Altera Cyclone V

NIOS II – is this the processor is that Cyclone V? it doesn’t say NIOS II on the processor anywhere?

Intel FPGA is this the processor? Or is one of the other chips connected to the processor the fpga? Or is part of it a programmable interface and part of it is a real processor fpga?

AtlasSOC DE0-Nano-SOC – this is the whole board? SOC

Avalon – separate from all these things? It’s not the processor just the bus the processor uses? Is it like a protocol of how a bus functions or just like the brand that created this bus?

Why in lab 2.0 do we connect clk reset and external connection in our vhdl but not s1 memory which is in our qsys/platform designer thing?

Pin planner diagram?

Navigating the documentation:

* I just found out that W15 is LED0, and you put W15 into the pin assignment for LED 0 in Quartus, but I’m not sure what that means? Like in microcontroller it made sense that you had macros with addresses the documentation said which macros are for which addresses. What are these pin assignments? What’s pin W15? Does that stand for some address in memory, like a C macro?
* Cyclone banks?
* Opposite to the pins (inside the chip) in the diagrams? Are those selectable modules?

Avalon bus slides, timing diagrams in 37 38… -> nothing is transferred right? Because read\_n is low so reading is not enabled

HPS = hard processor system: not synthesized on fpga fabric but real processor

VHDL:

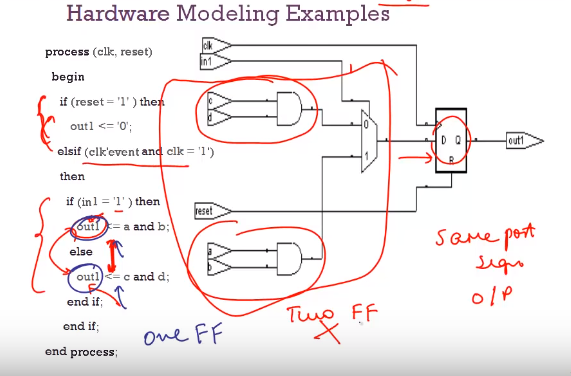
Where is avalon bus in quartus? What/where is SRAM? What does epcs do?

Are registers in VHDL signals?

What are the signals in platform designer? Like readvaliddata, Reset\_n, write, read, etc.

“process” are these like functions? When are they called?

Processes—sequenctial

* Processes execute in parallel (concurrently), statements inside the process execute consequentially
* Process(A, B, C, D) <- this is called sensitivity list. What are the changes in signals that my process is sensitive to. NOT what signals are changing in my process but if a signal changed would that effect another signal in my process (if yes then it should be included into the sensitivity list).
  + Sensitivity list only contains static signal names for which you can read. If your entity port has an output signal it can’t be in sensitivity list (can’t read from output)
* Ex:
  + Process(A, B, C, D)
  + Begin
    - Z <= A and B; --notice Z is not in the sensitivity list
    - Z <= C and D; --the last driver for Z will be the one that it ends up with
  + End process
* Signals assigned to process are only updated after the whole process suspends (finishes)
* Processes is repetitive execution of its sequence of statements
* The simulator however runs the process whenever one of the signals in sensitivity list changes
* Can either of sensitivity list or Wait statement
* Can use if statements (like when else statement (see below)) or
* case statements (like with statements in concurrent/parallel part (see below)) can also use ranges ‘when 0 to 10 =>’
  + also need a default (if you don’t want to do anything just used ‘when others => null’
* Again implemented as cascade of multiplexers (if else) or just one multiplexer (case)
* Combinatorial process: combinational logic x<=(a or b) and c
* Clocked process:
  + Process(clk)
    - Begin
    - If (clk’event and clk=’1’) then –clk’event called “clock tick event”
      * Q<=D;
    - End if;
  + End process.
  + “If (clk’event and clk=’1’)” specifies a RISING EDGE!
    - Can also use if “rising\_edge(clk)
  + “If (clk’event and clk=’0’)” specifies a FALLING EDGE!
  + Any assignment within clk’event always generates a flip flop in the hardware
    - Unless you have two separate assign statements to same signal (like if signal then a<=‘0’ else a<=‘1’) then just one flipflop with a gate
    - 
  + DON”T HAVE ELSE STATEMENT IN CLK process—just confuses the hardware
    - But you can have if clk’event inside another if else statements
    - You can have if elses inside if clk’tick but not on clk’event’s level
* Note: if you don’t specify an else condition for your if statement you get a latch in the hardware—try to avoid a latch unless you know you need it
  + Latch is combinatorial circuit which necessarily has feedback to hold the output of the previous value for the unspecified state/conditions (so the computer assumes you necessarily want to hold the previous value, even if it is not true now (if you don’t specify else)
  + Latches infer feedback and can cause difficulties in timing analysis and test insertion applications (most synthesizers provide warnings of inferred latches)
  + Specifying else will use mux
* Inside a process can declare a variable and assign the variable with := (not <=)
  + Temp : = a or b
  + Have to consume variable inside of process
  + Can use temp as connections that need values immediately
* Two processes cannot edit the same signal

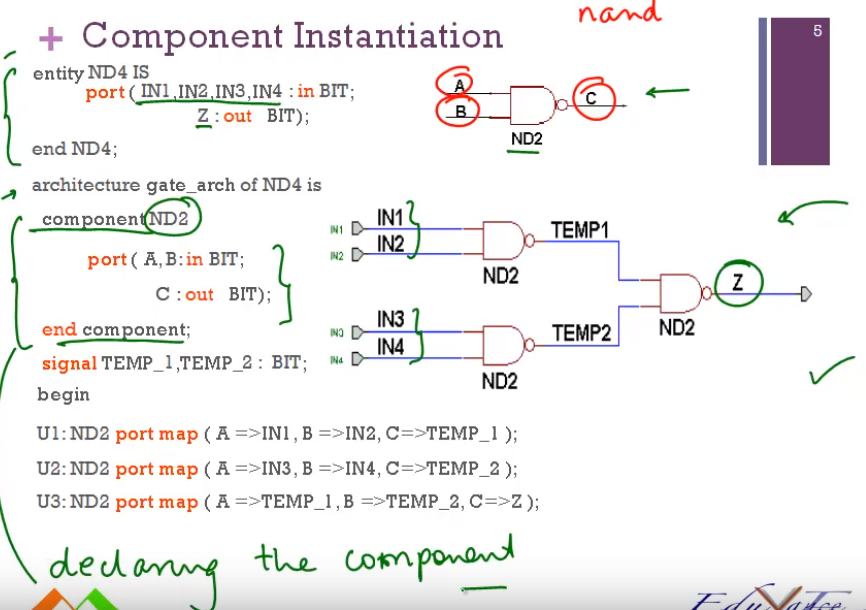
Reset forces signals to 0

Entity = interface

Architecture = implementation

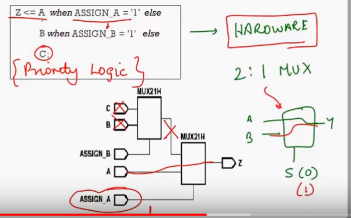
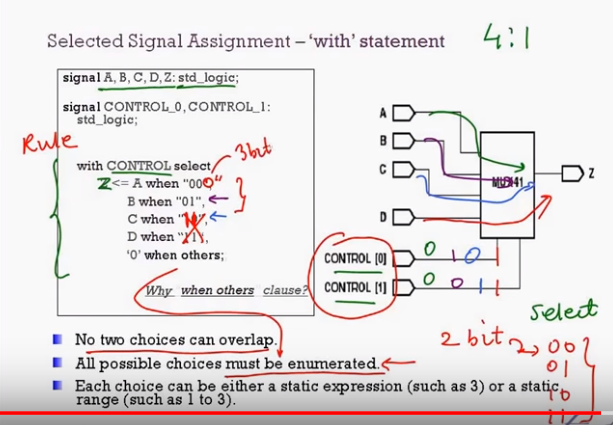
You can have only one entity but can have multiple architectures. –you can specify that in the configuration file assuming that you have more than on architecture

3 styles of modeling architecture:

* Structural and data flow🡪 internal connections are clear and straightforward, used for smaller designs. Structural is components, data flow is like using AND and OR keyword instructions
* Behavioral -> for larger designs you will describe general behavior (combine structural and data flow). Behavioral is using like if statements, cases statement, loops etc.
* Structural uses components
  + Keyword ‘portmap’ automatically maps signals to ports
  + Ex
  + Architecture struc of \_entity\_ is
    - Component XOR1
      * Port(P,Q : in BIT; R : out BIT);
    - End component
    - Begin
      * X1 : XOR1 port map(A, B, SUM); -- P<=A, Q<=B, R <=SUM
    - End
  + 

Driver is term that gives a value to signal or port

Concurrent (parallel not within process) sections

* Z<=A Concurrent signal assignment produces one driver for each signal assignment
  + Meaning when your in concurrent namespace (not in a process) everytime you assign something to a signal that’s a driver
* Can’t do Z<=A; Z<=B;
* Z<= A when assign\_a=’1’ else
  + B when assign\_b=’1’ else
  + C
* ^^This is implemented as 2 2:1 Muxes
* When else gives priority logic (first thing is priority check)
* With statement is like a case statement and when else is like an if
  + Need a default ‘when others’ with with statements
  + 
  + With statement is parallel logic with 1 multiplexer, when else is cascade of multiplexers

V HDL -> Compiler -> Synthesis tool (converts high level behavior model (if statements) into gates)

Signals are wires in circuit! Inside the architecture.

* You can connect components together with signals, they communicate changes
* Think hardware 🡪 don’t think of them as variables and objects think of them as wires and components

Combinational feedback: X <=X+Y (DON’T DO THIS THIS IN HDL (for outputs, you can do this if x is an internal signals). X is not held in a register and clocked like in C programming but immediately fed back into the input and will keep growing!

FSM

* Goes through fixed number of states and has fixed number of I/O combinations
* Conditions coming out of state
  + Mutually exclusive
  + Conditions must be all inclusive (cover all possibilities)
* NSL: Next state logic—which state to go to under what conditions
  + Modeled with if-else statements
* SM: state memory—remembers what state is in
  + Case-statements can model states.
  + State variables are constants
* OFL: output function logic—what to do in what state under what conditions
  + Conditional assignments

For writing, the write cycle should be 80 ms.

Don’t use more than 8-16 burstcount transfer