TUTORIAL WHDL:

Lista osjetljivosti - nalazi se u bloku process, to je ona lista signala cijom se promjenom utjece na promjenu izlaza tj. nalaze se simulatoru da ponovno proracuna sva izlaze koji ovise o tim signalima

```
process()
...
end process;
```

```
process(A,B, C...)
...
end process;
```

U blok process se mogu ubaciti svi signali koji su u sklopu i to bi radilo normalno ,ali mi trebamo minimalni broj tih signala...

signali koji pripadaju listi osjetljivosti ce se nalaziti unutar bloka process, ako su izvan nisu u listi. Sada cu objasniti postupak kako se traze signali liste osjetljivosti...

```
process(...)
begin
if S='0' then
Qint<= '0';
elsif T='0' then
Qint<= '1';
elsif rising_edge(clk) then
if U= '1' then
Qint<= not Qint;
end if;
end if;
Qout<= Qint;
end process;
```

Prvi signal:

```
if S='0' then
Qint<= '0';
```

S ulazi u listu osjetljivosti jer direktno utjece na Qint koji utjece na izlaz

Drugi signal

```
elsif T='0' then
Qint<= '1';
```

isto kao i gore utjece na izlaz pa je i T u listi

Treci signal:

```
elsif rising_edge(clk) then
if U= '1' then
Qint<= not Qint;
```

ako se dogodi rastuci(padajuci) brid CKL djeluje i mijenja izlaz. CLK ide u listu, dok primjerice U ne ide jer ce se on odviti samo ako se dogodi rastuci (padajuci) brid ,a nece ako CLK drzimo u 1 ili 0;

cetvrti signal:

```
end if;
Qout<= Qint;
end process;
```

Qint isto ide u listu osjetljivosti jer se direktno preslikava na izlaz.

da je recimo ovako:

```
end if;
end process;
Qout<= Qint;
```

onda Qint ne bi isao u listu osjetljivosti jer ne pripada bloku process

Sinkronost/asinkronost sklopa: Sklop je sinkron kada na njega djeluje signal takta odnosno CP(CLK,CLOCK) a asinkron kada ne djeluje.

```
process(clock, S)
begin
if rising_edge(clock) then
if T= '1' then
Qint<= not Qint;
end if;
end if;
if S='1' then
Qint<= '0';
end if;
end process;
```

Sinkroni ulaz

```
if rising_edge(clock)
then
if T= '1' then
Qint<= not Qint;
end if;
end if;
```

ovaj ulaz je sinkron jer djeluje pod clockom

asinkroni ulaz

```
end if;
if S='1' then
Qint<= '0';
end if;
```

Ovaj ulaz je asinkron. Odvija se nakon end if...

Prioriteti asinkronih ulaza: To je u biti gledanje koji od aktivnih ulaza ima prioritet tj. koji ce vise utjecati na konacan izlaz sklopa.

```
process(clk, G, H)
variable sel:std_logic_vector(1 downto 0);
begin
if rising_edge(clk) then
sel:=I&J;
case sel is
when "00"=> Qint<= Qint;
when "01"=> Qint<= '1';
when "10"=> Qint<= not Qint;
when "11"=> Qint<= Qint;
when others=> null;
end case;
end if;
if G='0' then
Qint<= '1';
end if;
if H='1' then
Qint<= '1';
end if;
end process;
```

Prvo sto moramo napraviti moramo vidjeti koji su to asinkroni ulazi

```
if rising_edge(clk) then
...
end if;
if G='0' then
Qint<= '1';
end if;
if H='1' then
Qint<= '1';
end if;
```

Boldani dio oznacuje dio kada je CLK prestao s djelovanjem (end if) i te sada imamo G i H asikrone ulaze. Prioritetniji je H jer kada se G izvrsi jos se mora H izvrsiti i on kakav god G bio moze stavit svoju vrijednost u Qint, znaci H

ako na primjer imamo:

```
if rising_edge(clk) then
...
end if;
if G='0' then
Qint<= '1';
elsif H='1' then
Qint<= '0';
end if;
```

ovdje je drugaciji slucaj tj. G ako se ispuni uopce se nece gledati H niti ce se uzimati u obzir i zato je u ovom slucaju G prioritetniji

```
sel:=I&J;
case sel is
when "00"=> Qint<= Qint;
when "01"=> Qint<= '1';
when "10"=> Qint<= not
Qint;
when "11"=> Qint<= Qint;
when others=> null;
end case;
```

Ovo inace ne spada pod nista, ni sinkrone ni asinkrone niti gledate njegove prioritete ,a ni listu osjetljivosti samo preskocite

Na sta djeluje signal takta:

na padajući brid ==========>>>>> if failing_edge(clk) ...
na rastući brid ==========>>>>>if rising_edge(clk)....
na logičku razinu 0 ==========>>>>>if cp='0'...
na logičku razinu 1 ==========>>>>> if cp='1'...