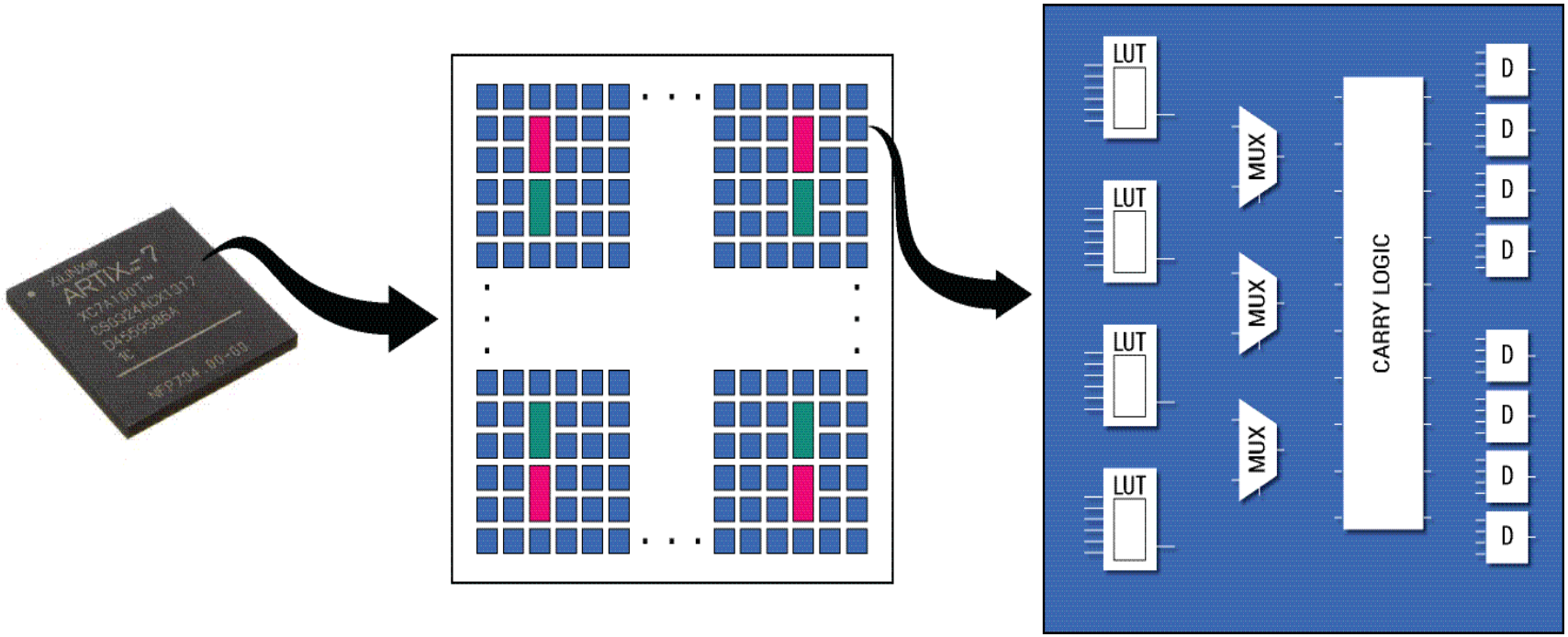
Digitala system EITA15

Elektro- och informationsteknik

VHDL

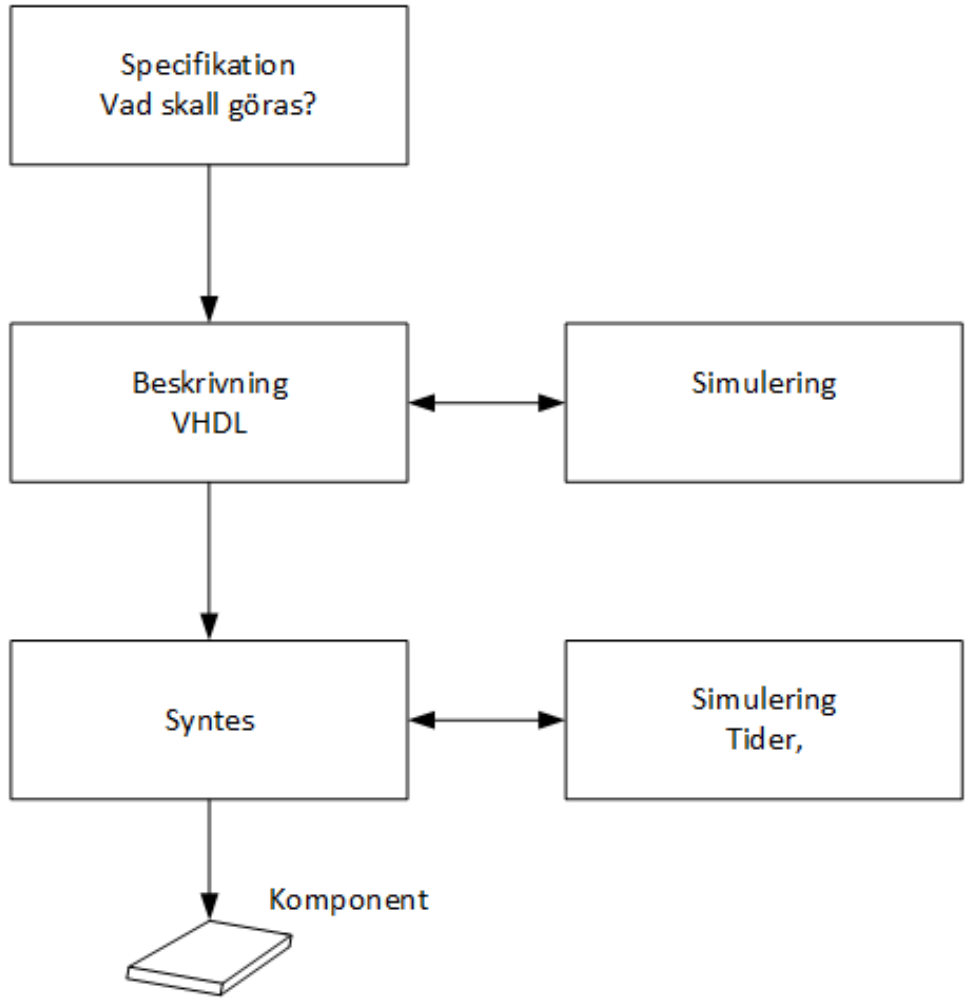
FPGA



Varje cell innehåller ett antal uppslagstabeller (Look Up Table - LUT) med flera ingångar och en utgång, ett antal multiplexrar och D-vippor. Utsignalen från en uppslagstabell är en funktion av dess insignaler.

Det finns även speciella block som hantera minne och beräkningar.

Intro till VHDL (hårdvarubeskrivande språk)



Inledning

- VHDL utvecklades för att beskriva och simulera digitala funktioner.
- Används mer och mer för att syntetisera (konstruera) digital logik.
- Verilog är ett annat beskrivningsspråk som har lånat en hel del från programmeringsspråket C.

Beskrivning i VHDL

Kretsbeskrivningen består av två delar:

Entity, Hur kretsen ser ut "utifrån"
vilket beskriver kretsens gränssnitt till omgivningen.
Portar (in- eller ut-signaler) , riktning av signaler.
SignalTyper (std-logic ……) , typ av signaler.

Architechure, beskriver hur kretsen ser ut "inuti".
Beskrivningen kan vara kretsens beteende.
eller hur kretsen är kopplad.

Syntes

- för alla C/Matlab-programmerare **Glöm allt!**
- Tänk hårdvara ---->> beskriv hårdvara med VHDL
- Genomlöps ej sekventiellt, utan samtidiga satser
- Det är hårdvara som produceras
- Vet ni inte, använd inte.

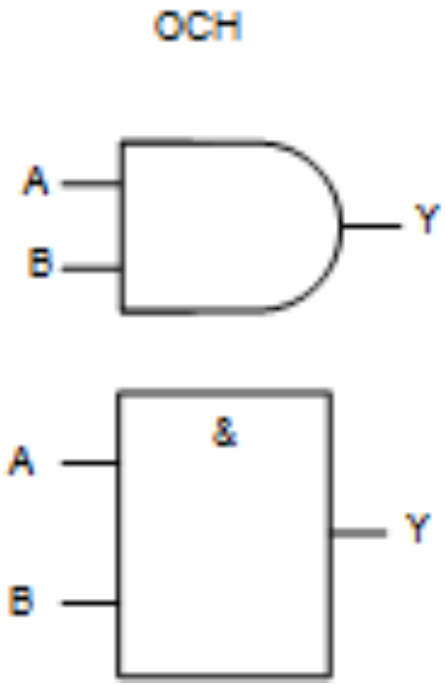
Några fördelar med VHDL

- Standardiserat språk
- starkt typat språk
- Beskrivning på flera olika abstraktionsnivåer
- Simulering
- Återanvändning
- God dokumentation
- Strukturerat arbetssätt

Lite syntax

--	kommentar till radslut
;	avslutar en sats
In / out	riktning av signal
std_logic , std_logic_vector	datatyper (vektorer)
<=	tilldela ett värde
and, not, or ….	Logiska operationer
+ - * /	Aritmetiska operationer
< > >=	Jämförelse
Unsigend (7 downto 0)	8-bitars unsigned (0 – 255)

Och-grind i VHDL (signaler)




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity och_grind is   -- A,B och Y har riktning
    Port ( A, B : in STD_LOGIC;
           Y: out STD_LOGIC);
end och_grind;

architecture Behavioral of och_grind is
 -- signal sig: std_logic;-- Intern signal = en bit tråd, ingen riktning
begin
Y<= A and B;
-- Alternativ
-- sig <= A and B;
-- Y <= sig;
end Behavioral;
```

- I vilken ordning man skriver satserna mellan begin och end spelar ingen roll.
- Satserna utföres så snart något till höger om tilldelningspilen ändrar värde.
- Alla satser betraktas som **samtidiga** (concurrent).
- All elektronik är parallell i grunden.

Vektorer, bussar

- N-bitars buss:
  std_logic_vector (N-1 downto 0)
  alternativ, std_logic_vector (0 to N-1)

Bussar kan delas upp och sättas samman

Ex (sätta samman).

```
buss1: std_logic_vector(3 downto 0)
buss2: std_logic_vector(4 downto 0)
buss : std_logic_vector(8 downto 0)

buss <= buss1 & buss2;
```


Concurrent (parallella) signaler

```
architecture Behavioral of komp is
 signal sig0, sig1 , ut: std_logic;
begin
 ut <= sig0 or sig1;

sig0 <= (not in0) and (not in1);
sig1 <= in0 and in1 ;

end behavioral;

-- ordningen på raderna har ingen betydelse
```

Conditional (villkorliga) satser

<target> <= <expression> **when** <condition> **else**
<expression>;

with <choose_exp> **select**
target <= <exp> **when** <choice>,
<exp> **when** <choice>
<exp> **when** others; täck alltid in alla övriga fall

Men, vad menar vi precis med alla övriga fall??

Process

- **process** (triggerlista)
  {typ_declaration….}
  begin
  {statement…..}
  end process;

- En process i vhdl är en funktion som beskriver speciella satser.
- En process exekveras endast om någon av triggsignalerna ändrar värde
- I princip är alla rader i vhdl processer
- If- och case-satser tillåtna