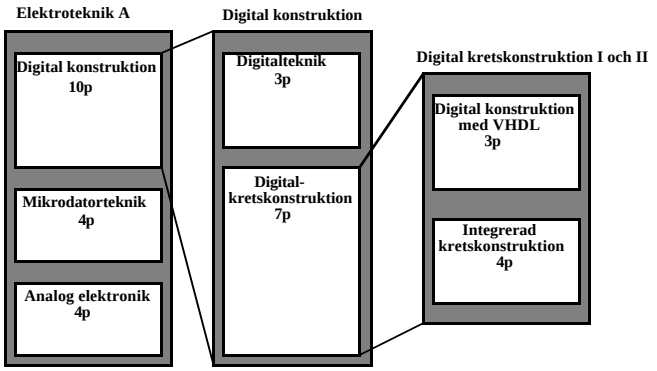
Digitalkonstruktion I 3p, vt 01

- **Kurslitteratur:**
  "VHDL för konstruktion"
  "A First Course in Digital Systems Design - An Integrated Approach"

- **Antal föreläsningar:** 7 (2h)
- **Antal laborationer:** 4 (4h)
- **Examinationsform:** Tentamen
- **Kursansvarig:** Bengt Oelmann
- **Lab. handledare:** Håkan Norell

Struktur: Elektroteknik A



Motivation och målsättning för kurserna i digital elektronik

- **Digitalteknik, 3p**
  - **Ge grundläggande kunskaper som bas för kommande konstruktionskurser**
    - **Switching algebra, formell analysteknik för digitala kretsar**
    - **Procedurer för att ta fram och manipulera kombinatoriska nät**
    - **Procedurer för att ta fram sekvensnät**

- **Digitalkonstruktion I, 3p**
  - **Konstruktionskurs med moderna konstruktionsmetoder**
    - **Grundläggande byggblock i digitala system**
    - **Hårdvarubskrivande språk (VHDL) för simulering och syntes**

- **Digitalkonstruktion II, 4p**
  - **Konstruktionskurs med inriktning på integrerad kretskonstruktion**
    - **Implementationstekniker (full-custom, standard cell, programmerbara kretsar)**
    - **analys och optimering av prestanda av IC**
    - **Konstruktionsmetoder och verktyg**

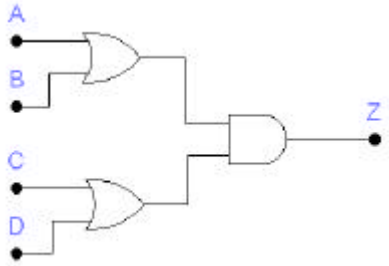
F1: Introduktion

- **Innehåll:**
  - **Kursintroduktion**
  - **Motivation till hårdvarubeskrivande språk (HDL)**
  - **Introduktion till VHDL**
    - **Syfte och bakgrund till VHDL**
    - **Konstruktionsflöde**
    - **Konstruktionsverktyg**
    - **VHDL exempel och stil**
  - **Programmerbar logik och laborationsutrustning**

Motivation till hårdvarubeskrivande språk 1(6)

- **Manuell syntes av följande logiska uttryck:**

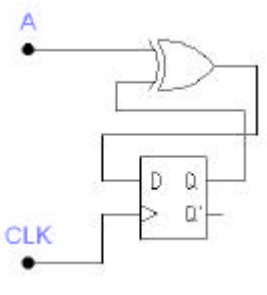
```
z = (A+B)·(C+D)
```



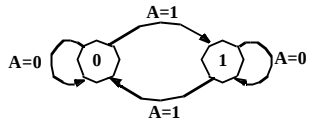
Motivation till hårdvarubeskrivande språk 2(6)

- **Analys av tillståndsmaksin**

Logiskt schema (implementering på grindnivå)



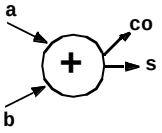
Tillståndsgraf



Motivation till hårdvarubeskrivande språk 3(6)

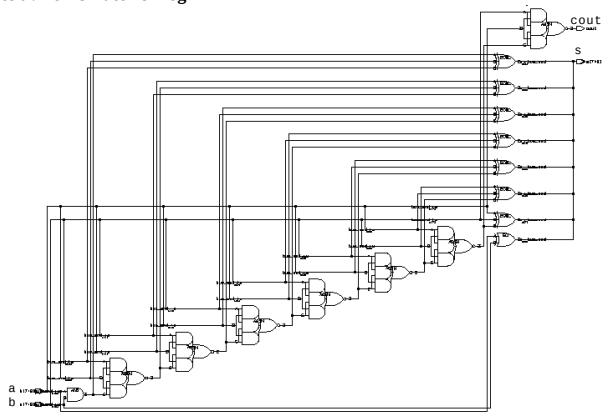
- **Beskrivning av en adderare i ett hårdvarubeskrivande språk (VHDL)**

```
Library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity add8 is
  port (
    a, b : in  std_logic_vector(7 downto 0);
    s    : out std_logic_vector(7 downto 0);
    cout : out std_logic);
end add8;

architecture rtl of add8 is
begin
  add : process (a,b)
  variable s_var : std_logic_vector(8 downto 0);

  begin  -- process add
    s_var := ('0' & a) + ('0' & b);
    s <= s_var(7 downto 0);
    cout <= s_var(8);
  end process add;
end rtl;
```



Motivation till hårdvarubeskrivande språk 4(6)

- **Syntes av kombinatorisk logik**



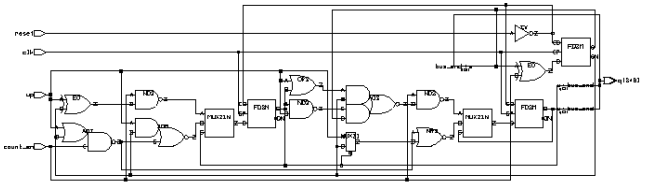
Motivation till hårdvarubeskrivande språk 5(6)

- **Beskrivning av en tillståndsmaskin i ett hårdvarubeskrivande språk (VHDL)**

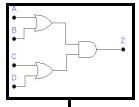
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity fsm is
  port (
    clk, reset, count_en, up : in  std_logic;
    q                        : out std_logic_vector(2 downto 0));
end fsm;

architecture rtl of fsm is
   signal count : std_logic_vector(2 downto 0);
begin  -- rtl
  process (clk, reset)
  begin  -- process
    if reset = '0' then                 -- asynchronous reset (active low)
      count <= (others=>'0');
    elsif clk'event and clk = '1' then  -- rising clock edge
      if count_en='1' then
        case up is
          when '1' => count <= count + 1;
          when others => count <= count -1;
        end case;
      end if;
    end if;
  end process;
  q <= count;
end rtl;
```


Motivation till hårdvarubeskrivande språk 6(6)

- **Logikschema för tillståndsmaskin**



Implementeringsteknologier

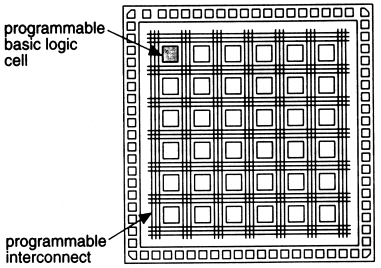


fysisk implementering

Standard-komponenter	Programmerbara kretsar	Kundanpassade eller tillämpningsspecifika kretsar
TTL familjen CMOS familjen	FPGA, PLD, etc.	ASIC

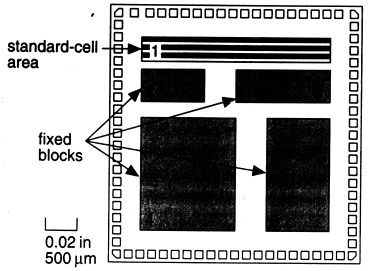
Programmerbar logik

- **Kärnan består av:**
  - **en regelbunden matris av programmerbara logiska celler som består av kombinatorisk logik samt D-vippor.**
  - **en matris med programmerbara sammankopplingar**
- **In- och utgångar**
  - **är programmerbara och omger kärnan**



Kundanpassade kretsar (ASIC)

- Konstruktören kan bestämma kretsens funktion ner på transistornivå
- Konstruktionen kan byggas upp med:
  - Standard celler där en cell motsvaras av en grind (OR, NAND, DFF etc.)
  - större färdiga moduler som minnen, mikroprocessorer.



Syfte och bakgrund till VHDL

- Problem
  - Det finns ett ökande behov av att snabbt kunna konstruera, implementera, testa och dokumentera allt mer komplexa digitala system
  - Logiska scheman och booleska ekvationer lämpar sig inte för komplexa IC (100.000 grindar)
- Lösning
  - Ett hårdvarubeskrivande språk (eng. Hardware Description Language = HDL) för att beskriva konstruktionen
  - HDL sammankopplat med datorstödd konstruktion (eng. Computer Aided Design = CAD) för automatisk syntes och simulering
  - Programmerbar logik för snabb implementering av hårdvaran
  - Kundanpassad kretsar (eng. Application Specific Integrated Circuit = ASIC) för massproduktion med låga priser

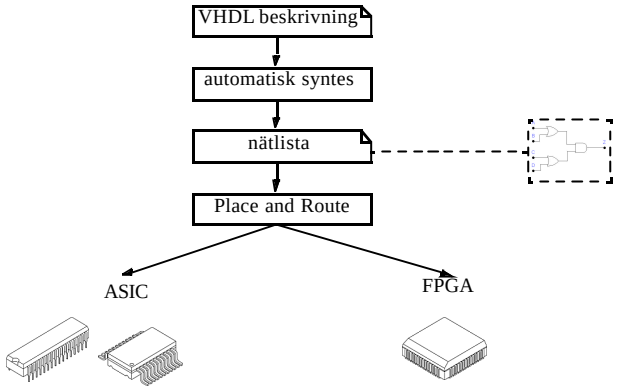
Syfte och bakgrund till VHDL

- Det finns två vanliga hårdvarubeskrivande språk (HDL) idag
  - VHDL
  - Verilog HDL
- VHDL - Very High Speed Integrated Circuit (VHSIC) Hardware Description Language
- VHDLs historia
  - Skapat av amerikanska försvarsdepartementet (Department of Defence - DoD) för att dokumentera militära konstruktioner som ska vara portabla
  - IEEE Standard 1076 (VHDL) 1987
  - Reviderad IEEE standard 1076 (VHDL) 1993
  - IEEE standard 1164 (objekttyper standardiserade) 1993
  - IEEE standard 1076.3 (syntes standardiserad) 1996

Syfte och bakgrund till VHDL

- Anledningar till att använda VHDL
  - Kraftfullt och flexibelt
  - konstruktionen görs oberoende av den komponent eller teknologi man ska använda
  - Portabelt mellan konstruktionsverktyg och komponenter
  - Övergång från prototyp till ASIC möjlig
  - Gör det möjligt att snabbt ta sig från ide till produkt med programmerbara kretsar
- Problem med VHDL
  - Tappar kontrollen över implementering på grindnivå
  - Ineffektiv implementering av logik med automatisk syntes
  - Variationer i kvalitet på syntesresultatet beroende på verktyg

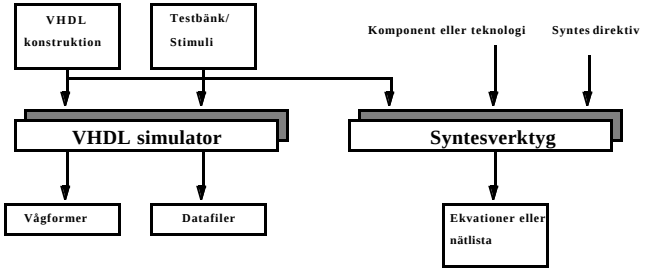
Konstruktionsflöde



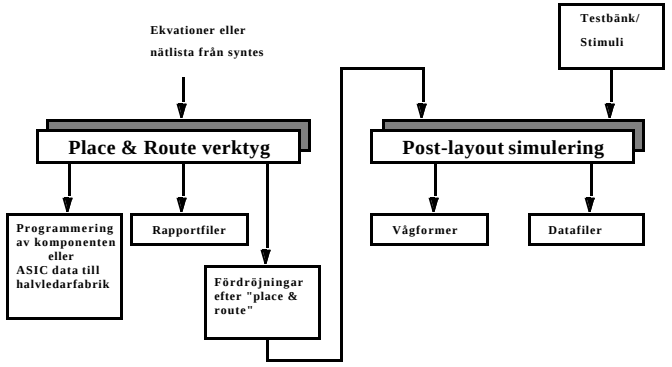
Konstruktionsflöde

- Definiera vilka krav det finns på konstruktionen (specifikation)
- Beskriva konstruktionen i VHDL
  - Top-down, hierarkisk konstruktionsmetod
  - Optimering av koden för syntes och/eller simulering
- Simulera VHDL källkoden
  - för att tidigt kunna detektera problem innan syntes
- Syntetisera, optimera och "place & route" av konstruktionen för en specifik komponent
  - syntes av VHDL koden till ekvationer och/eller nätlista (en strukturell beskrivning på grindnivå)
  - Optimering av ekvationerna så att de uppnår de fastställda kraven (t.ex krav på hastighet)
  - Gör "place & route" för att realisera kretsen i en given komponent eller teknologi
- Simulera konstruktionen för att verifiera funktionaliteten och timingen efter syntes och "place & route"
- Programmera komponenten eller sänd konstruktionsdata till halvledarfabrik som tillverkar kretsen

Konstruktionsverktyg I



Konstruktionsverktyg II



- **Konstruktionsverktyg på ITE**
  - **Syntesverktyg för Programmerbara kretsar (FPGA) och ASIC**
  - **VHDL-Simulatorer för FPGA och ASIC**
  - **Place & Route verktyg för FPGA och ASIC**
- **Komponenter**
  - **Prototypkort med FPGA (Xilinx 4005E)**

VHDL exempel och stil

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENITIY mymux1 IS
  PORT (a, b: IN bit_vector(3 downto 0); -- inputs
        sel: IN bit;
        y: OUT bit_vector(3 downto 0)); -- outputs
END mymux1;


ARCHITECTURE behavior OF mymux1 IS
BEGIN
mux: PROCESS(a, b, sel)
BEGIN
  IF sel = '0' THEN y <= a;
  ELSE y <= b;
  END IF;
  END PROCESS mux;
END PROCESS;
END behavior;
```


VHDL exempel och stil

- **Abstraktionsnivåer**
- **Beteende (behavioural)**
  - **Hög nivå, algoritmer, sekventiell exekvering**
  - **Svårt att syntetisera (långt från hårdvaran)**
  - **Enkelt att skriva och förstå**
- **Dataflöde**
  - **Mellannivå, register-till-register beskrivning, parallell exekvering**
  - **Lätt att syntetisera**
  - **Svårare att skriva och förstå (hårdvarunära beskrivning)**
- **Struktur**
  - **Lågnivå, nätlista, komponentinstantiering och sammankoppling**
  - **Trivialt att syntetisera (ingen syntes krävs)**
  - **Mycket svårt att förstå och skriva (lägsta möjliga nivå)**

Programmerbar logik

- **PAL (Programmable Array Logic) eller PLD**
  - **enkel programmerbar *AND/OR matris***
  - **vippor kan finnas för in- och utgångar**
- **CPLD (Complex Programmable Logic Device)**
  - **matris med ett antal PAL-liknande block**
  - **programmerbara sammankopplingar mellan dessa block**
- **FPGA (Field Programmable Gate Array)**
  - **matris bestående av enkla logiska celler**
  - **sammankoppling av dessa sker med ledningar i ledningskanaler**

Programmerbar logik

- Digital integrerad krets
  - dess funktion bestäms av konstruktören som gör tillämpningen
  - implementeringen görs "på plats" av konstruktören
- Många typer av programmerbara kretsar
  - Kallas ibland generellt för PLDs (Programmable Logic Devices)
  - PAL eller PLD
  - CPLD (Complex PLD)
  - FPGA

Programmerbar logik

- Konfigurering av logiska grindar
- Tekniker för att rekonfiguera
  - Antifuse -- programmering sker genom att "bränna" kopplingar
  - EPROM -- programmering sker genom att konfiguera EPROM-celle
  - SRAM -- programmering sker genom att konfigurera SRAM-celler (SRAM = statisk RAM)

Tekniker för konfigurerbar logik

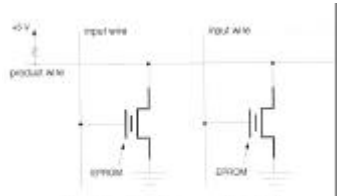
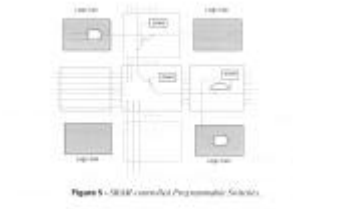


Figure 4 - *EPROM Programmable Switches.*



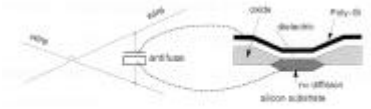


Figure 6 - *Actel Antifuse Structure.*

Name	Re-programmable	Volatile	Technology
Fuse	no	no	Bipolar
EPROM	yes out of circuit	no	UVCMOS
EEPROM	yes in circuit	no	EECMOS
SRAM	yes in circuit	yes	CMOS
Antifuse	no	no	CMOS+

Table 1 - *Summary of Programming Technologies*

PLD arkitekturer (PLA)

- Programmable Logic Device
  - en PLA komponent
- Logiken i en PLD implementeras m.h.a PLA
  - PLA -- AND-OR struktur
  - Summa-av-produktioner formas av OR-funktioner av produkttermer
  - Varje produktterm formas av en AND-funktion.

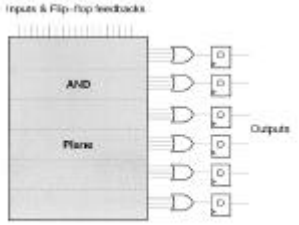


Figure 1 - *Structure of a PAL*

PLD arkitektur (CPLD)

- Complex PLD
  - Flera PLD block integreras på ett enda chip
  - PLD blocken sammankopplas via en omkopplingsmatris

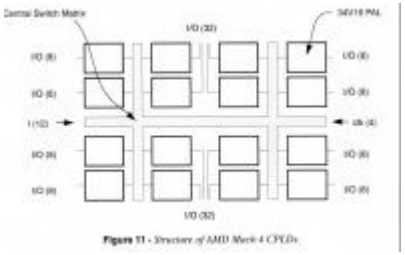


Figure 11 - *Structure of AMD Mach 4 CPLDs*

PLD karakteristik

- Förutbestämbar fördröjning
- Komplexa funktioner med många variabler kan realiseras
- Exempel på tillämpningar:
  - PLD och CPLD komponenter används ofta som interface-kretsar för systembussar
    - Innebär låg risk eftersom gränssnittet kan ändras många gånger
  - Medger effektiv implementering av kontrollkretsar (tillståndsmaskiner)

FPGA arkitektur

- Field Programmable Array
- Matris bestående av logiska block
  - kopplas samman med programmerbara omkopplare
  - ett block är en konfigurerbar logisk funktion
  - En FPGA kan ha upp till 16000 LB (4 miljoner grindar)

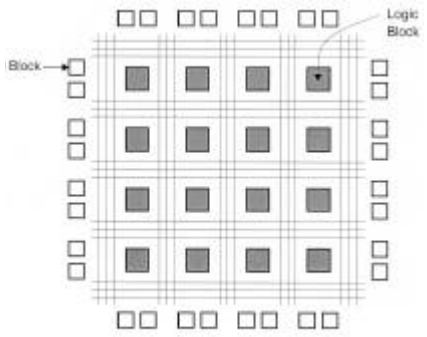


Figure 2 - *Structure of an FPGA*

FPGA arkitektur -- exempel på ett logiskt block

- Ett logiskt block (Xilinx) består av:
  - Tre stycken LUT (Look-Up Table)
  - Två register

En LUT kan programmeras till en godtycklig logisk funktion med 4 variabler

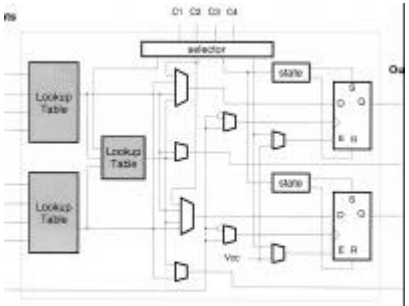
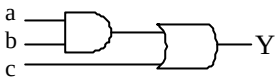


Figure 18 - *Xilinx XC4000 Configurable Logic Block (CLB).*

Exempel: Logiskt uttryck implementerat med LUT

$Y = a \wedge b \vee c$

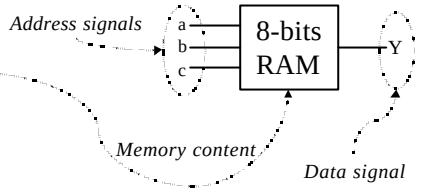
Gate implementation



Truth table

a	b	c	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Look up table Implementation



FPGA karakteristik

En FPGA karakteriseras av

- Stort antal flip-flops, 2 i varje LB
- Kompakt logik
- Snabba sammankopplingar för lokala signaler
  - Hard wired
- Långsamma sammankopplingar för globala signaler
  - Signalen måste gå genom flera omkopplare

Konsekvenser:

- Pipelining är gratis
- Speciella aritmetiska block måste användas för att de lokala sammankopplingarnas snabbhet ska kunna användas

Fördelar med programmerbar logik

- **medger flexibel konstruktion**
- **medger hög grad av automatisk konstruktion**
- **hög packningstäthet, få kapslar på kretskortet**
- **billigare**
- **låg effektförbrukning**
- **hög prestanda**

Programmeringsteknologier

- **Programmerbar sammankoppling sker genom en pass-transistor som kontrolleras från en minnesbit**
- **EPROM - programmeras elektriskt och raderas med UV-ljus**
- **EEPROM - programmeras elektriskt och raderas elektriskt**
- **Flash minne - programmeras elektriskt och raderas elektriskt**
- **SRAM - statiskt RAM, flyktigt minne**
- **Anitfuse - permanent programmering elektriskt**

PREP benchmarks

- **PREP (Programmable Electronics Performace Company) : icke-kommersiell konsortium med PLD tillverkare**
- **Användbar standard för att jämföra programmerbara logiska komponenter från olika tillverkare**
- **Nio stycken vanliga logiska funktioner som implementeras (räknare, tillståndsmaskiner, etc.)**
- **PREP rapporterar baserat på tillverkarnas tester**
  - **kapacitet : antal instanser av varje funktion som får plats i varje komponent**
  - **prestanda : maximum klockfrekvens**