INF3430 H2008

KONGSBERG

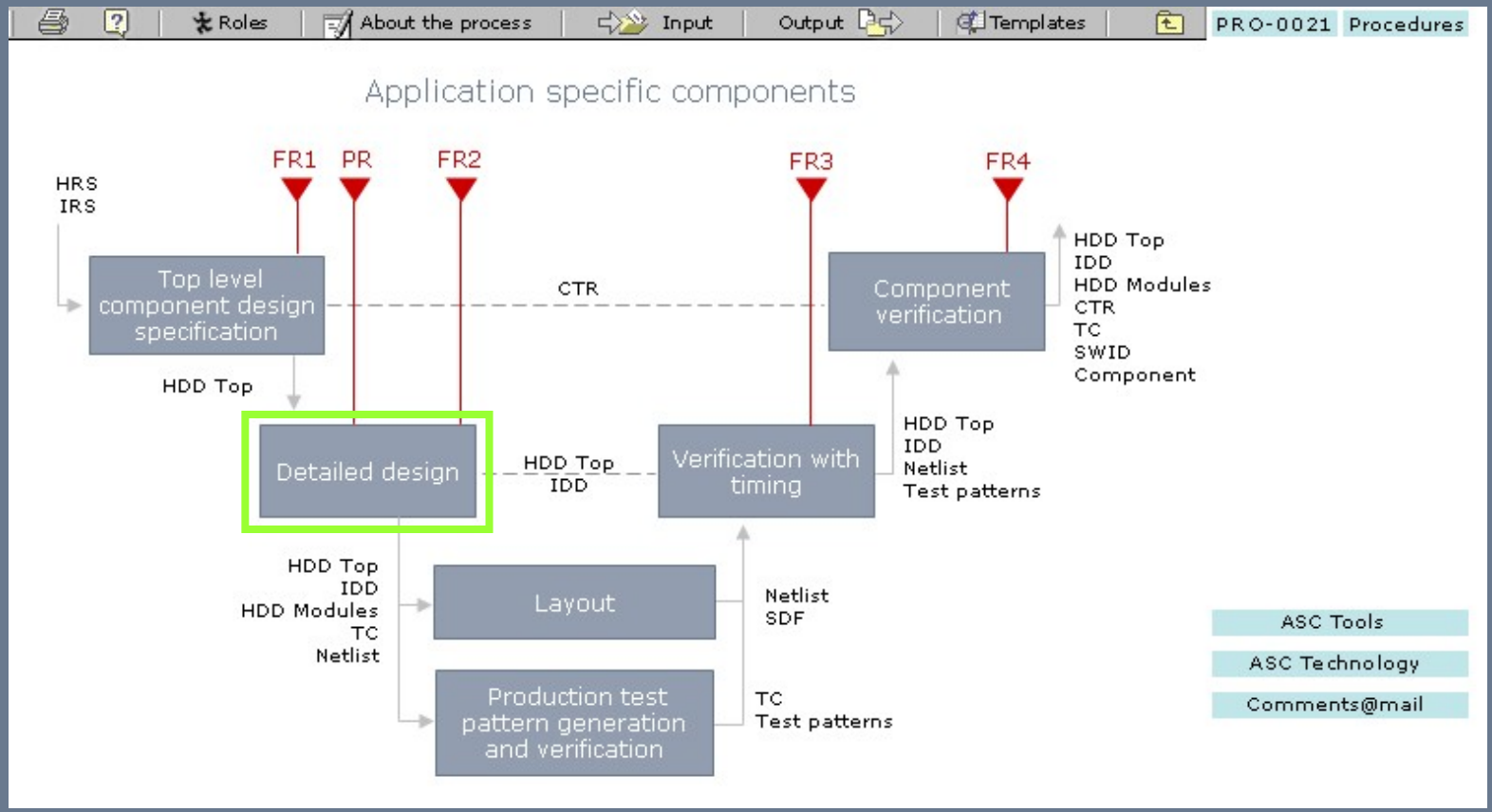
FPGA/ASIC koderegler og designmetodikk

Hvordan en FPGA designer kan være minst mulig på lab'en

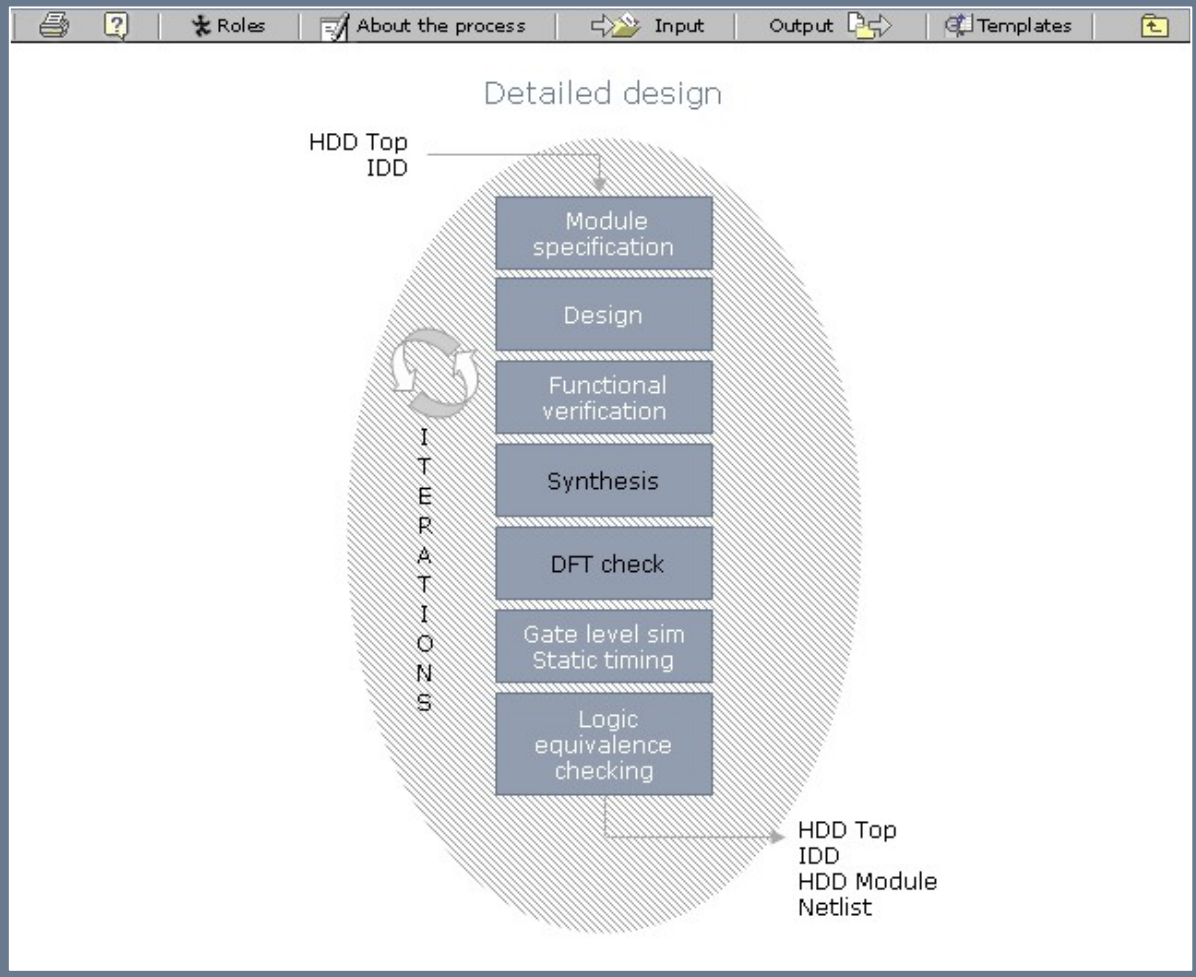
Agenda

- Utviklingsprosessen i KDA
- Design regler
- Parallellisering og pipelining
- Simulering vs. lab. debug
- Konklusjon

FPGA utviklingsprosess



Detailed Design



Design regler

KONGSBERG

- KDA egen erfaring + erfaring fra Tandberg Storage, Nordic Semiconductor og FFI.
  - VHDL koderegler
  - Navnregler
  - DFTregler
  - Metodikk med selvsjekkende testbenker og krav om script for syntese og P&R.

- VHDL Modelling Guidelines, ESA Estec 1994 og Reuse Methodology Manual, Kluwer

Kode regler I

KONGSBERG

- Only a single statement shall be allowed per line.
- Use (if possible) only active high signals. External signals that are active low shall be inverted in the first entered module. An exception is an active low reset signal and signals to RAMs and IPs.
- Avoid internal tri-state busses.
- Allowed types:
  - **std_logic**
  - **std_logic_vector** - only used for busses that are not numbers
  - **unsigned** - used for all unsigned numbers
  - **signed** - used for all signed numbers
  - **integer** – **shall be avoided if possible**
  - **enumerated types** - can be used for state machine variables. If used in several modules, they shall be defined in common project packages.
  - **boolean** – used for boolean operations (std_logic is preferred)
  - composite types – collection of above types. Records can be used for grouping signals (e.g. cpubus = data + control).

Kode regler II

KONGSBERG

- Vectors shall be defined as MSB down to LSB, e.g. std_logic_vector (7 downto 0). LSB shall always be 0 if there are no other special reasons.
- An original signal type shall be used throughout the hierarchy, if the target port is of the same type. Signals from/to the core (or higher) module shall be of the type **std_logic** and **std_logic_vector only**.
- Allowed packages:
  - ieee.std_logic_1164.all
  - ieee.numeric_std.all
  - ieee.std_logic_textio
  - std_textio
  - std_developerskit
  - project and company packages
- Only explicit port mapping is allowed.
- Port ordering in entity shall be: clocks, resets, common signals, signals grouped by functionality or module (e.g. from redif, from ntc, from fxc, to ac, to pif). Group signals by interface alphabetically, inputs first, outputs and then I/O.

Kode regler III

KONGSBERG

- Do not use **block** statement.
- Concurrent statements shall only be used for assigning the outgoing port to its internal signal (e.g. **res <= res_i**) and for creation of tri-state busses.
- Avoid **with** and **when** statements (they make simulation slower).
- Do not use too many, or too few processes. Process shall describe a function (functional sub-module).
- Finite state machines can be described either in two processes (one sequential and one combinatorial) or just as a single process.
- It is recommended to use functions **rising_edge** and **falling_edge** instead of '**event** when describing clock edges.
- A multi level **if-else** statement shall only be used when a priority encoder is intended. Otherwise **case** statement shall be used. Always use default value if latch is not intended. Default value can also be set first in the process, above **if/case** statement. In a **case** statement, use **others** (do not use **null**). Signals in the reset part of the process as well as default assignments shall be listed in alphabetical order.
- Variables can be used both for internal process calculations and for register inferring. If the variable is only used for intermediate calculation, always assign the variable before it is used.

Kode regler IV

KONGSBERG

- Use parenthesis to group expressions in IF-statements to improve readability.
- Avoid purely combinational modules as they are not recommended for synthesis. If possible, all output signals of a module shall come from registers.
- Use of multi dimensional ports is not allowed.
- Asynchronous signals or signals crossing clock domain boundaries shall be synchronized to avoid metastability. Such signals shall typically be synchronized twice (“double flopping”) before they are used, but this depends on the clock frequency. Asynchronous input signals shall be synchronized in the first entered module.
- Use tabs (automatically substituted with spaces) when writing code in Emacs. Indentation shall be 2 or 3 spaces.
- Longer concept description comments for the module shall be part of the header or placed early in the file. Shorter line comments shall be used for each process or functional part of the code. A comment declaration of each port and signal (each on a separate line) shall be used. Comments shall be placed above or to the right of the code. Align comments if possible.

Navn regler I

KONGSBERG

- Upper case shall only be used for: constants, enumerated type literals, generics and process labels. Lower case shall be used for remaining names including file names. Mixing of cases shall only be allowed for functions and procedures. All names shall be as short as possible, but always meaningful.
- Always separate design units (entity, architecture) in separate files.
- Never mix architecture types (e.g. rtl and str).
- File and design unit naming:
  - entity uart_ent.vhd
  - architecture uart_rtl.vhd
  - configuration uart_cfg.vhd
  - package def. nova_pck.vhd
  - package body nova_bdy.vhd
  - testbench tb_uart.vhd or t_uart_vhd
  - testbench config tb_uart_cfg.vhd or t_uart_cfg.vhd
- Entity and instance names:
  - Always use instance labels starting with _0 (e.g. fxc_reg_0, fxc_reg_1).
  - Use prefix for modules connected with a higher level structural module. (e.g. “reg” leaf module with name “fxc_reg” connected to fxc structural module).

Navn regler II

KONGSBERG

- Predefined architecture types:
  - str structural
  - rtl register transfer level
  - beh behavioural
  - dmy dummy (empty)
- Predefined suffixes for signals
  - _n negative polarity
  - _i internal signal of outgoing port
  - _d1, _d2 delayed signal (i.e. number of cycles).
  - _s1, _s2 synchronized signal
  - _str strobe
- Predifined names:
  - Clock and reset signals shall be preserved throughout the hierarchy.
  - Clocks: default clock signal shall be mclk. If other clocks exist, the name shall be clk and include frequency (m=MHz and k=kHz), e.g. clk_34k, clk_10m, clk_10m24
  - Resets:
    - Synchronous: rst, rst_n
    - Asynchronous: arst, arst_n
    - NB! If only one reset type is used, use rst or rst_n whatever style!
  - Interrupts: signal names shall be “irq_” e.g. irq_fifo_empty.
  - Process labels shall start with prefix P_ (e.g. P_DATA_READ:)
  - Generate labels shall start with prefix G_ (e.g. G_MUX:)

Navn regler III

KONGSBERG

- Reserved names
  - Do not use VHDL or Verilog reserved names.
- Module names
  - Use short module names.
  - Do not use underscore in module names (except when prefix used).
  - Preferably 2 to 5 characters (except when prefix used; e.g. fxc_reg)
- From-to type signal names
  - For processor interface signals, use prefix (e.g. pif_cs, pif_data).
  - Do not mix prefix and suffix style (redif_we, data_redif).
  - Read data from module to processor interface shall preferably not include module name, and shall preferably be the same for all modules, e.g. data_2pif.
  - Modules that are connected with processor interface shall use data connection signal with module name as part of signal name, e.g. data_ac2pif.
  - If the same logical signal with different timing goes to several modules, use _2 type: (e.g. fhc_2ac, fhc_2netm).
  - If signals with identical names are found in other modules, and it is necessary to differentiate between them or show the direction of the signals, use _2 (e.g. fhc_ntc2ac, fhc_redif2pif).
  - Avoid using direction as part of signal names between internal modules, e.g. data_in, res_out.
  - Avoid using _2 at top level.

Katalog struktur

KONGSBERG

Klassisk ASIC katalog struktur med ClearCase versjonskontroll på UNIX:

- Topp nivå: top, core, package, ip, scripts og flashfiles. Evnt. bin, doc og adm.
- Under top: hdl, tb, sim, syn, pr.
- Under core: hdl, module1, module2, module3, osv. Evnt. tb, sim, syn og pr.
- For alle moduleX: hdl, submodule1, submodule2, submodule3, osv.

- Katalogene tb og sim i module kataloger ved modulsimuleringer i tillegg til topp nivå simuleringer.
- Katalogene syn og pr vanligvis bare i Top modulen.
- Sim har en katalog for hvert simulerings case.

- Lange simuleringer kan kjøres utenfor ClearCase med ”external view” av VHDL source filer i ClearCase ”vob”.
- Syntese og P&R kan kjøres utenfor ClearCase versjonskontroll. All kildefiler og resultatfiler (vanligvis ikke mellomresultater) versjonskontrolleres med ClearCase.
- Kun tekst filer (XEmacs med VHDL editor).

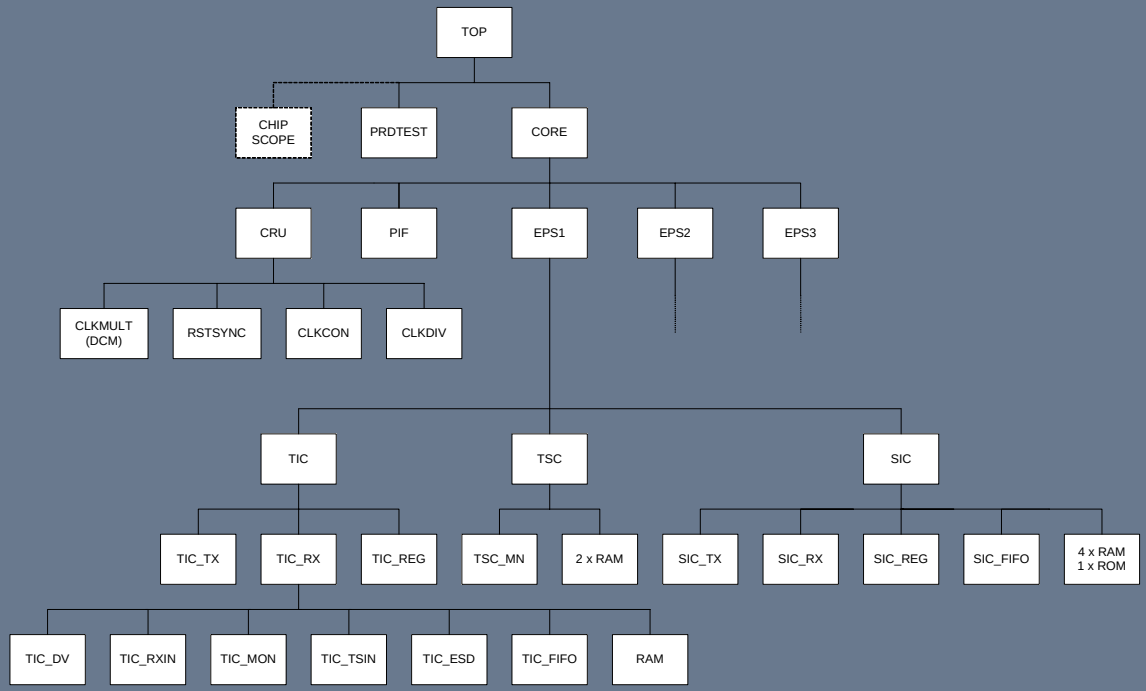
Design hierarki

KONGSBERG

- Ren ”rtl løvnode” og ”str” modul struktur.
- 8-12 moduler på hvert nivå; ikke for små og ikke for store … .. .
- Navn regler for filer viser type:

  (?_ent.vhd, ?_rtl.vhd, ?_str.vhd,?_pck.vhd, ?_bdy.vhd, osv.)
- Adresse map og globale design typer + konstanter i en package.

Design eksempel



- Teknologi (Spartan3, Virtex4, osv.) i modulene Top og CRU, og RAM/ROM i ”str” arkitekturer.
- ChipScope i for eksempel Spartan3_1000 for lab. test, men fjernet i pin kompatibel Spartan3_400 pga. for få RAM moduler.
- Asynkron prosessor aksess til registre og RAM gjennom PIF.
- Interrupt til prosessor via PIF modul.
- Synkron intern bus til alle registre i REG og RAM moduler.
- Modifisert gjenbruk av VHDL kode for REG modulene.
- Bare antall registre og register navn forskjellig

VHDL packages

KONGSBERG

- Memory map og felles typer + konstanter (syntetiserbar)
- Project testbench
- KDA testbench
- Std_developerskit.iopak

Memory Map package

KONGSBERG

- Prosjekt VHDL memory map package for
  - FPGA syntese kode
  - Modul og toppnivå testbenker
  - Kilde for SW memory map
- Eksempel:
  - subtype reg_addr is std_logic_vector(15 downto 0);
  - constant regA : reg_addr:= x”0000”; -- U8
  - constant regB : reg_addr:= x”0001”; -- U8
  - constant regC : reg_addr:= x”0002”; -- U8
  - constant regD : reg_addr:= x”0004”; -- U16
  - constant regE : reg_addr:= x”0008”; -- U32

Parallellisering og pipelining

KONGSBERG

- Design blir vanligvis delt i overordnet kontroll struktur og datapath.
- Ferdigstille overordnet kontroll struktur først (”mem” versjon)
  - Eksterne prosessor grensesnitt (PIF).
  - Intern aksess til registre, RAM, osv.
  - Ekstra funksjonalitet for å teste interrupt. Legger til registre slik at test SW kan styre interrupt.
  - Evnt. prosessor kommunikasjon mellom intern og ekstern prosessor.
  - ”Komplett” krets med Top, Core, CRU, osv. Intet ”blindspor”.
  - Dummy moduler hvor alle innganger er koblet mot alle utganger evnt. gjennom et register (pga. verktøy problemer).
  - SW må lage PIF test program for REG/RAM aksess og interrupt testing.
- Datapath og datapath kontroll
  - Full eller evnt. inkrementell release av datapath (f. eks. EuroCom og E1).
  - Evnt. inkrementell verifikasjon
  - Kan gi endringer på moduler som inngår i ”mem” versjonen.
- Evnt. dummy versjon for PCB testing som verifiserer FPGA lasting.

Simulering vs. lab. debug

KONGSBERG

- Meget viktig med gjennomsimulert design og gode testbenker før lab. testing begynner.
- Alle warnings fra simulatorer, syntese og P&R verktøy skal være fjernet, eller forklart og bestemt at de ikke skal/kan fjernes. Føles av og til som ”Design for DAK” (DFD ☺).
- Bruker ChipScope (el.lign.) eller logikk analysator med intern probing (f. eks. Agilent Dynamic Probe) for å finne interne FPGA bug’s, men mest av alt for å avsløre feil/misforståelser i ekstern funksjonalitet og timing.
- Feil i design og testbenk skal identifiseres ved simulering og deretter rettes og simuleres før ny lab. testing. **Ikke tro, men vite at denne feilen er rettet!!**
- Simuleringsmiljøet brukes dermed aktivt under lab./system test.

Konklusjon

KONGSBERG

- Utviklingsprosess med koderegler, navnregler, sjekklister, dokumentasjonsmaler, filstruktur, scriptmaler, osv.
- Vedtatt design metodikk følges og kontrolleres i design reviews.
- Gjennomsimulert design.
- Parallellisering og pipelining av utvikling og lab. test (f. eks. dmy, mem, EuroCom, E1, final).
- Simuleringsmiljøet brukes aktivt ved lab./system test