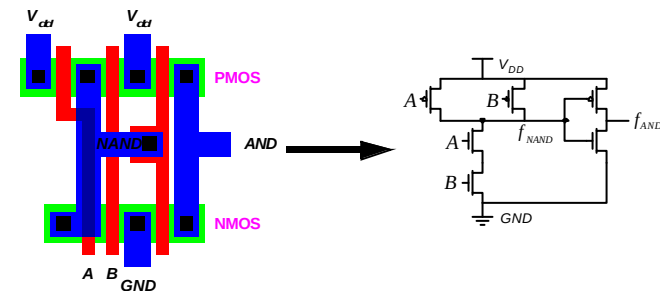
- Vid automatisk syntes från ett nätlisteformat till ett annat så ska man kontrollera att inga fel införs av verktyget
- Två elektriska kretsar är identiska om de grafer de representerar är isomorfisk
  - varje graf har samma antal komponenter
  - varje komponent i den ena kretsen har en motsvarande komponent i den andra



27 (42)

## Layout extrahering

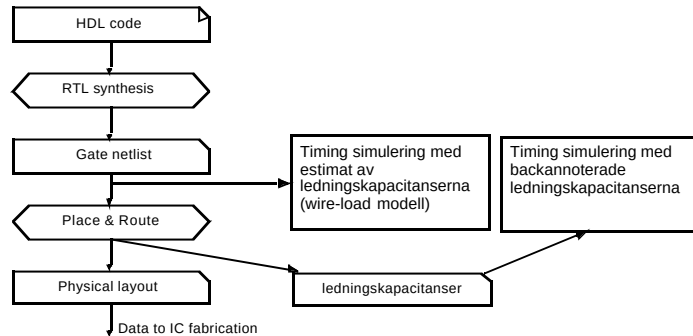
- Utifrån layout extraheras komponenter och deras kopplingar (en nätlista) baserat på kombination av lager
- Det extraherade nätlistan används i LVS och för backannotering av parasitkapacitanser



28 (42)

## Backannotering

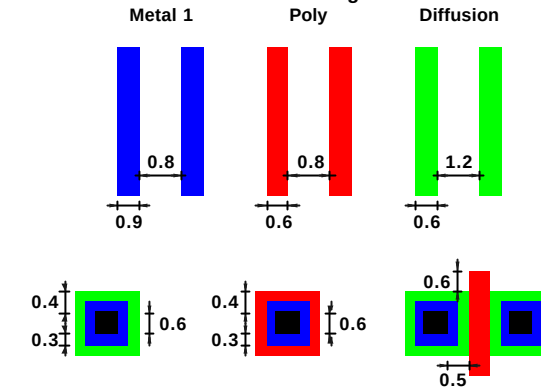
- För att få "realistiska" fördröjningar i gate-level simulering återförs data om parasitkapacitanser till simulatoren.
- Backannotering görs också till statisk timing analys



29 (42)

## Design Rule verifisering (DRC)

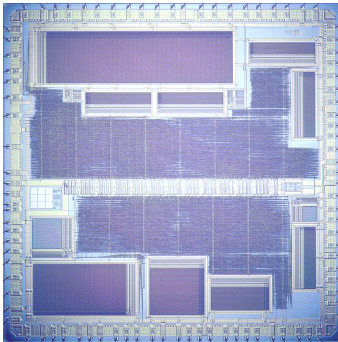
- Oavsett om layouten har skapats automatiskt med verktyg eller manuellt ska den kontrolleras av en Design Rule Checker.



30 (42)

## Automatisk Place & Route

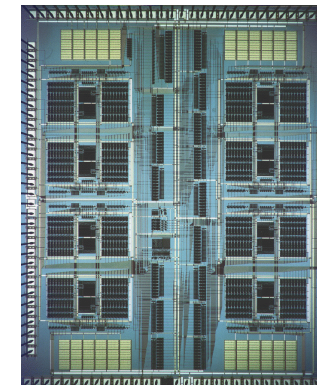
- Konstruktionen är fta hierarkiskt utplattad innan layout.



31 (42)

## Manuell layout av ett chip

- Oftast bibehållen hierarki (strukturerad layout)



32 (42)



## Ekonomi

- Utvecklingskostnad
  - konstruktion (design hus)
  - Produktion (process hus och test hus)
- Försäljningspris för komponent

$$S_{total} = \frac{C_{total}}{1-m}$$

$C_{total}$  = den totala tillverkningskostnaden  
 $m$  = önskad vinstmarginal

- Kostnad för att ta fram en IC består av
  - Nonrecurring costs (NREs) - engångskostnad
  - Recurring costs - kostnader som kommer med varje komponent
  - Fasta kostnader - dokumentation, support, marknadsföring, försäljning mm

## NREs

- Engångskostnader under utveckling av IC
  - utvecklingskostnader för konstruktionsarbetet
  - kostnader för prototypframtagning

$$F_{total} = E_{total} + P_{total}$$

$E_{total}$  = kostnader för konstruktionsarbetet

$P_{total}$  = kostnader för prototypframtagning

## Kostnader för konstruktionsarbete

- Om allt går väl första gången så är det en engångskostnad
  - Personalkostnader
    - arkitekturkonstruktion
    - inmatning och funktionell verifiering av logisk implementering
    - layout av moduler och chip
    - verifiering av timing
    - verifiering av layout (DRC och LVS)
    - Tape-out hantering
    - Testmönstergenerering
  - Driftskostnader
    - datorutrustning
    - CAD-program
    - utbildning och vidareutbildning

## Kostnader för prototypframtagning

- tillverkningsmasker
- testfixtur
- i fall då en speciell kapsel krävs : verktyg för att hantera denna

## Återkommande kostnader (recurring costs)

$$C_{total} = C_{process} + C_{package} + C_{test}$$

$C_{package}$  = kapselkostnad

$C_{test}$  = test kostnad

$$C_{process} = \frac{W + P}{NY_w Y_{pa} Y_{ft}}$$

W = wafer kostnad

P = Proceskostnad

N = antal brickor (chips) på en wafer

$Y_w$  = chip-utbyte

$Y_{pa}$  = utbyte i kapsling

$Y_{ft}$  = utbyte i sluttest

## Datablad

- Ett datablad kan vara
  - intern dokumentation
  - kommunikation mellan konstruktör och kretskortskonstruktör
  - en del i marknadsföringen av komponenten

## Innehåll i datablad

- Summering
  - Produktnamn och ett beskrivande namn
  - Kort beskrivning av vad kretsen gör
  - speciella funktioner man vill lyfta fram (=skryta med)
  - Blockdiagram på högsta nivå
- Pinout
  - pin-namn
  - pin typ
  - kort beskrivning av pinnarnas funktion
  - kapselnummer
- Kort beskrivning av kretsens funktioner

## DC specifikation

- Absolut maxvärden för
  - matningsspänning
  - signalspänningar
  - temperatur (omgivningstemperatur mest relevant för en användare)
- Varje pinnes elektriska specifikation
  - $V_{IL}$ ,  $V_{IH}$  för ingångar
  - $V_{OL}$ ,  $V_{OH}$  för utgångar
  - ingångslast på varje ingång
  - drivförmåga för utgångar
  - vilostrom
  - läckström
  - power-down ström

- AC specifikation
  - setup- och hold tider på alla ingångar
  - 'clock to output' fönster (snabbaste och långsammaste fall)
  - all annan kritisk timing
- Kapsel diagram

## Summering

- Komplexa konstruktioner tas fram genom systematik och verktyg som stödjer detta.
- Val av verktyg som simulator är en avvägning mellan önskad noggrannhet och simuleringstid.