



# Projektovanje digitalnih sistema

Uvod



## ■ Definicije

- Small Scale Integration - SSI > 10 tranzistora na čipu
- Medium Scale Integration - MSI > 100 tranzistora na čipu
- Large Scale Integration - LSI > 1K tranzistora na čipu
  - potrebni CAD/CAE alati
- Very Large Scale Integration - VLSI > 10k tranzistora na čipu
- Nota bene: Različiti izvori => različite definicije!
  
- Moore-ov zakon: Gordon Moore (suosnivač Intel-a) je 1965. godine prevideo da će se broj tranzistora koji se mogu integrisati na jednom čipu duplirati svakih 18 mjeseci
- Predviđanje se ostvarilo i tempo se drži do današnjih dana



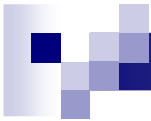
## ■ Problemi modelovanja (zašto HDL?)

- Suviše kompleksan dizajn: današnji čipovi imaju i preko 2.000.000.000 tranzistora
- Nemoguće ručno dizajnirati: po tranzistoru 10 sekundi => za dizajn čitavog sistema potrebno 1268 godina
- Nemoguće verifikovati na eksperimentalnoj pločici
- Savremeni dizajn je zasnovan na sofisticiranim alatima
  - Sistem se opisuje na visokom nivou apstrakcije (tekstualno ili grafički)
  - Nije potrebno unaprijed izabrati tehnologiju fabrikacije (alati za logičku sintezu će konvertovati dizajn za bilo koju tehnologiju fabrikacije)
  - U slučaju prelaska na novu tehnologiju nema potrebe za redizajnom
  - Alati za logičku sintezu će optimizovati kolo, i u smislu prostora i u smislu brzine rada, za novu tehnologiju



## ■ Problemi modelovanja (zašto HDL?) - nastavak

- Funkcionalna verifikacija se može obaviti u ranom stepenu razvoja čime se smanjuje vjerovatnoća pojave grešaka
- Alatima za sintezu se dobija opis na nižem nivou apstrakcije: gejtovi, tranzistori
- Uz pomoć alata, u zavisnosti od tehnologije, sistem se implementira
- Eventualno se neki veoma mali kritični dijelovi optimizuju ručno
- Tekstualni opis (sa komentarima) je najlakši način za razvoj i za otklanjanje neispravnosti
- Reprezentacija je veoma koncizna u poređenju sa šematskim prikazom
  - šematski prikaz je gotovo neupotrebljiv kod veoma složenih sistema



## ■ Zašto Verilog?

- Verilog je jednostavan i elegantan
- Njegove konstrukcije za opisivanje hardverskih elemenata su u konciznoj i čitljivoj formi
- Za upoređenje: opis istog elementa u nekom drugom jeziku za opis hardvera može biti i duplo duži
- Sa Verilogom projektant mora da nauči samo jedan jezik za sve aspekte logičkog dizajna
- Omogućava da se koriste različiti nivoi apstrakcije, pomiješani u istom modelu
- Simulacija dizajna zahtijeva funkcionalne modele, hijerarhijske strukture, testne vektore, interakciju čovjek-uređaj...
  - U Verilogu je sve ovo omogućeno jednim jezikom



## ■ Zašto Verilog? - nastavak

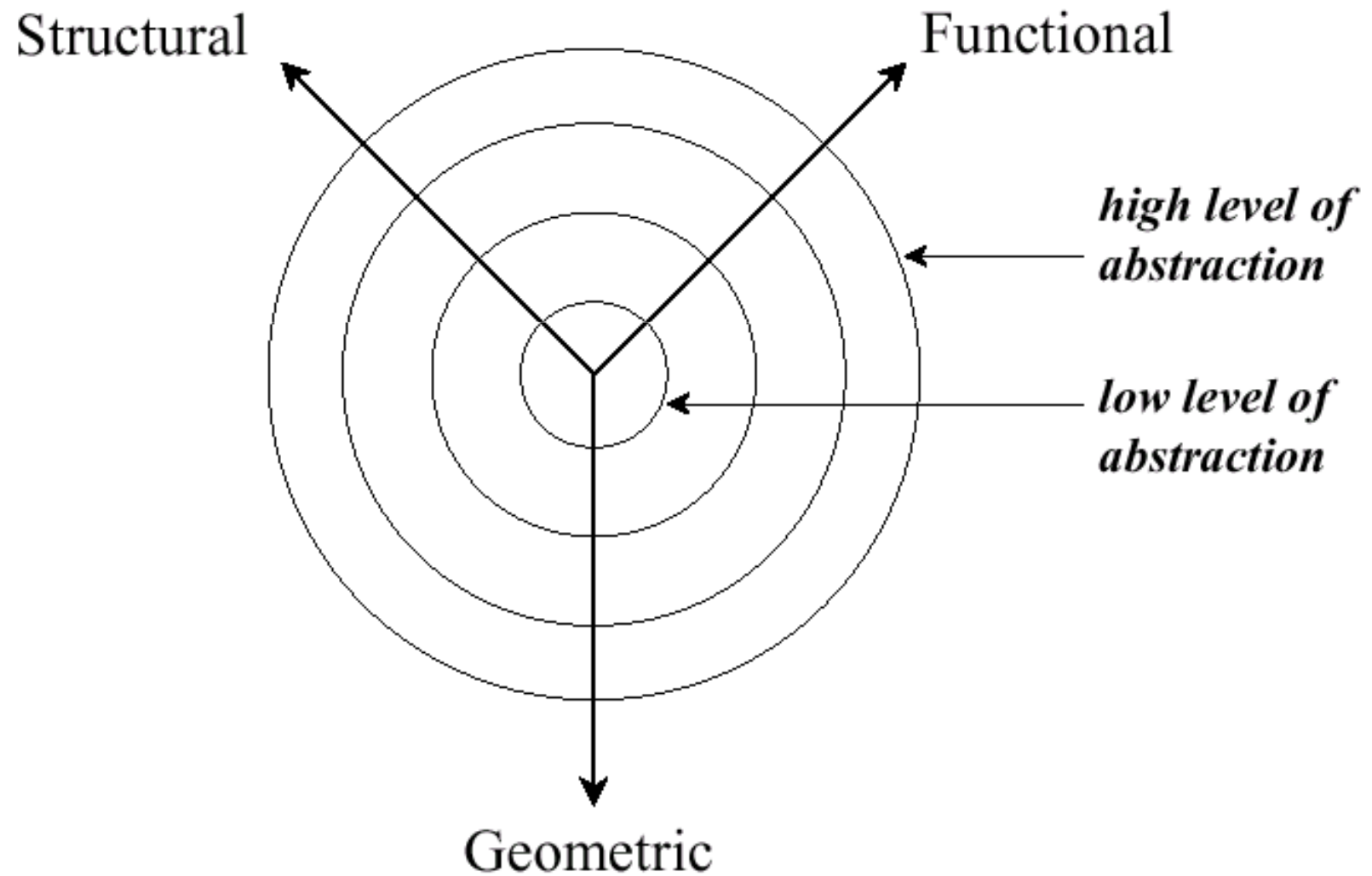
- Verilog se lako uči
  - Veoma je sličan programskom jeziku C
  - Pošto je C jedan od najviše korišćenih programskih jezika, većina projektanata bi trebalo da su familijarni sa njim i zato im je lako da nauče Verilog
- PLI (*Programming Language Interface*) je moćna funkcija koja omogućava korisniku da napiše prilagođeni C kod za interakciju sa internim strukturama podataka u Verilogu
- Dizajneri mogu da prilagode Verilog HDL simulator njihovim potrebama koristeći PLI



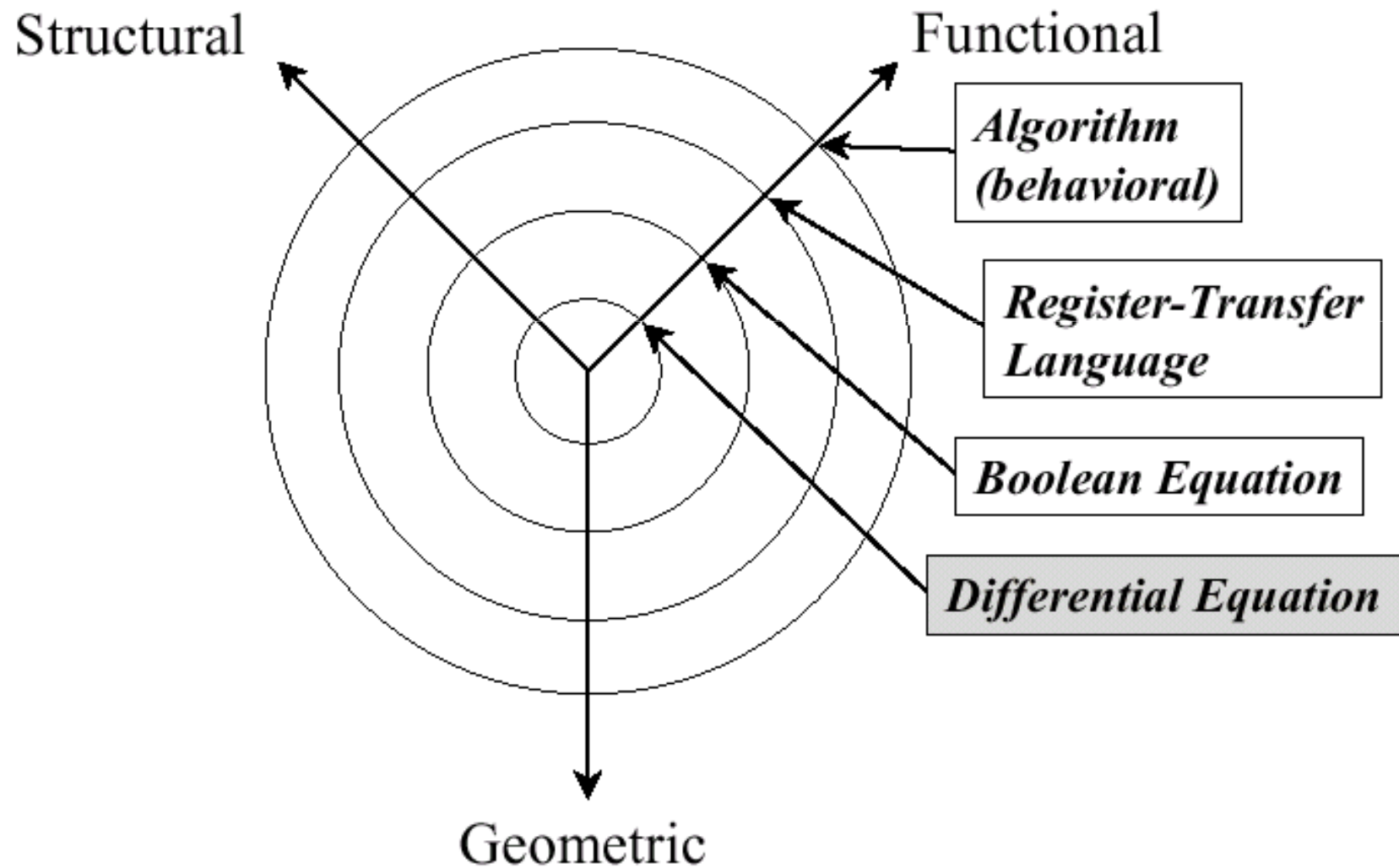
## ■ Trend u razvoju: System On a Chip (SOC)

- Čitav sistem (procesor sa potrebnim periferijama) se smješta na jedan čip:
  - kompaktniji sistem
  - jednostavniji dizajn ostatka sistema
  - jeftiniji sistem

## ■ Domeni modelovanja



## ■ Domeni modelovanja



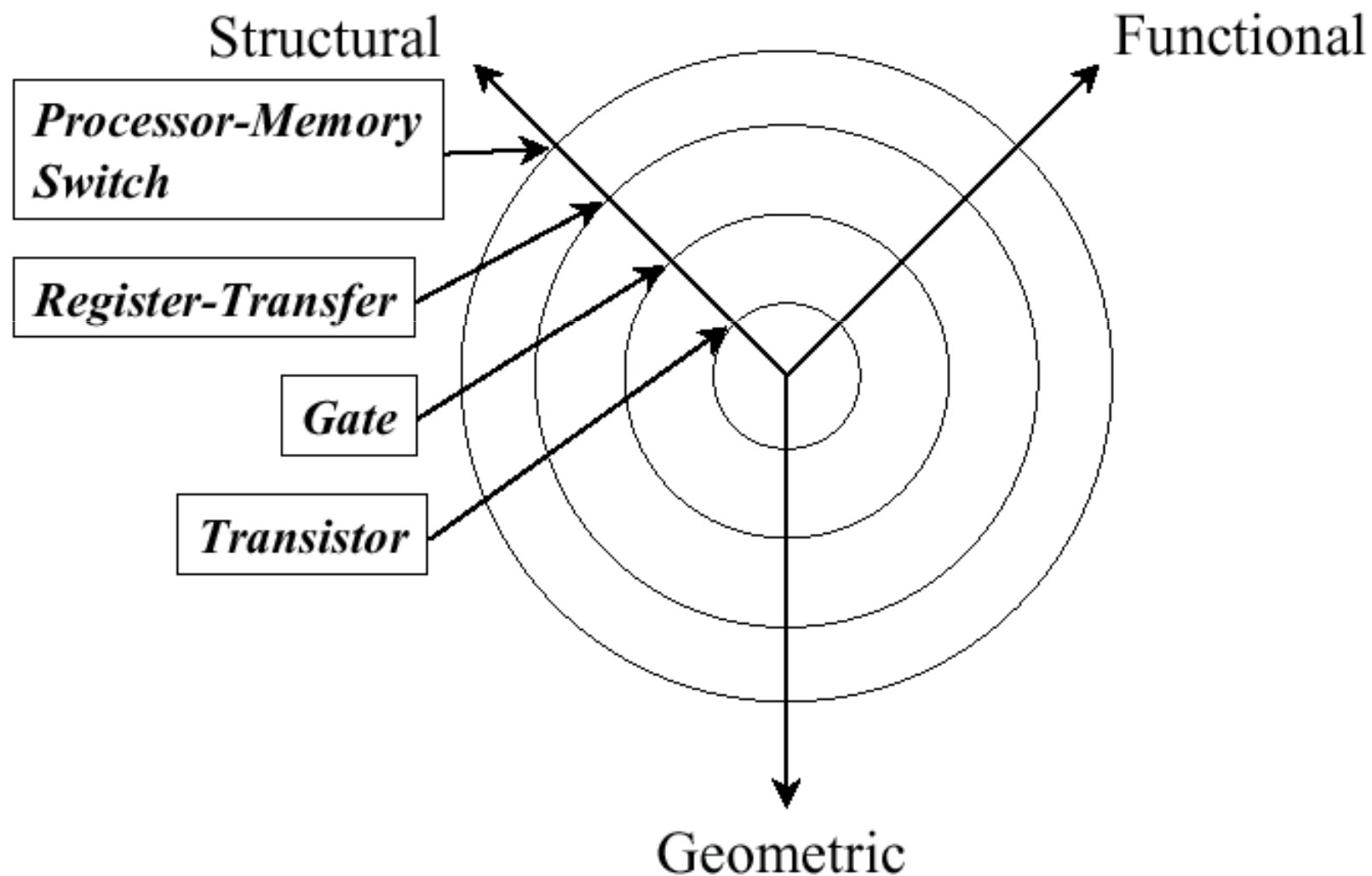


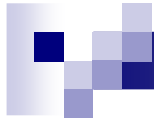
## ■ Domeni modelovanja

### ■ Funkcionalni (behavioral) domen

- Opis ponašanja sistema u funkciji ulaza i proteklog vremena
- Daje odgovor na pitanje: šta ili kako sistem radi?
- Definiše algoritam rada sistema i interfejs između sistema i okruženja
- Rezultat: funkcionalna reprezentacija u obliku dijagrama toka, programa u višem programskom jeziku, opisa u HDL-u, matematičke formule, grafikona, ...

## ■ Domeni modelovanja



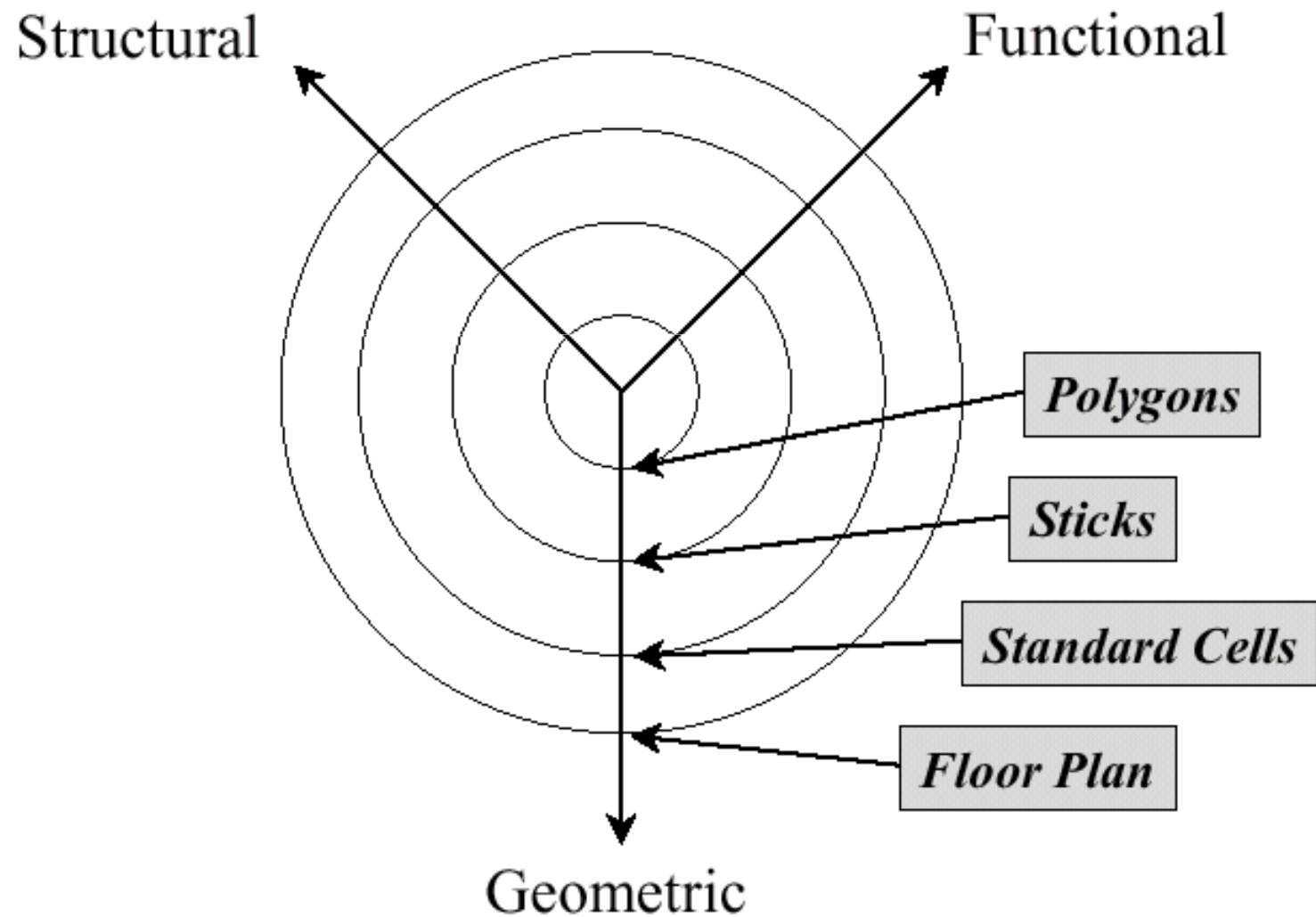


## ■ Domeni modelovanja

### ■ Strukturni domen

- Razlaže sistem na skup komponenti i njihovih veza
- Daje odgovor na pitanje: kako pomoću raspoloživih komponenti realizovati sistem zadate funkcije?
- Rezultat: strukturna reprezentacija u vidu blok dijagrama, šematskog prikaza, netliste, ...

## ■ Domeni modelovanja





## ■ Domeni modelovanja

### ■ Fizički domen

- Definiše fizičke karakteristike sistema (dimenzije i poziciju komponenata iz strukturnog opisa)
- Bavi se fizičkim raspoređivanjem i povezivanjem komponenti
- Rezultat: fizička reprezentacija u obliku *layout*-a čipa, crteža štampane ploče, ...

### ■ Konačni proizvod projektovanja na osnovu koga se može direktno realizovati sistem ili fabrikovati čip




## ■ Načini unosa dizajna

### ■ Tekstualni

- dovoljan obični tekstualni editor
- fleksibilan
- mogućnost opisa hardvera i na nivou algoritma
- pregledan
- pogodan za dokumentovanje

### ■ Grafički

- potrebni posebni alati
- nije pogodan za opis na višem nivou apstrakcije



## ■ Testiranje - Verifikacija

- Prije fabrikacije, sva testiranja se obavljaju simulacijom
- Koristi se **Test bench** model:
  - entitet bez portova
  - arhitektura koja sadrži: model koji se testira + dodatni procesi/entiteti koji:
    - generišu test vektore koji se dovode na ulaz testiranog kola
    - kontrolišu ispravnost izlaznih signala (ukoliko se provjera vrši automatizovano)
- Provjera ispravnosti izlaznih signala može da se obavlja i ručno, korišćenjem simulatora



## ■ Regresivno testiranje (regression test)

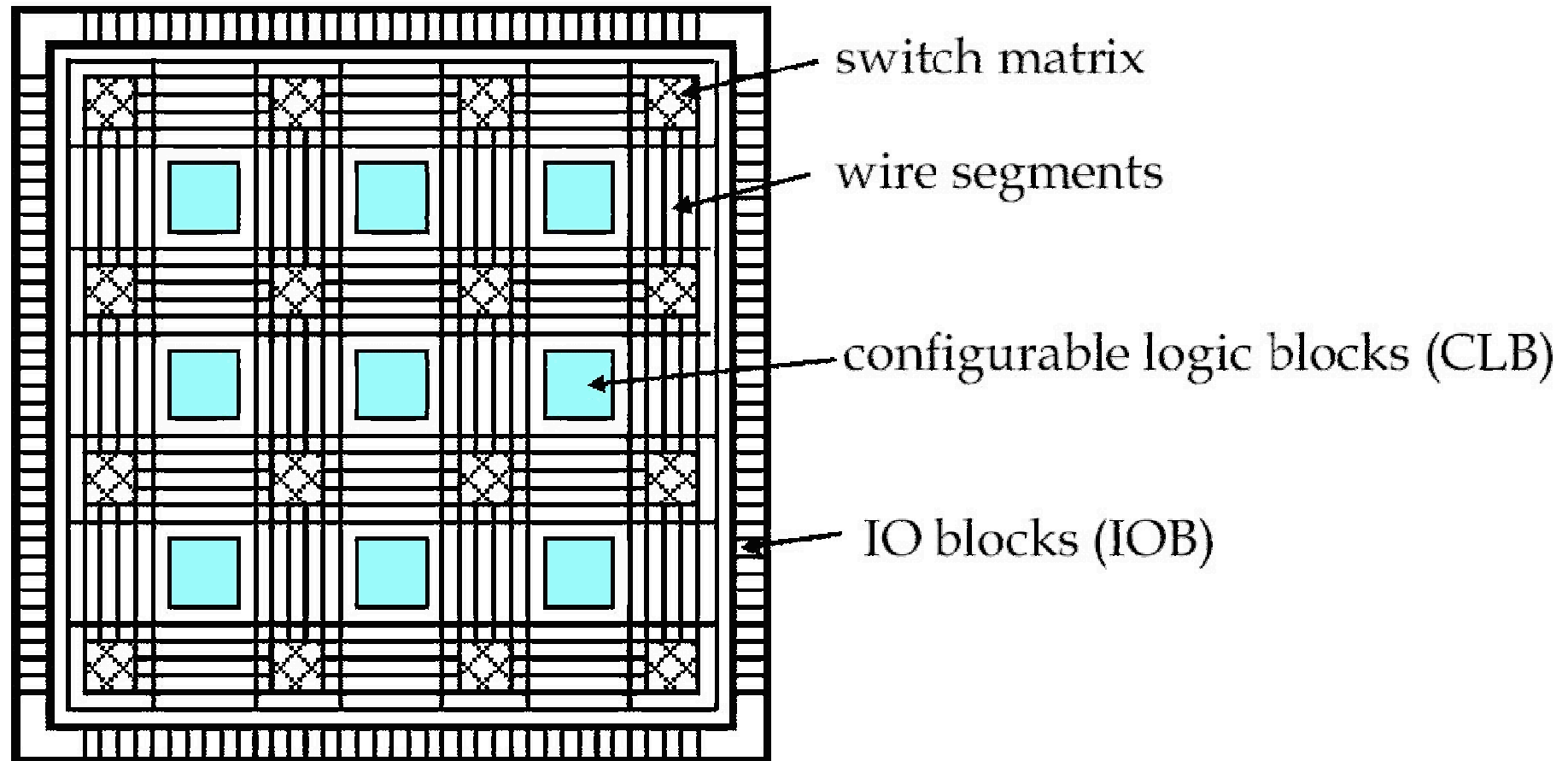
- Potrebno je potvrditi ispravnost (kasnijeg) dizajna na nižem nivou
  - **strukturni** model na niskom nivou apstrakcije treba da daje iste rezultate kao i **funkcionalni** model na višem nivou apstrakcije
- Test bench sadrži dvije instance koje se testiraju
  - strukturni i funkcionalni model
  - oba modela se paralelno simuliraju sa istim ulaznim test vektorima
  - porede se rezultati na izlazu
- Potrebno je voditi računa o vremenu
  - funkcionalni model lako može da previdi određena kašnjenja koja su posljedica strukture modela



## ■ ASIC

- ASIC = Application Specific Integrated Circuit
- Specijalno dizajniran za konkretnu aplikaciju
- Opisan u nekom HDL jeziku.
- Verifikovan korišćenjem simulatora.
- Sintetizovan korišćenjem nekog alata za sintezu
- Implementiran u FPGA, Standard Cell, Gate Array ili Full Custom tehnologiji
- Povezan sa okruženjem preko standardizovanih protokola i električnih interfejsa

## ■ FPGA = Field Programmable Gate Arrays





## ■ Šta je konfigurabilno?

- CLB

- LUT

- interno povezivanje

- IOB

- Veze između CLB

- Namjenske komponente na čipu

- množači

- memorija

- kontrola takta



## ■ FPGA – prednosti i mane

### ■ Prednosti:

- relativno jeftini
- veliki kapacitet
- mogućnost reprogramiranja čak i kada je proizvod isporučen
- kratko vrijeme razvoja

### ■ Mane:

- manji kapacitet i brzina u odnosu na integrisana kola
- potrebni sofisticirani alati



## ■ FPGA – prednosti i mane - nastavak

<b>Karakteristika</b>	<b>FPGA</b>	<b>Full-Custom</b>
Vrijeme razvoja	Kratko	Dugo
Cijena proizvoda (u velikim serijama)	Visoka	Niska
Mogućnost izmjene nakon fabrikacije	Moguće	Nemoguće
Performanse	Srednje	Veoma visoke
Gustina pakovanja	Srednja	Veoma visoka
Potrošnja energije	Visoka	Niska
Minimalna veličina serije	Jedan	Velika
Kompleksnost dizajna	Srednja	Visoka
Kompleksnost testiranja	Srednja	Visoka
Vrijeme za ispravku greške	Sati	Mjeseci

Moguće  
formirati  
proizvoljnu  
logičku funkciju

## Configurable Logic Block

Inputs

Look-Up  
Table  
(LUT)

Clock

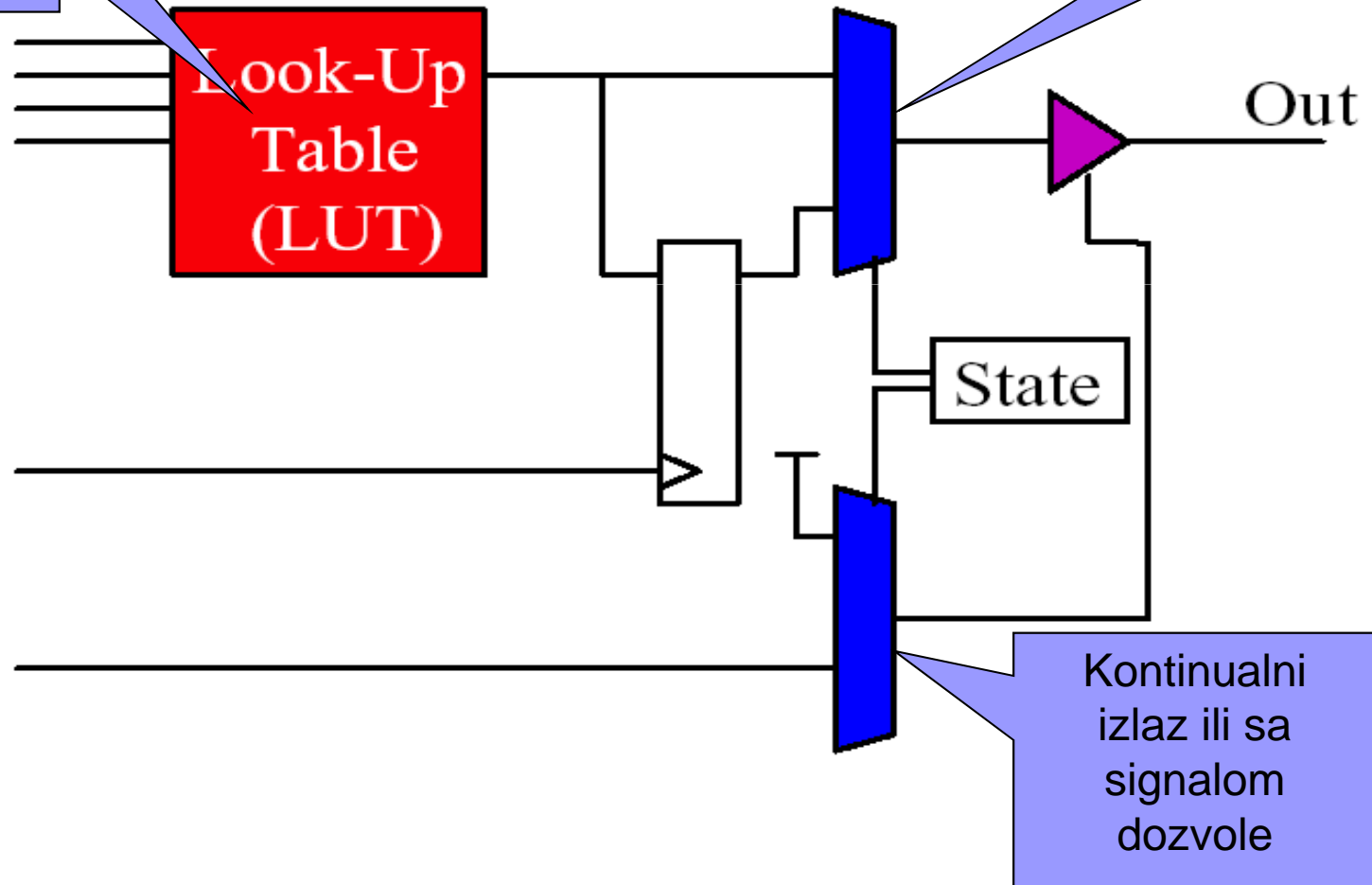
Enable

Sinhroni ili  
asinhroni izlaz

Out

State

Kontinualni  
izlaz ili sa  
signalom  
dozvole





## ■ CLB - nastavak

- U zavisnosti od konkretne implementacije mijenja se:
  - broj i veličina lut tabele
  - broj i vrsta memorijskih elemenata
  - interna kombinaciona logika (mogućnost internog povezivanja)
  - dodatne funkcionalnosti



## ■ LUT – Look-Up Table

- 1 bitna memorija
- Ulazi su vezani na adresne linije
- Izlaz je sadržaj memorije sa adrese opisane tekućim vrijednostima na ulazu
- Konstantno kašnjenje

ads	A	B	C	D	F
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0

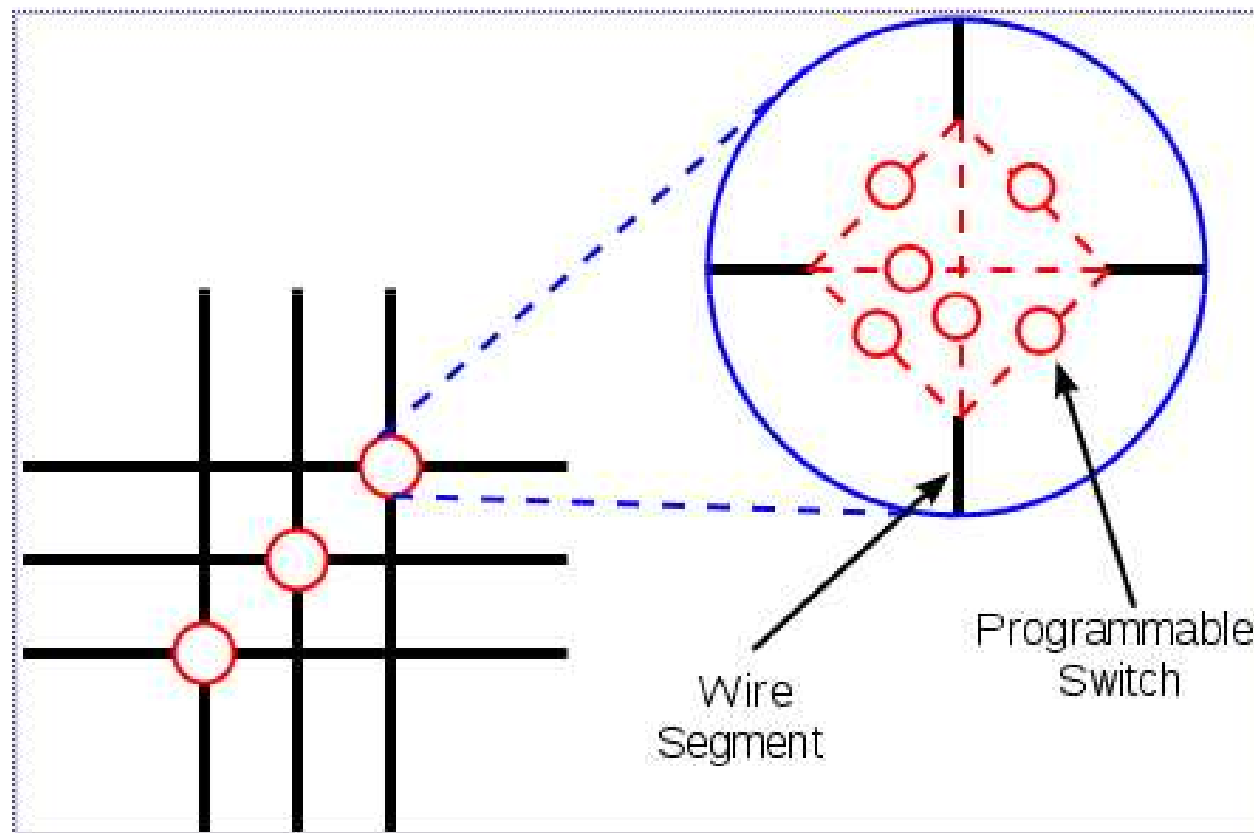


## ■ Implementacija logičkih funkcija

- Pretpostavka: n-ulazni CLB
- Funkcije se raščlanjuju na n-ulazne funkcije
- Svaka n-ulazna funkcija se mapira na jedan CLB
- Povezivanjem CLB-ova se formira tražena funkcija

## ■ Povezivanje

- 2D matrica vodova koji u presjeku imaju konfigurabilne veze





## ■ Povezivanje - nastavak

### ■ Zbog efikasnijeg povezivanja veoma često postoji više tipova veza:

- one koje povezuju susjedne elemente
- one koje povezuju prve nesusjedne elemente
- one koje povezuju svakih  $n$ -ti, gdje je  $n$  cio broj veći od 2
- krajnje elemente
- ...



## ■ IOB – Input/Output Block

- I/O interfejs je implementiran u obliku I/O blokova
- I/O blokovi su djelovi FPGA arhitekture pozicionirani periferno i povezani sa I/O pinovima kao i sa unutrašnjim interkonekcijama
- I/O blokovi su grupisani u tzv. banke - grupa od susjednih pinova koji koriste isti ili kompatibilan I/O standard u isto vrijeme
- I/O blok obično sadrži:
  - Programabilne I/O bafere (da bi se prilagodili različitim standardima)
  - D - ff (koriste se kao registri ili za eventualno unošenje kašnjenja)
  - pull-up/pull-down otpornici (postavljaju pinove na logičku nulu odnosno jedinicu, a koji bi inače imali neodređeni nivo)
  - Polja za kašnjenje (obezbjeđuju programirano kašnjenje I/O signala)
  - Kola za zadržavanje (zadržavaju posljednje stanje na magistrali ako su svi njeni priključci u stanju visoke impedanse)



## ■ IOB – Input/Output Block - nastavak

### ■ Mogućnost konfiguracije:

#### ➤ smjer:

- ulazni
- izlazni
- bidirekcioni

#### ➤ sinhronizacija:

- direktni
- sinhronizovan sa signalom takta

#### ➤ niz drugih mogućnosti (pull-up otpornici, pull-down otpornici, ...)



## ■ Memorija

- Korišćenje CLB odnosno LUT kao registara ne obezbjeđuje dovoljno memorijskog prostora i dovoljno fleksibilnosti za mnoge upotrebe
- Vremenski zavisne aplikacije (npr. real-time) koje vrše mnogo računanja zahtijevaju ugrađenu memoriju
- Glavne prednosti ugrađene memorije su:
  - kratko vrijeme pristupa
  - veliki propusni opseg
  - velika fleksibilnost (može da se ponaša kao):
    - RAM
    - ROM
    - bafer (FIFO, LIFO, FILO, ...)
    - pomjerački registar
    - ...

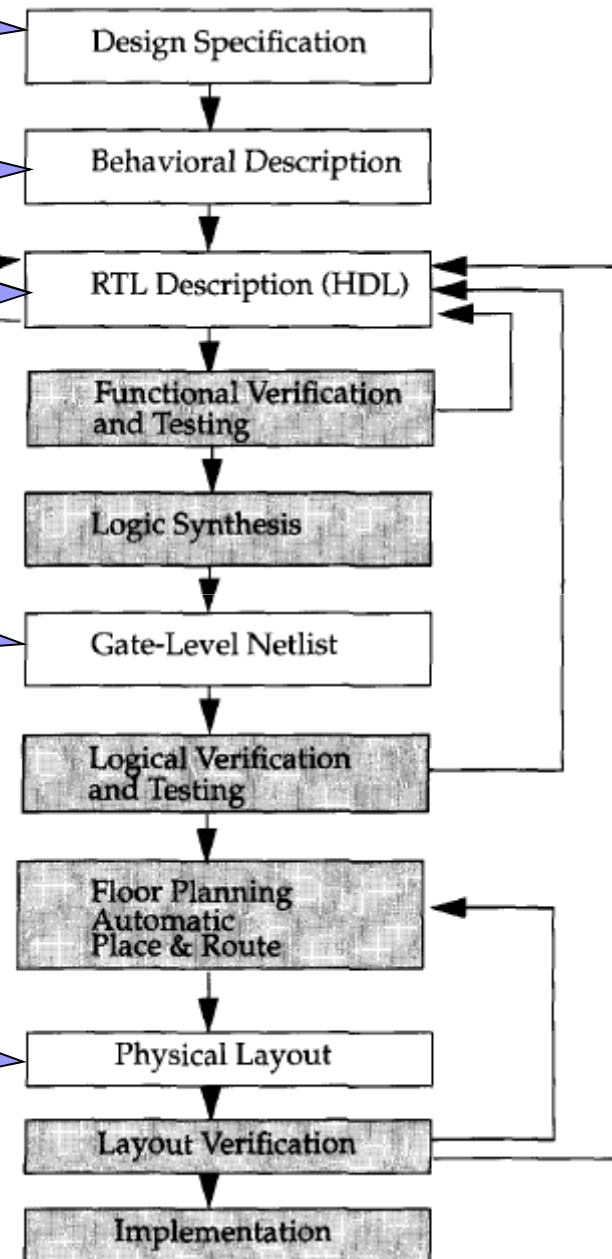
Apstraktno opisuje funkcionalnost, interfejse i opštu arhitekturu digitalnog sistema koji se

Analizira dizajn u smislu funkcionalnosti, performansi, zadržavanja i standarda i

Behavioral opis se konvertuje u RTL opis u HDL-u. Dizajner mora da opiše tzv. data-flow koji će implementirati željeno digitalno kolo. Od ove tačke pa nadalje, proces dizajna se obavlja uz pomoć CAD alata.

Alati za logičku sintezu konvertuju RTL opis u gate-level netlist-u. Ona predstavlja opis kola u smislu logičkih kapija (gate) i veza između njih.

Gate-level netlist-a predstavlja ulaz u alate za Automatic-Place-and-Route, koji kreiraju konačni *layout*. On se verifikuje i fabrikuje na čipu.



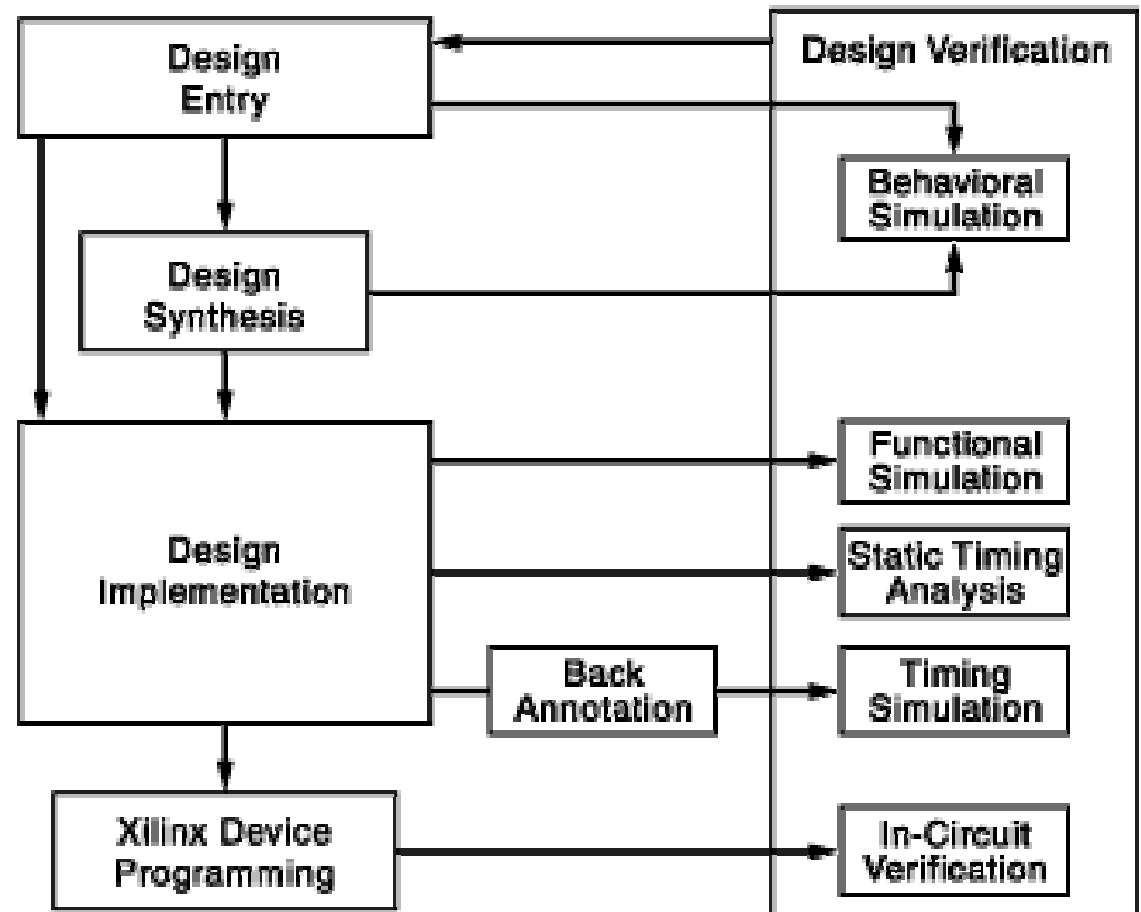


## ■ Tok procesa dizajna - nastavak

- Dakle, najveća aktivnost dizajnera je kod optimizovanja RTL opisa sistema
- U daljem procesu CAD alati rade najveći dio posla
- Dizajniranje na RTL nivou je značajno skratilo vrijeme potrebno za projektovanje
- U posljednje vrijeme se sve više koriste i alati za *behavioral* sintezu
  - Kreiraju RTL opis iz algoritamskog ili opisa ponašanja kola
  - Dizajn digitalnih kola postaje sličniji računarskom programiranju na jezicima višeg nivoa
- **Nota bene:** iako CAD alati omogućavaju automatizaciju procesa i skraćuju vrijeme dizajna, dizajner je i dalje onaj ko kontroliše kako će alati uraditi posao.
  - GIGO: Garbage IN => Garbage Out

## ■ Tok procesa dizajna (Xilinx)

- Razvoj FPGA-baziranog sistema se može podijeliti na sljedeće faze:
  - dizajn i sinteza sistema
  - implementacija dizajna
  - on-chip verifikacija





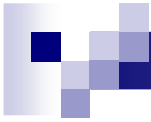
## ■ Faza dizajna sistema

- Počinje sa fazom ***unosa dizajna*** pomoću:
  - HDL – Hardware Description Language
  - schematic editor-a
- Softverski paketi obezbjeđuju integrisano okruženje za ovu fazu
- Na raspolaganju je širok dijapazon *biblioteka* sa razvijenim komponentama (jednostavni procesori, interfejsi, kontroleri, brojači, dekoderi, ..., sve do logičkih kola)
- Softver omogućava hijerarhijski unos dizajna
- Kada se završi unos dizajna, prelazi se na njegovu ***sintezu*** – proces koji ga transformiše iz HDL oblika u formu najnižih logičkih kola (tzv. RTL – Register Transfer Level)
- Ova faza je nezavisna od ciljne platforme – rezultujući RTL opis može da se smjesti u bilo koji FPGA čip



## ■ Faza implementacije

- Uobičajeno se naziva ***Place-And-Route*** faza
- Alati za implementaciju uzimaju ulaznu RTL *netlist*-u i mapiraju logiku unutar raspoloživih resursa konkretnog FPGA čipa
- Nakon toga se traže najbolje lokacije za kreirane blokove dizajna, na osnovu njihovih međusobnih veza i željenih performansi
- Na kraju se izvrši povezivanje i dodjeljuju se I/O pinovi odgovarajućim signalima
- Ova faza je zavisna od ciljne platforme jer se implementacija vrši unutar konkretnog FPGA čipa
- Zato se alati za Place-And-Route razvijaju i isporučuju od strane proizvođača FPGA čipova
- Razvijeni su da u potpunosti iskoriste prednosti konkretne FPGA arhitekture i da obezbijede optimalne performanse za dati dizajn
- Rezultat ove faze je konfiguracioni fajl koji se upisuje u FPGA – kao početna verzija dizajna



## ■ Faza on-chip verifikacije

- Ova se faza obavlja kad se dizajn upiše u FPGA čip
- Daje mogućnost dizajneru da provjeri dizajn (i traži greške) u realnom radu (okruženju)
- Koriste se posebni kablovi koji se dobijaju da razvojnim kompletima, za povezivanje FPGA sa računarom
- Pomoću njih se mogu pročitati sadržaji internih registara i memorije



## ■ "Intellectual Property" blokovi (IP)

- Kompletan dizajn nekih složenijih sistema, razvijen od strane proizvođača FPGA i optimizovan za rad na njihovim FPGA (npr: mikrokontroleri, mikroprocesori, ethernet interfejs, ...)



## ■ Napomene

- Verilog liči na C
- Međutim, **jezici za opis hardvera nisu programski jezici**
- U FPGA ne postoji mikroprocesor koji bi “izvršavao” Verilog kod
- Alati za sintezu na osnovu opisa pripremaju opis na nižem nivou
- Fajl koji se na kraju dobije, upisuje se u FPGA i na taj način se postiže konfiguracija čipa
- Za sintezu se koristi samo podskup čitavog jezika