Forelesning 01: Kombinasjonskreter på Port-nivå

HIEU NGUYEN

Definisjon av Kombinasjonskrets

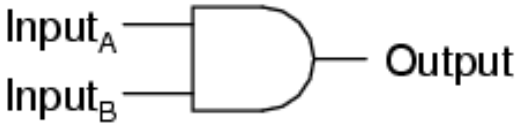
- Kombinasjonskrets er den som ikke ha minne-element
- Utgangssignaler er avhengige av kun inngangssignaler
- Eksempel: En-bits Likhetsdetektor med nedfølgende sannhetstabell er kombinasjonskrets

Inngang 1	Inngang 2	Utgang
i0	I1	eq
0	0	1
0	1	0
1	0	0
1	1	1

Port-Nivå/Gate-Level

- Gate-level/Port-Nivå betyr at vi bruker de grunnleggende logiske komponenter å oppbygge kretser
- Grunnleggende logiske komponenter
  - OG-Port/AND-Gate

2-input AND gate

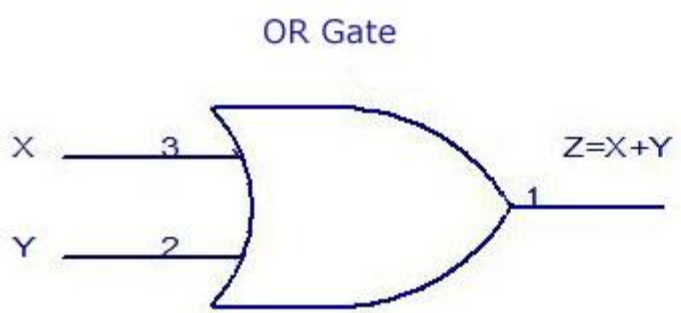


A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Port-Nivå/Gate-Level

- Grunnleggende logiske komponenter
  - ELLER-Port/ OR-Gate

2 Input OR Gate

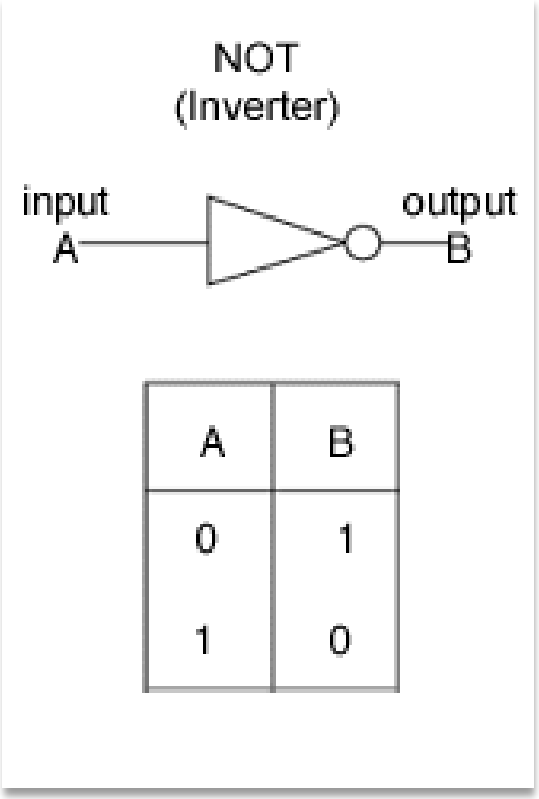


TRUTH TABLE

INPUTS		OUTPUT
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Port-Nivå/Gate-Level

- Grunnleggende logiske komponenter
  - Invertering-Port/ NOT-Gate

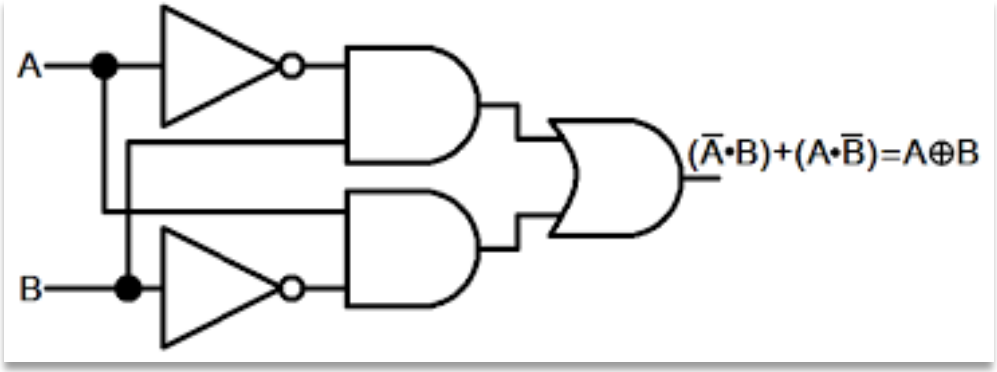


A	B
0	1
1	0

Port-Nivå/ Gate-Level

- Eksempel: Bruk bare tre grunnleggende komponenter: OG-Port, ELLER-Port, og Invertering-Port å designe en kombinasjonskretser med sannhetstabell

Inngang	Utgang
A B	Z
0 0	0
0 1	1
1 0	1
1 1	0

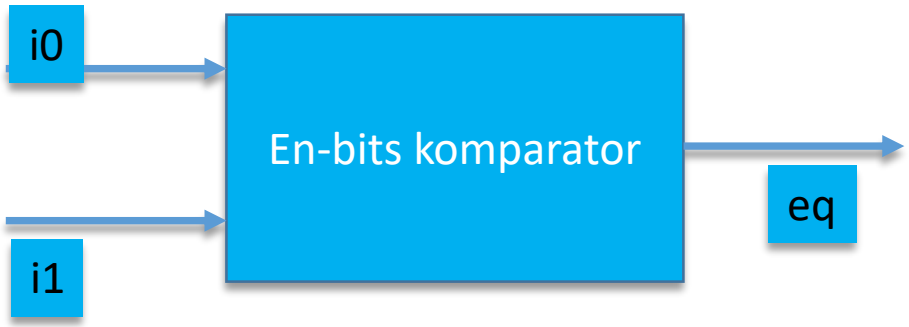


VHDL-Innledning

- VHDL: Very High Speed Integrated Circuit Hardware Description Language
- Opprinnelig sponsor av U.S Departmenent og Defense, senere overført til Institute of Electrical and Electronics Engineering (IEEE)
- VHDL blir formelt definert av IEEE standard 1076 og ratifisert i 1987 (VHDL 87)
- Boken følger revisjonen-1993 (VHDL 93)
- VHDL er brukt å beskrive og modellere digitale systemer på forskjellige nivåer, og det er et ekstremt komplekst språk!!!!
- Vi konsentrerer oss på viktige VHDL-syntetiserbare konstruksjoner, ikke alle aspekter av VHDL i dette emnet.

En-bits Komparator med VHDL/ Port-Nivå

- En-bit komparator med to inngangssignaler og ett utgangssignal

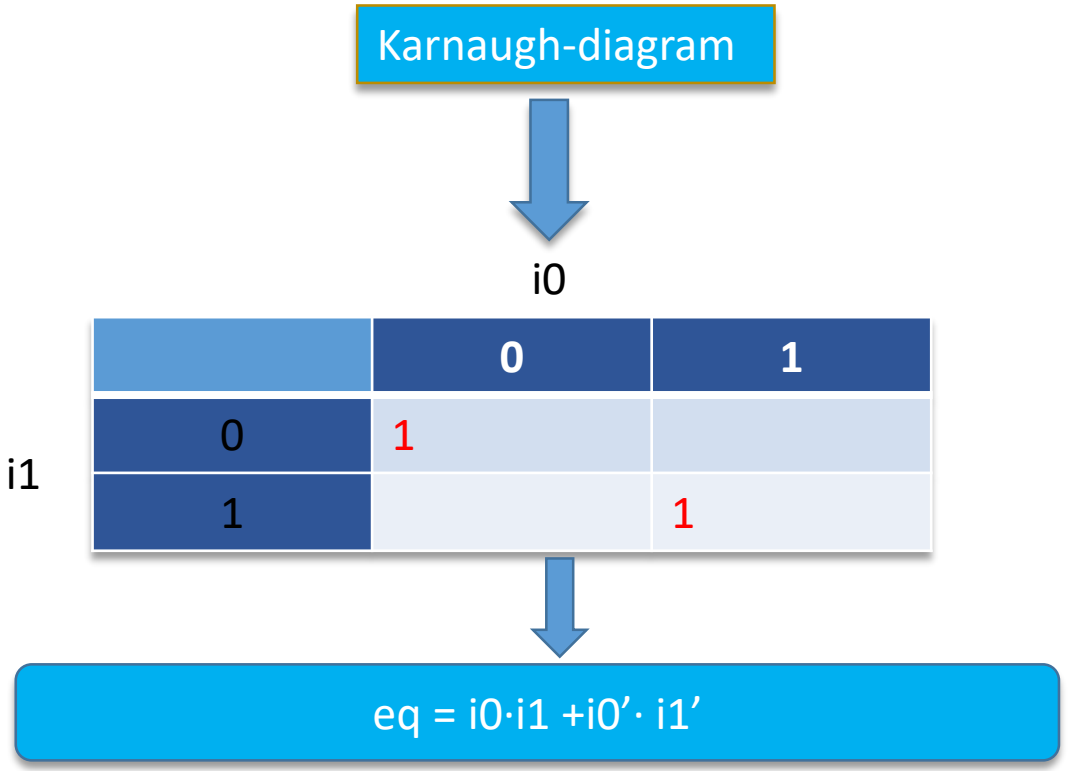


Inngang	Utgang
i0 i1	eq
0 0	1
0 1	0
1 0	0
1 1	1

En-bits Komparator med VHDL/ Port-Nivå

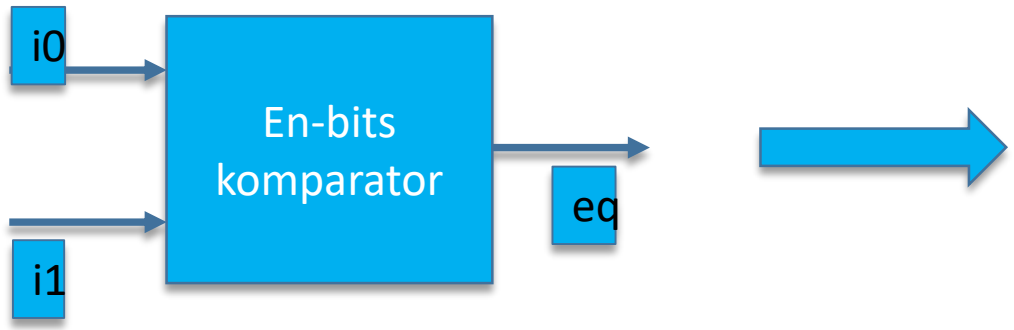
- Boolsk funksjon

Inngang	Utgang
i0 i1	eq
0 0	1
0 1	0
1 0	0
1 1	1



En-bits Komparator med VHDL/ Port-Nivå

- VHDL-implementering



eq = i0·i1 +i0'· i1'

```vhdl
-- Listing 1.1
library ieee;
use ieee.std_logic_1164.all;
entity eq1 is
   port(
      i0, i1: in std_logic;
      eq: out std_logic
   );
end eq1;

architecture sop_arch of eq1 is

   signal p0, p1: std_logic;

begin
   -- sum of two product terms
   eq <= p0 or p1;
   -- product terms
   p0 <= (not i0) and (not i1);
   p1 <= i0 and i1;
end sop_arch;
```

Leksikale Regler

- **Regel 1:** VHDL er tegn-ufølsom. Signal/variabler a eller A er samme
- **Regel 2:** Fri formattering (rom eller blanklinjer kan legges fritt)
- **Regel 3:** Datanavnet/Identifier til et objekt kan lages av 26 bokstaver, sifre, understrekning for eksempel
  - i0, i1, data_bus_1_enable, clock_pulse
- **Regel 4:** Datanavnet må begynne med bokstaven, IKKE med tall
- **Regel 5:** Kommentar må begynne med «--». Eksempel: *-- this is my comment*
- **Regel 6:** *Ikke bruke Norsk bokstaver i alle TILFELLER*

Struktur av et VHDL-Program

```
1  -- Listing 1.1
2  library ieee;
3  use ieee.std_logic_1164.all;
4
5  entity eq1 is
6     port(
7        i0, i1: in std_logic;
8        eq: out std_logic
9     );
10 end eq1;
11
12 architecture sop_arch of eq1 is
13
14    signal p0, p1: std_logic;
15
16 begin
17    -- sum of two product terms
18    eq <= p0 or p1;
19    -- product terms
20    p0 <= (not i0) and (not i1);
21    p1 <= i0 and i1;
22 end sop_arch;
```

Library

Entity

Architecture

Struktur av et VHDL-Program Library

- Linje 2 og 3

```
1  -- Listing 1.1
2  library ieee;
3  use ieee.std_logic_1164.all;
4
```

Library

  - **Library**: er nøkkelord
  - ieee: er navnet til biblioteket som vi vil bruke
  - Use: betyr at vi vil bruke en spesifikk pakke i biblioteket
  - All: betyr at vi vil bruke alle funksjoner, operatorer og signaltyper i pakken
- Vi må deklarere «library» i alle VHDL-program!!!!

Struktur av et VHDL-Program
Entity

- Entity-deklarasjon presenterer inn- og utgangssignaler av en krets
- Linjer fra 5 til 10 er for «entity»

```
 5  entity eq1 is
 6     port(
 7        i0, i1: in std_logic;
 8        eq: out std_logic
 9     );
10  end eq1;
```

  - Linje 5:indikerer navnet til kretsen. Her heter kretsen «eq1»
  - Linjer 6,7,8,9 er port-deklarasjon:
    - i0, i1,eq er port-navn
    - in/out er modus
    - std_logic: datatype

Struktur av et VHDL-Program
Entity

- Syntaks for Entity

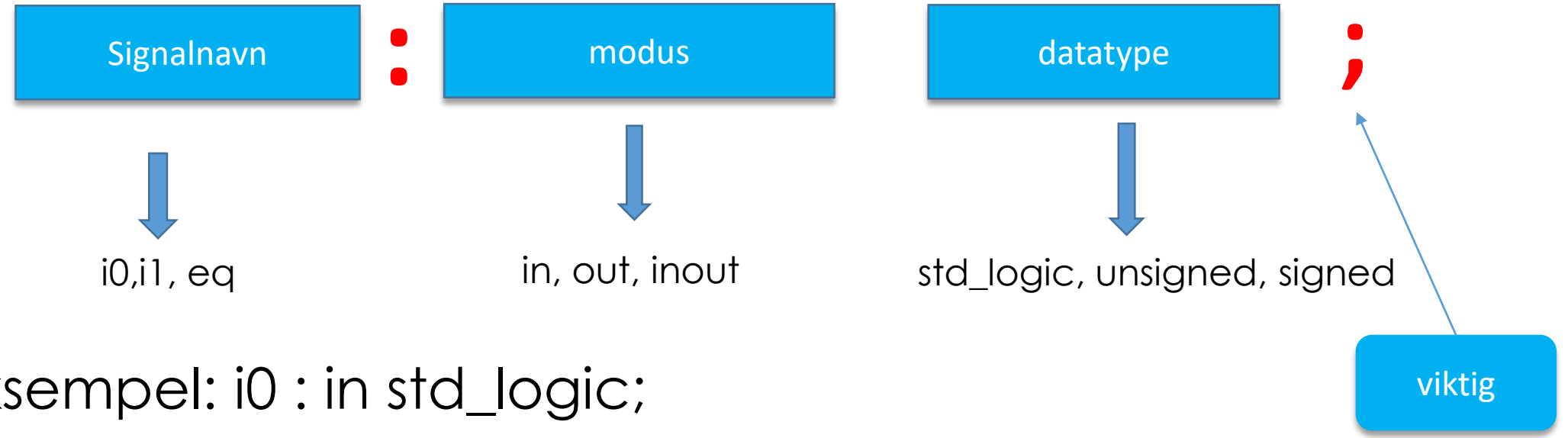
```
entity entity_name is
port (port_name_1 : mode    datatype ;
      port_name_2 : mode    datatype;
      .....
      port_name_n : mode    datatype);
end name_entity;
```

```
 5  entity eq1 is
 6     port(
 7        i0, i1: in std_logic;
 8        eq: out std_logic
 9     );
10  end eq1;
```

Struktur av et VHDL-Program
Entity

- Syntaks for Port-Deklarasjon

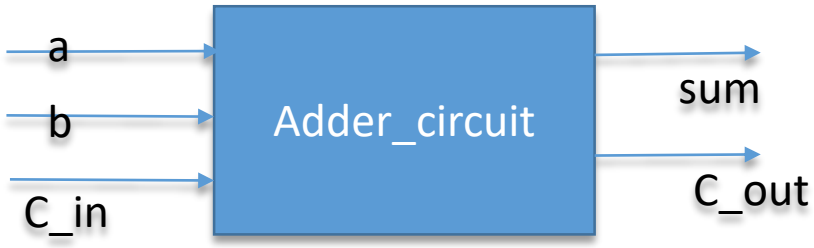
```
 5  entity eq1 is
 6     port(
 7        i0, i1: in std_logic;
 8        eq: out std_logic
 9     );
10  end eq1;
```



- Eksempel: i0 : in std_logic;

Struktur av et VHDL-Program
Entity

- Eksempel: Skriv entity-deklarasjon for kretsen nedenfor



- Navnet til kretsen: Adder_circuit
- Inngangssignaler: a,b,C_in
- Utgangssignaler: sum, c_out

```
25  entity adder_circuit is
26      Port ( a       : in  STD_LOGIC;
27             b       : in  STD_LOGIC;
28             C_in    : in  STD_LOGIC;
29             sum     : out  STD_LOGIC;
30             C_out   : out  STD_LOGIC);
31  end adder_circuit;
```

Struktur av et VHDL-program Entity/Datatype

- VHDL er et strong-typed språk
  - Hvert objekt må ha en datatype
  - Kun definerte verdier og operasjoner kan brukes til objektet
- Std_logic type og sine varianter
  - Std_logic er definert i std_logic_1164 pakke
  - Std_logic består 9 verdier
  - Tre hovedverdier
    - '0': logic 0
    - '1': logic 1
    - 'Z' høy impedans/åpen krets
  - 'X' og 'U' verdier: forholdvis «unknown/ukjent» og «uninitialized/uinitialisert»
  - '-','H','L','W' er ikke brukt i denne boken

Struktur av et VHDL-Program Entity/Datatype

- Std_logic_vector
  - Et signal i digital krets består ofte mange biter (8,16,32…). Std_logic_vector er definert for denne hensikten
- Std_logic_vector er definert som en datatabell/array med hvert element av std_logic type.
- Eksempel
  - A: in std_logic_vector(7 downto 0); -- minkende rekkefølge
  - Vi kan bruke A(7 downto 4) for å spesifisere en rekke av biter
  - Vi kan bruke A(1) for å få tilgang til et element av datatabellen
  - A : in std_logic_vector(0 to 7); -- økende rekkefølge
- Minkende rekkefølge format er mest brukt siden vi vil at Mest signifikant bit (MSB) ligger lengst til venstre posisjon.

Oppsummering

- Struktur av en VHDL-fil: 3-deler
  - Bibliotek (IEEE, std_logic_1164 pakke)
  - Entity
    - Syntaks for entity deklarasjon
    - Port deklarasjon
    - Port modus: in, out, inout
    - Datatype: std_logic, std_logic_vector(N-1 downto 0)
  - Architecture

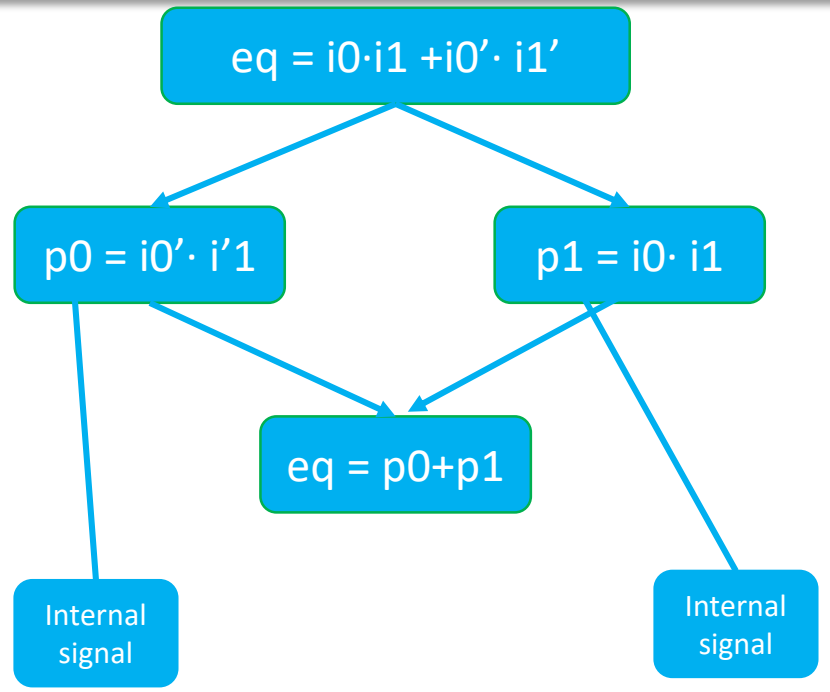
Struktur av et VHDL-Program Architecture

- Syntaks for archiecture

```
architecture name_architecture of entity_name is
      --internal signals here
      begin

      -- setninger placed here
end name_architecture;
```

Husk

We code together now!!!!



Struktur av et VHDL-Program Architecture

- Architecture er plassen hvor vi beskriver operasjonen av kretsen

```
architecture name_architecture of entity_name is
      --internal signals here
      begin

      -- setninger placed here
end name_architecture;
```

```
12  architecture sop_arch of eq1 is
13
14     signal p0, p1: std_logic;
15
16  begin
17     -- sum of two product terms
18     eq <= p0 or p1;
19     -- product terms
20     p0 <= (not i0) and (not i1);
21     p1 <= i0 and i1;
22  end sop_arch;
```

- VHDL tillater muiltiple-architecture assosiert med entity. Derfor blir architecture identifisert med sitt eget navn: sop_arch

Struktur av et VHDL-Program Architecture

```
12 architecture sop_arch of eq1 is
13
14    signal p0, p1: std_logic;
15
16 begin
17    -- sum of two product terms
18    eq <= p0 or p1;
19    -- product terms
20    p0 <= (not i0) and (not i1);
21    p1 <= i0 and i1;
22 end sop_arch;
```

Begynnelsen av architecture deklarasjon

Valgfri deklarasjonsdel: constants, internal signals,…

Hoved beskrivelse ligger mellom begin og end

Slutten av architecture deklarasjon

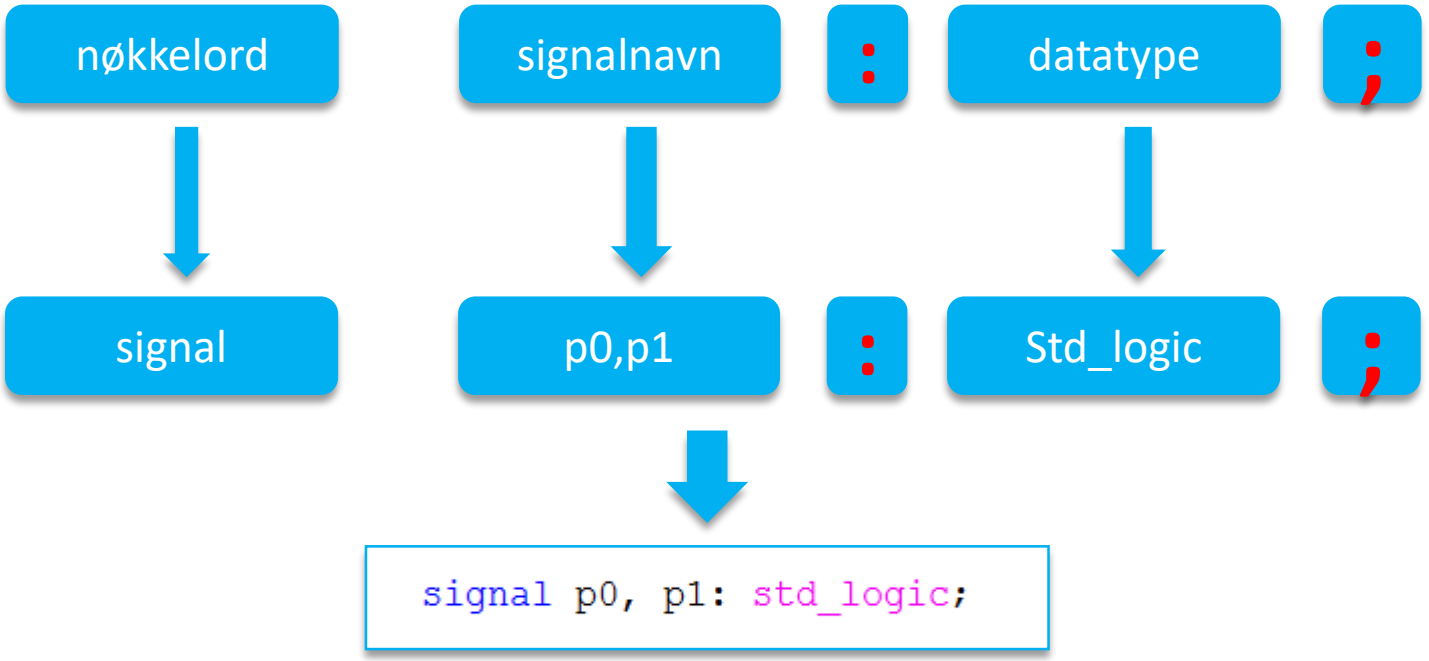
Struktur av et VHDL-Program Architecture

```
12 architecture sop_arch of eq1 is
13
14     signal p0, p1: std_logic;
15
16 begin
17     -- sum of two product terms
18     eq <= p0 or p1;
19     -- product terms
20     p0 <= (not i0) and (not i1);
21     p1 <= i0 and i1;
22 end sop_arch;
```

Tre «concurrent statements/samtidige setninger»

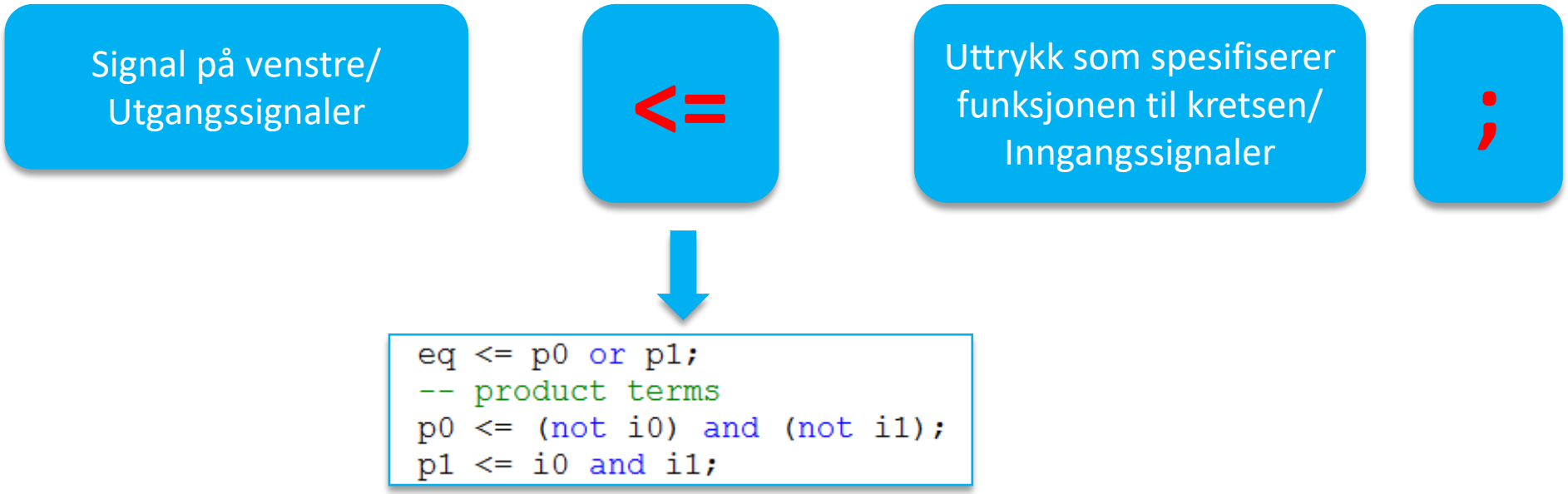
Struktur av et VHDL-Program Architecture

- Interne signaler er deklarert slik:



Struktur av et VHDL-Program
Samtidig Setning

- Samtidig Setning /Concurrent Statment struktur



Struktur av et VHDL-Program
Samtidig Setning

- Samtidige setninger opererer parallelt. Rekkefølge av setningene spiller ingen rolle!!!! [Man kan bytte rekkefølgen av de tre setningene]

```
eq <= p0 or p1;
-- product terms
p0 <= (not i0) and (not i1);
p1 <= i0 and i1;
```

- Når verdier av p0 og p1 endres, blir den første setningen aktivert og uttrykket på høyre side blir evaluert. Signalet eq derfor blir tilordnet en ny verdi etter en standardforplantingsforsinkelse

Struktur av et VHDL-Program

```
1  -- Listing 1.1
2  library ieee;
3  use ieee.std_logic_1164.all;
4
```

```
entity entity_name is
port (port_name_1 : mode : datatype ;
      port_name_2 : mode : datatype;
      .....
      port_name_n : mode : datatype);
end name_entity;
```

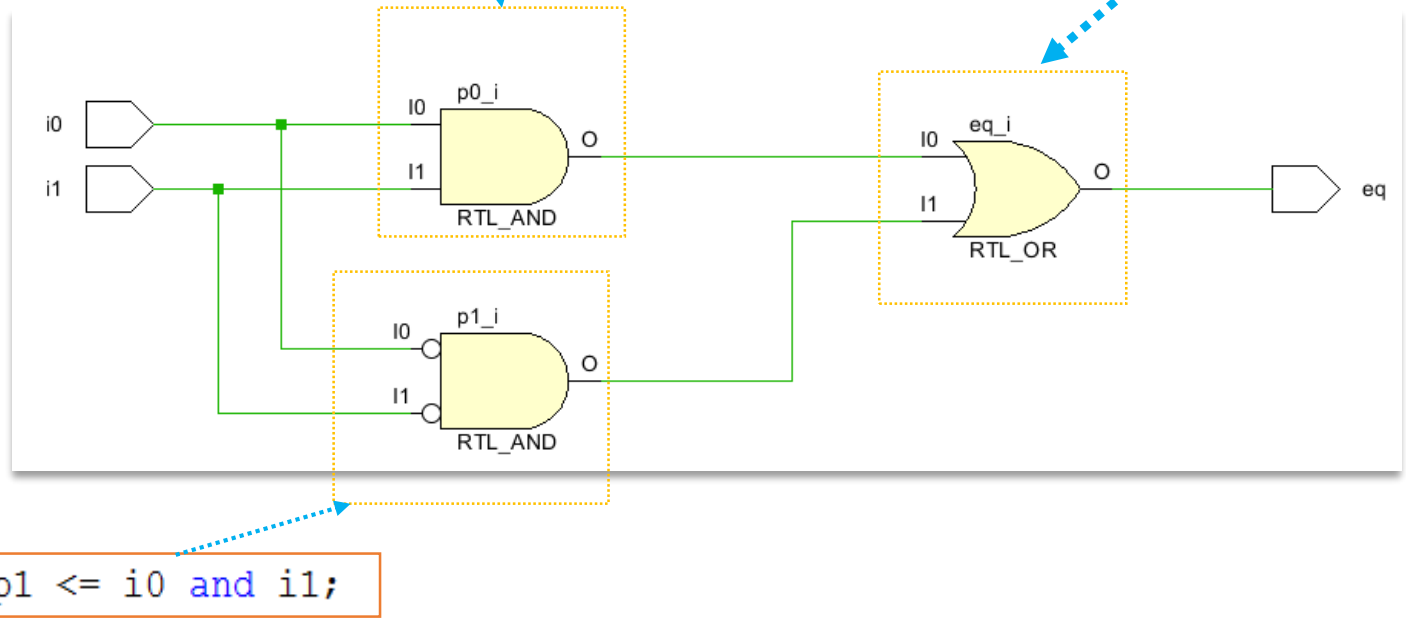
```
architecture name_architecture of entity_name is
      --internal signals here
      begin

      -- setninger placed here
end name_architecture;
```

```
1  -- Listing 1.1
2  library ieee;
3  use ieee.std_logic_1164.all;
4
5  entity eq1 is
6     port(
7        i0, i1: in std_logic;
8        eq: out std_logic
9     );
10 end eq1;
11
12 architecture sop_arch of eq1 is
13
14    signal p0, p1: std_logic;
15
16 begin
17    -- sum of two product terms
18    eq <= p0 or p1;
19    -- product terms
20    p0 <= (not i0) and (not i1);
21    p1 <= i0 and i1;
22 end sop_arch;
```

Struktur av et VHDL-Program
Grafisk Presentasjon

```
p0 <= (not i0) and (not i1);
```

```
eq <= p0 or p1;
```

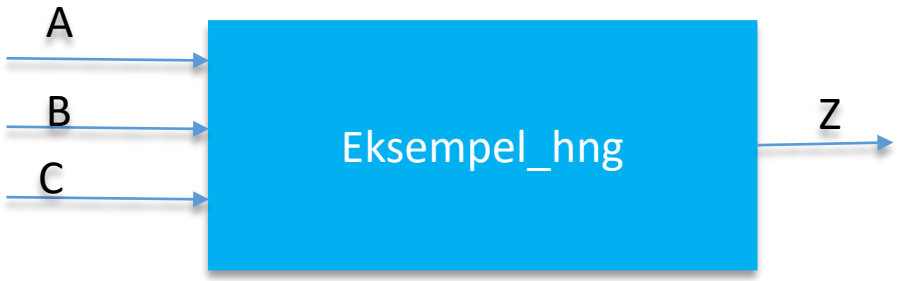



```
p1 <= i0 and i1;
```

```
12  architecture sop_arch of eq1 is
13
14      signal p0, p1: std_logic;
15
16  begin
17      -- sum of two product terms
18      eq <= p0 or p1;
19      -- product terms
20      p0 <= (not i0) and (not i1);
21      p1 <= i0 and i1;
22  end sop_arch;
```

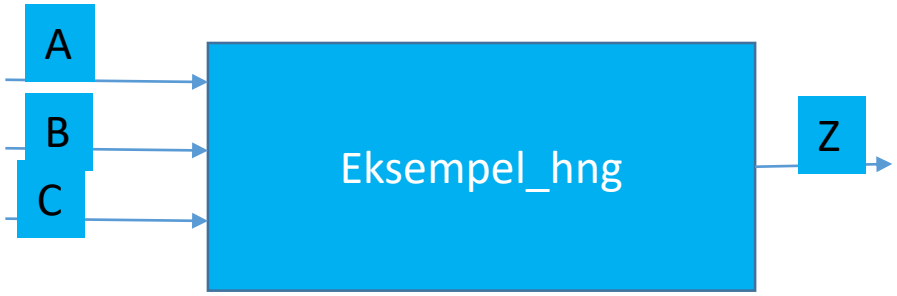
Struktur av et VHDL-program Eksempel

- Navnet til entity: eksempel_hng
- Navnet til architecture: sop_arch



Struktur av et VHDL-program Eksempel

- Navnet til entity: eksempel_hng
- Navnet til architecture: sop_arch

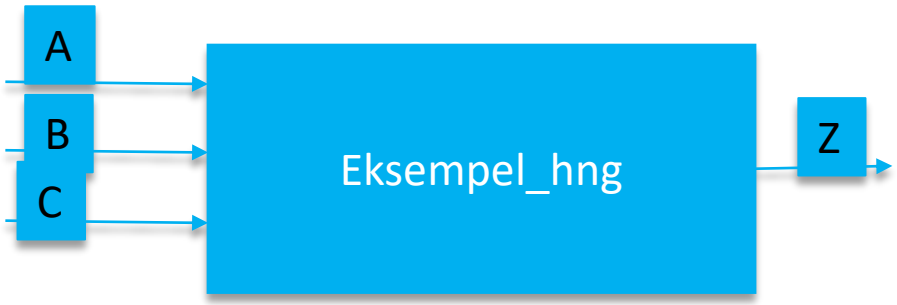


Biblioteksdel

```
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
```

Struktur av et VHDL-program Eksempel

- Navnet til entity: eksempel_hng
- Navnet til architecture: sop_arch

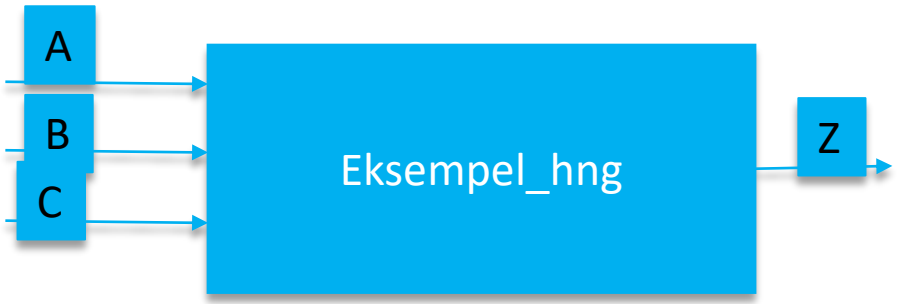


Entity-del

```
34  entity eksempel_hng is
35      Port ( A : in  STD_LOGIC;
36             B : in  STD_LOGIC;
37             C : in  STD_LOGIC;
38             Z : out  STD_LOGIC);
39  end eksempel_hng;
```

Struktur av et VHDL-program Eksempel

- Navnet til entity: eksempel_hng
- Navnet til architecture: sop_arch

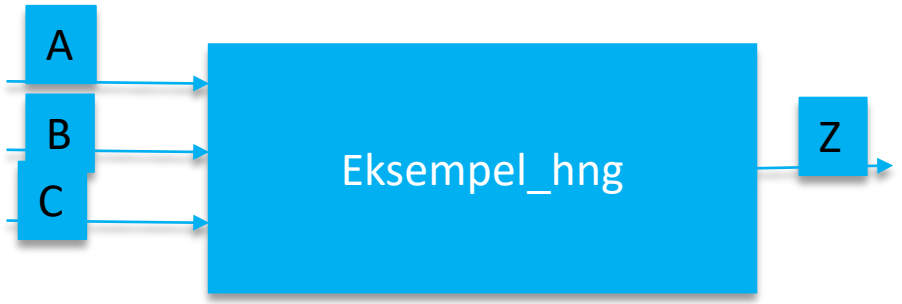


architecture-del

```
41
42  architecture sop_arch of eksempel_hng is
43
44  signal p0, p1, p2 : std_logic;
45
46  begin
47
48     p0     <= A and (not B);
49
50     p1     <= B and (not C);
51
52     p2     <= C and (not A);
53
54     Z      <= p0 or p1 or p2;
55
56  end sop_arch;
```

Struktur av et VHDL-program Eksempel

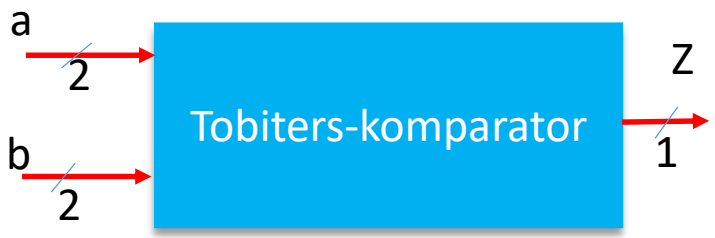
$Z = A \cdot B' + B \cdot C' + C \cdot A'$




```
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  entity eksempel_hng is
26      Port ( A : in  STD_LOGIC;
27             B : in  STD_LOGIC;
28             C : in  STD_LOGIC;
29             Z : out  STD_LOGIC);
30  end eksempel_hng;
31
32  architecture sop_arch of eksempel_hng is
33  signal p0, p1, p2 : std_logic;
34  begin
35
36     p0    <= A and (not B);
37     p1    <= B and (not C);
38     p2    <= C and (not A);
39     Z     <= p0 or p1 or p2;
40  end sop_arch;
41
```

To-biters Komparator

- Inngangssignaler a og b, hvert har to biter og datatype av std_logic_vector
- Utgangssignalet Z (en bit), datatype av std_logic



- Forholdet mellom inngangssignalene og utgangssignalet er:

Hvis a er lik b, har Z logisk verdi '1'. Ellers, har Z logisk verdi '0'.

To-biters Komparator

Sannhetstabellen til to-biters komparator

a(1)	a(0)	b(1)	b(0)	Z
0	0	0	0	1
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	1
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	1
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	1

To-biters Komparator

- Trinn 2: Finn boolsk funksjon ved bruk av Karnaugh-diagramet

a(1 downto 0)

b(1 downto 0)	00	01	11	10
00	1			
01		1		
11			1	
10				1

- Boolsk funksjon

Z = a'(1)a'(0)b'(1)b(0) +a'(1)a(0)b'(1)b(0) +a(1)a(0)b(1)b(0)+a(1)a'(0)b(1)b'(0)

To-biters Komparator: VHDL-Kode

- Trinn 1: Bibliotek-deklarasjon

```
19  ----------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
```

- Trinn 2: Entity-deklarasjon

Std_logic_vector

```
24  entity tobiters_komparator is
25      Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
26             b : in  STD_LOGIC_VECTOR (1 downto 0);
27             z : out  STD_LOGIC);
28  end tobiters_komparator;
29
```

To-biters Komparator: VHDL-Kode

- Trinn 3: Architecture-del med interne signaler

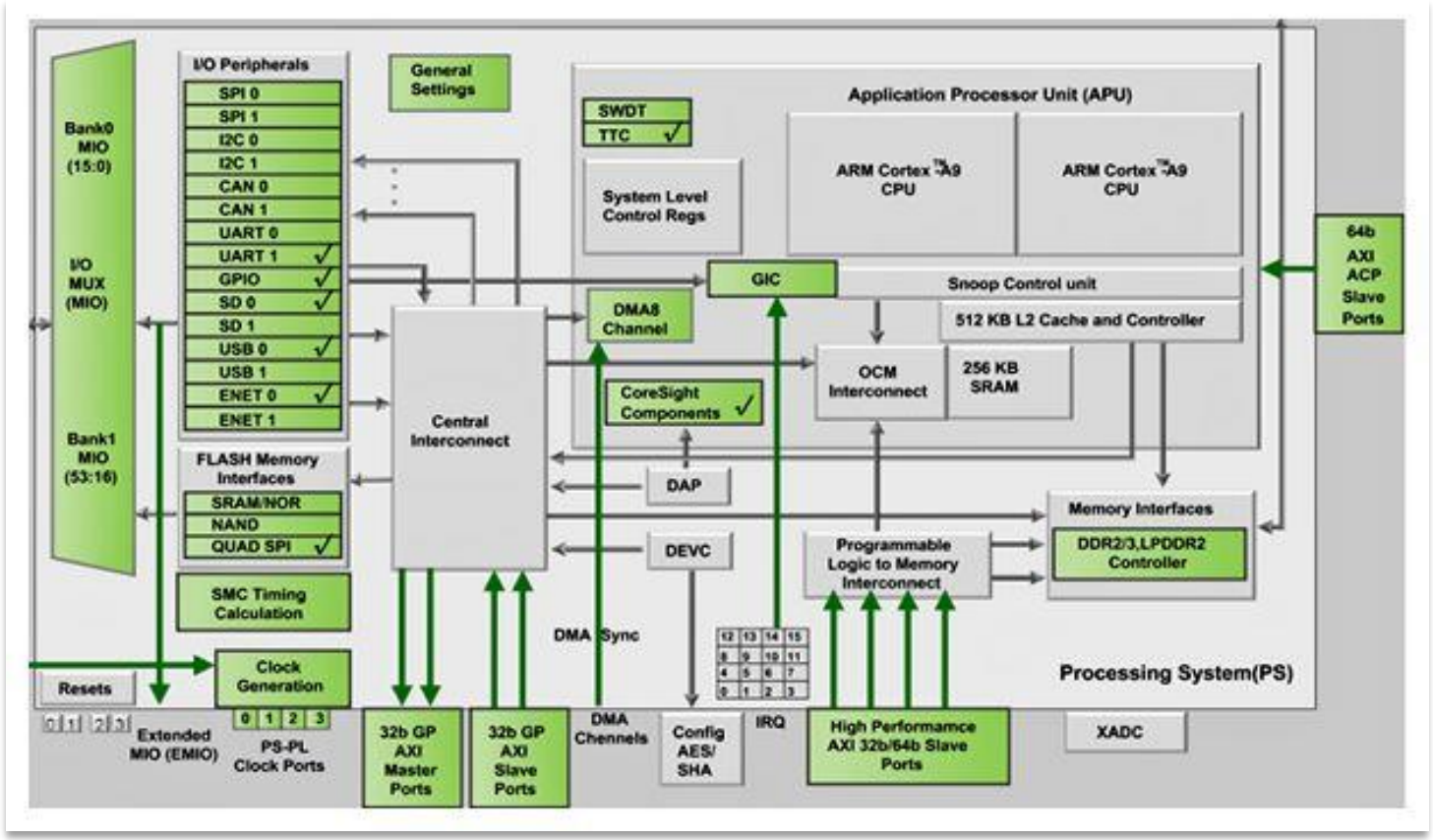
```
30
31  architecture Behavioral of tobiters_komparator is
32
33  ----------internal signals--------------------
34  signal p0, p1,p2, p3 : std_logic;
35  ------------------------------------------------
36
37  begin
38
39  p0 <= (not a(1)) and (not a(0)) and (not b(1)) and (not b(0));
40
41  p1 <= (not a(1)) and (a(0)) and (not b(1)) and (b(0));
42
43  p2 <= ( a(1)) and (not a(0)) and (b(1)) and (not b(0));
44
45  p3 <= a(1) and  a(0) and b(1) and b(0);
46
47  Z <= p0 or p1 or p2 or p3;
48
49  end Behavioral;
50
```

Interne signaler

Samtidige setninger

Z = a'(1)a'(0)b'(1)b(0) +a'(1)a(0)b'(1)b(0) +a(1)a(0)b(1)b(0)+a(1)a'(0)b(1)b'(0)

Strukurell Design Metode



Strukturell Metode
To-Biters Komparator

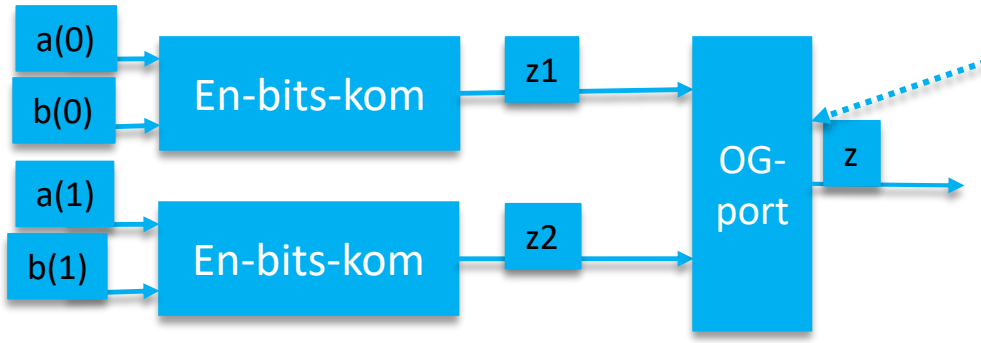
Strukturell metode er en designmetode der et stort system blir bygd basert på mindre sub-systemer/ bygge-blokker.

- Fordeler av den strukturelle metode
  - Enkelhet
  - Lett å håndtere og verifisere funksjonaliteter av kretsen
  - Hurtigutviklingsprosess ved bruk av for-designede/pre-designed komponenter

Strukturell Metode
To-Biters Komparator

- Vi designer two_bit_comparator basert på one_bit_comparator.
- Observasjon:

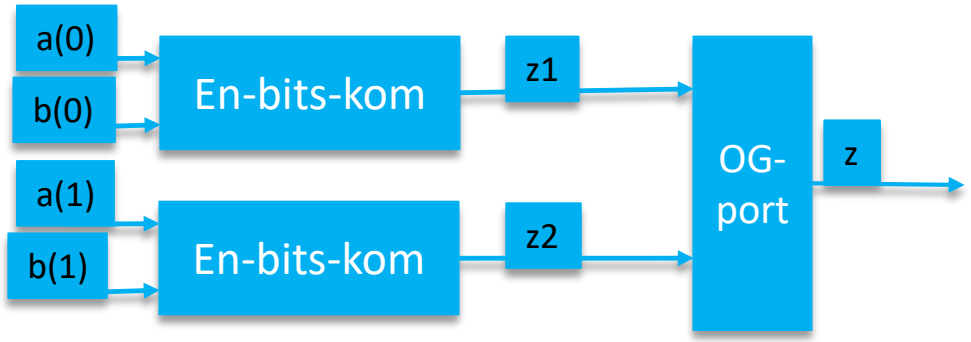
Hvis a(1 downto 0) = b(1 downto 0) hvis bare hvis
a(1) = b(1) og a(0) = b(0)



z1	z2	z
0	0	0
0	1	0
1	0	0
1	1	1

Strukturell Metode
To-Biters Komparator

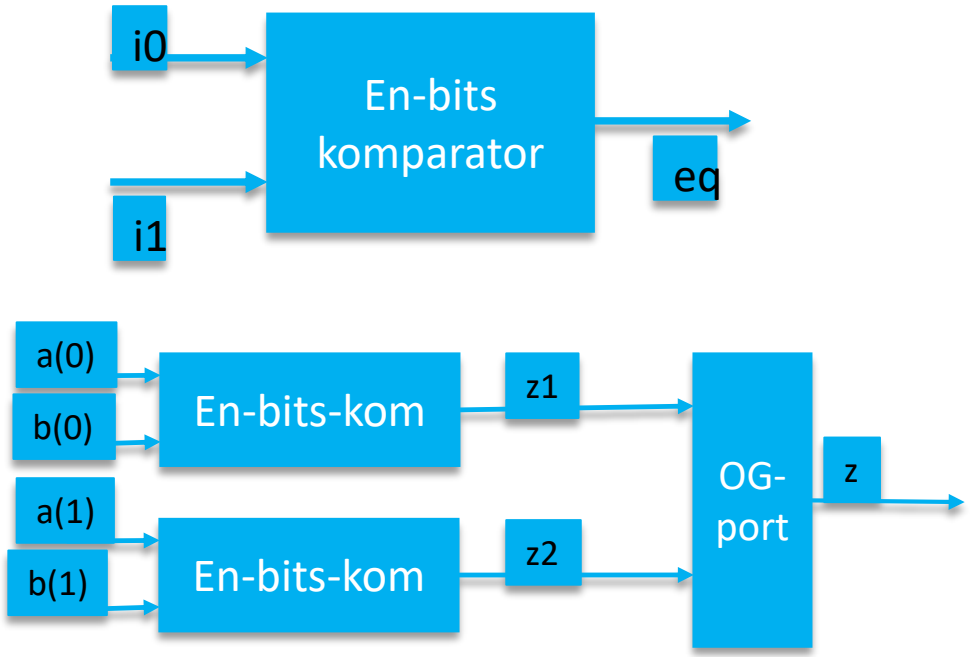
- VHDL gir oss en mekanisme som kalles «component instantiation» til å konstruere digitalt system ved bruk av den strukturelle metoden
- Kildekoden for two_bit_comparator med «component instantiation»




```
19  ----------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  entity tobiter_komparator_strukt is
24      Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
25             b : in  STD_LOGIC_VECTOR (1 downto 0);
26             Z : out  STD_LOGIC);
27  end tobiter_komparator_strukt;
28
29  architecture Behavioral of tobiter_komparator_strukt is
30
31  ----------internal signals----------------------
32  signal z1, z2 : std_logic;
33
34  begin
35
36  ------------- Instantiate two one-bit comparator---------
37  komparator_bit_0 : entity work.eq1(sop_arch)
38              port map (i0 => a(0), i1 => b(0), eq => z1);
39
40  komparator_bit_1 : entity work.eq1(sop_arch)
41              port map (i0 => a(1), i1 => b(1), eq => z2);
42
43  ------------------------------------------------------------
44
45  z  <= z1 and z2;
46
47  end Behavioral;
```

Strukturell Metode
To-Biters Komparator

- Tilbakekalling av en-bits-komparator




```
 1  -- Listing 1.1
 2  library ieee;
 3  use ieee.std_logic_1164.all;
 4
 5  entity eq1 is
 6     port(
 7        i0, i1: in std_logic;
 8        eq: out std_logic
 9     );
10  end eq1;
11
12  architecture sop_arch of eq1 is
13
14     signal p0, p1: std_logic;
15
16  begin
17     -- sum of two product terms
18     eq <= p0 or p1;
19     -- product terms
20     p0 <= (not i0) and (not i1);
21     p1 <= i0 and i1;
22  end sop_arch;
```

Strukturell Metode
To-Biters Komparator

En-bits-komparator

```
1  -- Listing 1.1
2  library ieee;
3  use ieee.std_logic_1164.all;
4
5  entity eq1 is
6     port(
7        i0, i1: in std_logic;
8        eq: out std_logic
9     );
10 end eq1;
11
12 architecture sop_arch of eq1 is
13
14    signal p0, p1: std_logic;
15
16 begin
17    -- sum of two product terms
18    eq <= p0 or p1;
19    -- product terms
20    p0 <= (not i0) and (not i1);
21    p1 <= i0 and i1;
22 end sop_arch;
```

To-biters-komparator

```
19 ----------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 entity tobiter_komparator_strukt is
24     Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
25            b : in  STD_LOGIC_VECTOR (1 downto 0);
26            Z : out  STD_LOGIC);
27 end tobiter_komparator_strukt;
28
29 architecture Behavioral of tobiter_komparator_strukt is
30
31 ----------internal signals---------------------
32 signal z1, z2 : std_logic;
33
34 begin
35
36 ------------ Instantiate two one-bit comparator---------
37 komparator_bit_0 : entity work.eq1(sop_arch)
38             port map (i0 => a(0), i1 => b(0), eq => z1);
39
40 komparator_bit_1 : entity work.eq1(sop_arch)
41             port map (i0 => a(1), i1 => b(1), eq => z2);
42
43 ----------------------------------------------------------
44
45 z  <= z1 and z2;
46
47 end Behavioral;
```

Strukturell Metode
To-Biters Komparator

- Syntaks for component instantiation: Alternativ 1

Port-klartlegging

```
unit_label: entity lib_name.entity_name(architecture_name)
        port map (formal_signal_1 => actual_signal_1,
                  formal_signal_2 => actual_signal_2,
                  .........
                  formal_signal_n => actual_signal_n);
```

```
36  ------------- Instantiate two one-bit comparator---------
37  komparator_bit_0 : entity work.eq1(sop_arch)
38              port map (i0 => a(0), i1 => b(0), eq => z1);
```

- Unit_label : komparator_bit_0
- Nøkkelord: entity
- Lib_name: work
- entity:_name: eq1
- Architecture_name: sop_arch
- Formal_signal: i0,i1, eq
- Actual signal: a(0), b(0), z1

Strukturell Metode
To-Biters Komparator

- Syntaks for component instantiation: Alternativ 2 at vi deklarerer komponentene før vi bruker dem.
- Eksempel

```
19  ----------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  entity tobiter_komparator is
24      Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
25             b : in  STD_LOGIC_VECTOR (1 downto 0);
26             z : out  STD_LOGIC);
27  end tobiter_komparator;
28
29  architecture Behavioral of tobiter_komparator is
30
31  ----------------component declaration-------------
32  component eq1 is
33     port(
34        i0, i1: in std_logic;
35        eq: out std_logic
36     );
37  end component eq1;
38
39  ---------------internal signals-----------------
40  signal z1, z2 : std_logic;
41
42  begin
43
44  ----------------Instantiate two one-bit comparator----
45  komparator_bit_0 : eq1
46  port map (i0 => a(0), i1 => b(0), eq => z1);
47
48  komparator_bit_1 : eq1
49  port map (i0 => a(1), i1 => b(1), eq => z2);
50
51  ------------------------------------------------------
52
53  Z   <= z1 and z2;
54
55  end Behavioral;
```

Deklarerer komponenten mellom architecture og begin

Instantiate uten å spesifisere biblioteket

Strukturell Metode
To-Biters Komparator

- Syntaks for komponent deklarasjon [mellom architecture og begin]

```
component entity_name
   port (port_name_1 : mode    datatype ;
         port_name_2 : mode    datatype;
         .....
         port_name_n : mode    datatype);
end component name_entity;
```

```
31  ----------------component declaration------------
32  component eq1
33      port(
34          i0, i1: in std_logic;
35          eq: out std_logic
36      );
37  end component eq1;
```

```
 5  entity eq1 is
 6      port(
 7          i0, i1: in std_logic;
 8          eq: out std_logic
 9      );
10  end eq1;
```

Forskjell av et nøkkelord

For lathets skyld: kopier og lim

Strukturell Metode
To-Biters Komparator

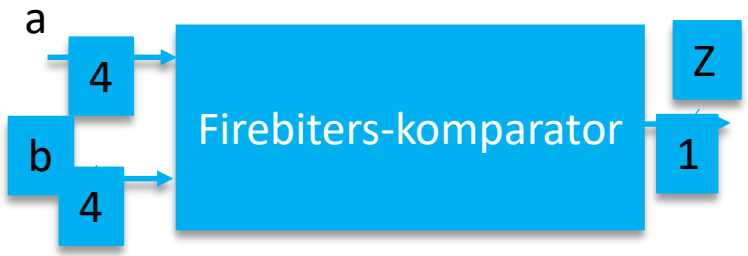
- Syntaks for komponent instantiation med komponent-deklarasjon

```
unit_label : entity_name(architecture_name)
      port map (  formal_signal_1  => actual_signal_1,
                  formal_signal_2   => actual_signal_2,
                  .........
                  formal_signal_n => actual_signal_n);
```

```
44  ----------------Instantiate two one-bit comparator----
45  komparator_bit_0 : eq1
46  port map (i0 => a(0), i1 => b(0), eq => z1);
47
48  komparator_bit_1 : eq1
49  port map (i0 => a(1), i1 => b(1), eq => z2);
50
51  ------------------------------------------------------
```

Strukturell Metode
Four_bit_comparator

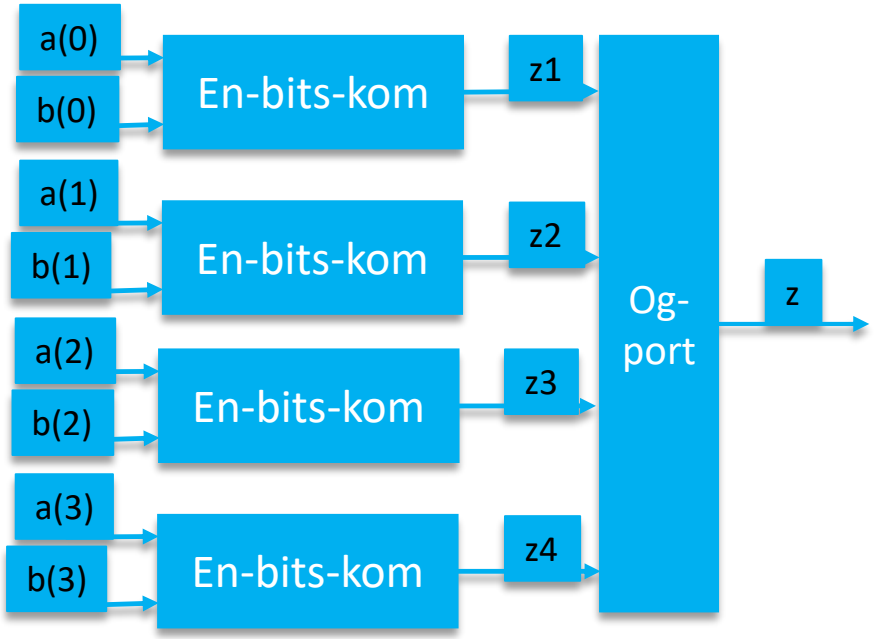
- Vi må bruke den strukturelle metode for å designe fire-biters komparator



- Det er to alternativer
  - Alternativ 1: Bygge opp fire-biters komparator basert på en-bits komparator
  - Alternativ 2: Bygge opp fire-biters komparator basert på to-biters komparator

Strukturell Metode
Four_bit_comparator

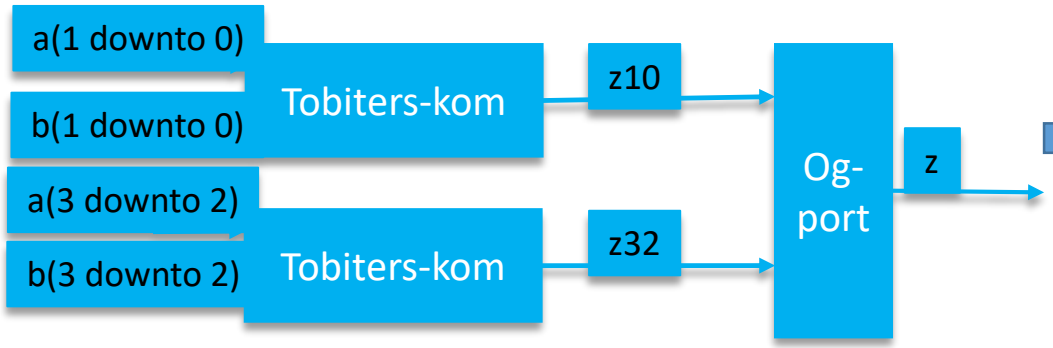
- Skjematisk diagram for alternativ 1




```
19  ----------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  entity firebiters_komparator_alternativ1 is
24      Port ( a : in  STD_LOGIC_VECTOR (3 downto 0);
25             b : in  STD_LOGIC_VECTOR (3 downto 0);
26             z : out  STD_LOGIC);
27  end firebiters_komparator_alternativ1;
28
29  architecture Behavioral of firebiters_komparator_alternativ1 is
30  ------------------internal signals----------------------
31  signal z1, z2, z3, z4 : std_logic;
32
33  begin
34
35  ------------------instantiate komponent----------------
36  komparator_bit_0 : entity work.eq1(sop_arch)
37  port map (i0 => a(0), i1 => b(0), eq => z1);
38
39  komparator_bit_1 : entity work.eq1(sop_arch)
40  port map (i0 => a(1), i1 => b(1), eq => z2);
41
42  komparator_bit_2 : entity work.eq1(sop_arch)
43  port map (i0 => a(2), i1 => b(2), eq => z3);
44
45  komparator_bit_3 : entity work.eq1(sop_arch)
46  port map (i0 => a(3), i1 => b(3), eq => z4);
47
48  ----------------------------------------------------------
49
50  z <= z1 and z2 and z3 and z4;
51
52  end Behavioral;
53
```

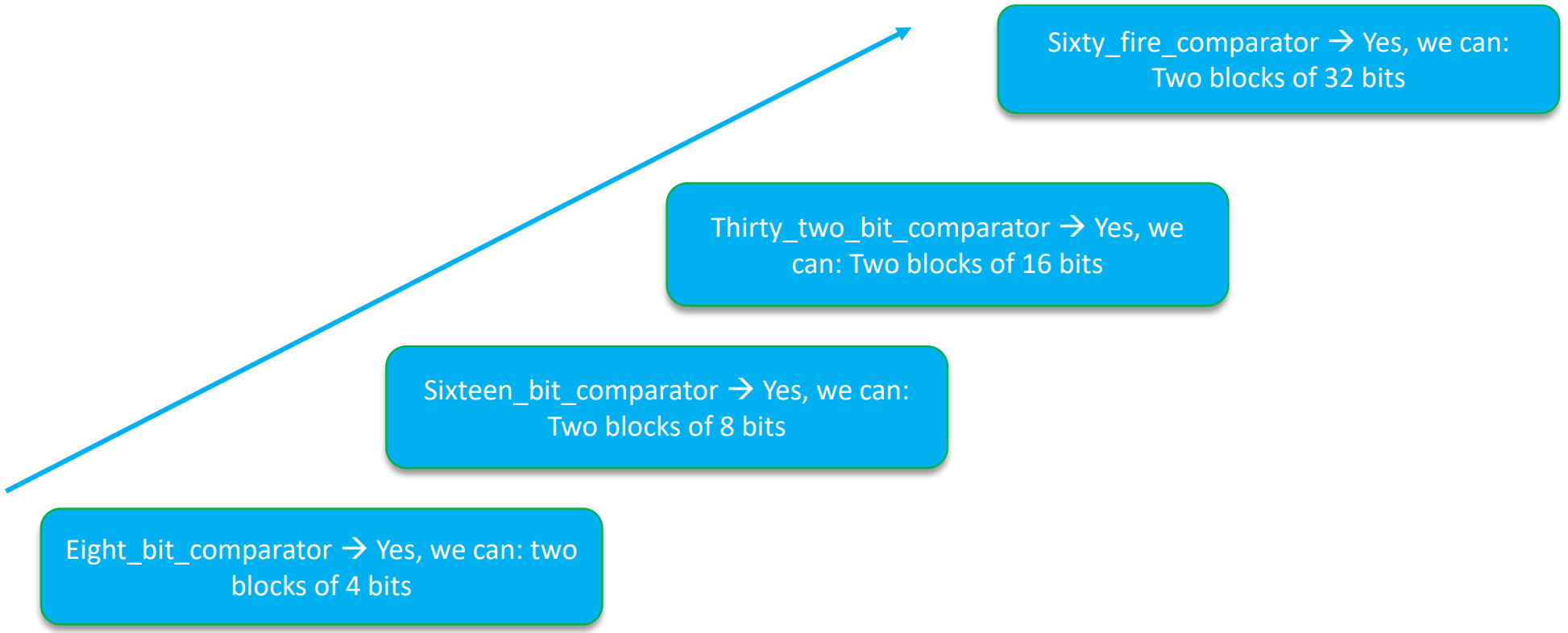
Strukturell Metode
Four_bit_comparator

- Skjematisk diagram for alternativ 2




```
19  ----------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  entity firebiters_komparator_alternativ2 is
24      Port ( a : in  STD_LOGIC_VECTOR (3 downto 0);
25             b : in  STD_LOGIC_VECTOR (3 downto 0);
26             z : out  STD_LOGIC);
27  end firebiters_komparator_alternativ2;
28
29  architecture Behavioral of firebiters_komparator_alternativ2 is
30  -------------internal signals----------------------
31  signal z10, z32 : std_logic;
32
33  begin
34
35  --------------Instantiate tobiters komparator-------------------------
36  komparator_bit_1_0: entity work.tobiter_komparator_strukt(Behavioral)
37  port map (a => a(1 downto 0), b => b(1 downto 0), z => z10);
38
39  komparator_bit_3_2: entity work.tobiter_komparator_strukt(Behavioral)
40  port map (a => a(3 downto 2), b => b(3 downto 2), z => z32);
41
42  ----------------------------------------------------------------------------
43
44  z   <= z10 and z32;
45
46  end Behavioral;
47
```

Strukturell Metode: N_bit_comparator



Oppsummering

- Struktur av en VHDL-design: 3 deler (library, entity og architecture)
- Port-deklarasjon
- Data type: std_logic, std_logic_vector
- Internal signal deklarasjon (mellom architecture og begin)
- Samtidige setninger (plasseres mellom begin og end architecture, rekkefølge spiller ingen rolle)
- Rutine for flow design (Flyt-design)
  - Sett opp sannhetstabell
  - K-diagra
  - Boolsk likning
  - Utvikle VHDL-kodelinjer

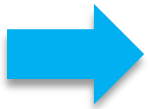
Oppsummering

- Strukturell metode
  - Dele stort system i mange mindre oppgaver
- VHDL-syntaks for strukturell design
  - Komponent-deklarasjon (plasseres mellom architecture og begin)

```
component entity_name is
   port (port_name_1 : mode : datatype ;
         port_name_2 : mode : datatype;
         .....
         port_name_n : mode : datatype);
end component name_entity;
```

  - Komponent-instantiation (plasseres mellom begin og end-architecture)

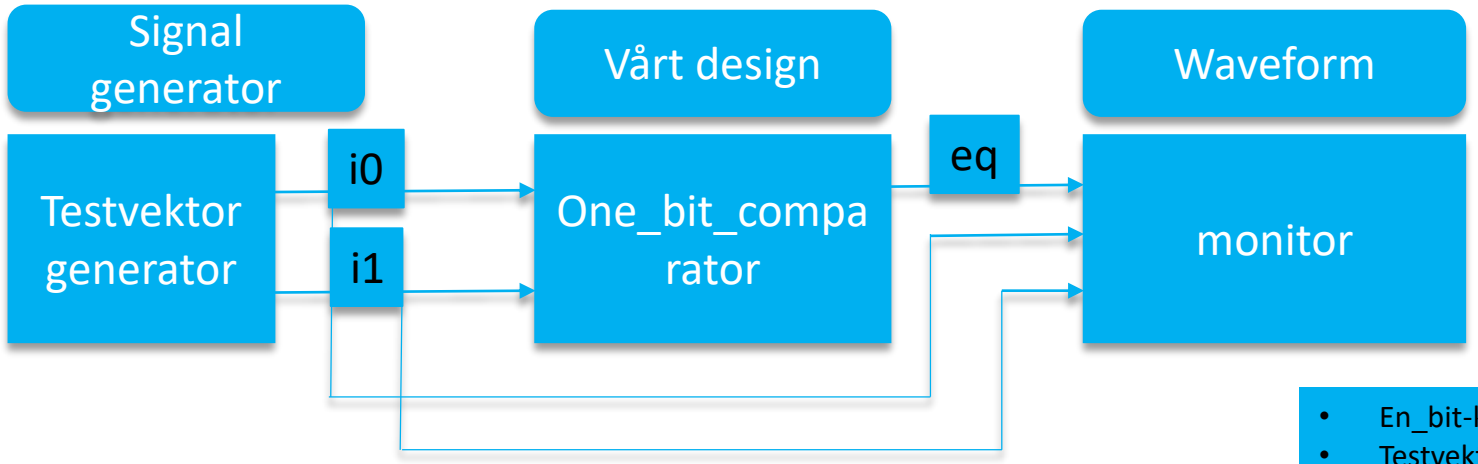
```
unit_label : entity_name(architecture_name)
      port map (  formal_signal_1  => actual_signal_1,
                  formal_signal_2   => actual_signal_2,
                  .........
                  formal_signal_n => actual_signal_n);
```



```
44  ----------------Instantiate two one-bit comparator----
45  komparator_bit_0 : eq1
46  port map (i0 => a(0), i1 => b(0), eq => z1);
47
48  komparator_bit_1 : eq1
49  port map (i0 => a(1), i1 => b(1), eq => z2);
50
51  ------------------------------------------------------
```

Testbenk/Testbench

- Etter utvikling av VHDL-kode, blir koden simulert i vertsdatamaskin for å verifisere funksjonalitet av kretsen. Vi skaper et spesielt VHDL-program, kalt testbench.
- Testbenk etterlikner/hermer en fysisk labbenk
- Eksempel av testbenk for tobiters-komparator



- En_bit-komparator er unit under test
- Testvektor generator genererer testmønster
- Monitor eksaminerer utgangsrespons

Testbenk/Testbench: VHDL-Kode

Download Testbench-Template from Canvas

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
---------------------------------------------------------------
ENTITY tb_hieu_template IS
END tb_hieu_template;
---------------------------------------------------------------
ARCHITECTURE behavior OF tb_hieu_template IS
 ---------------------------------------------------------------
    -- Component Declaration for the Unit Under Test (UUT)
 ---------------------------------------------------------------

---------------------------------------------------------------
   --Input signals here
---------------------------------------------------------------

---------------------------------------------------------------
   --Output signals here

---------------------------------------------------------------

BEGIN

---------------------------------------------------------------
   -- Instantiate the Unit Under Test (UUT)
---------------------------------------------------------------
---------------------------------------------------------------
   -- Stimulus process
   stim_proc: process
   begin
      -- generate the test vector-1 here
      wait for 100 ns;

      -- generate the test vector-2 here
      wait for 100 ns;



      wait;
   end process;
---------------------------------------------------------------
END;
```

Testbenk/Testbench: VHDL-Kode

- Bibliotek og entity deklarasjoner

```
27  ----------------------------------------
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30  
31  
32  ENTITY one_bit_comparator_tb IS
33  END one_bit_comparator_tb;
```

Ingen inngangs- og utgangssignaler

Testbenk/Testbench: VHDL-Kode

- Architecture- Del 1

```
36  ARCHITECTURE behavior OF one_bit_comparator_tb IS
37
38      -- Component Declaration for the Unit Under Test (UUT)
39
40      COMPONENT eq1
41      PORT(
42           i0 : IN  std_logic;
43           i1 : IN  std_logic;
44           eq : OUT  std_logic
45          );
46      END COMPONENT;
47
48
49     --Inputs
50     signal i0 : std_logic := '0';
51     signal i1 : std_logic := '0';
52
53     --Outputs
54     signal eq : std_logic;
55
```

Component declaration

Input Signals

Output Signals

Testbenk/Testbench: VHDL-Kode

- Architecture- Del 2

```
67
68    -- Stimulus process
69    stim_proc: process
70    begin
71       -- hold reset state for 100 ns.
72       wait for 100 ns;
73
74       -- insert stimulus here
75       i0 <= '0';
76       i1 <= '1';
77       wait for 100 ns;
78
79       i0 <= '1';
80       i1 <= '0';
81       wait for 100 ns;
82
83       i0 <= '1';
84       i1 <= '1';
85       wait for 100 ns;
86
87       i0 <= '0';
88       i1 <= '0';
89       wait for 100 ns;
90
91       wait;
92    end process;
```

Denne setningen indikerer at verdiene skal vare i 200 ns

To setninger spesifiserer verdier for to inngangssignaler

Testbenk/Testbench: VHDL-Kode

Testbench for two_bit_comparator