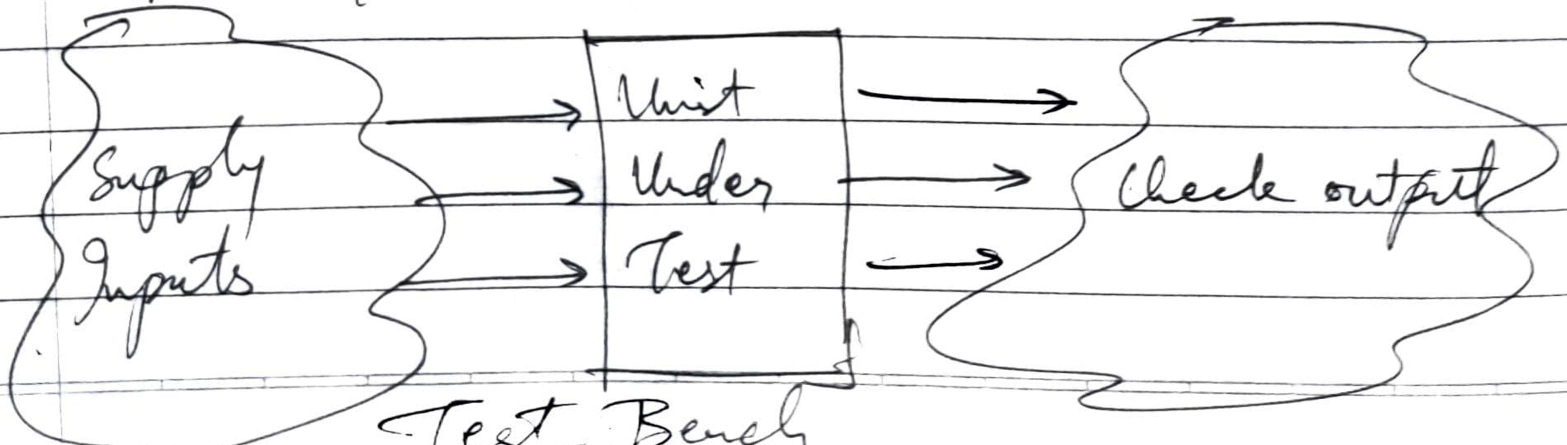
☆ TestBench

* A test bench is a model that is used to exercise to verify the correctness of a h/w model

* Test bench:
1) Generate stimulus
2) Apply this stimulus to the entity under test
3) Compare o/p responces with expected values.



* A typical TB format:

```
entity TB is
end
architecture TB_BEH of TB is
    component entity_under_test
        port(                    );
    end component;
begin
    --generate waveforms using behavioural style
    --apply to entity-under-test as:
    EUT: entity_under_test  port map(port-association);
    --monitor values & compare with expected values
end TB_BEH;
```

☆ Waveform Generation

① Create waveforms to apply at certain discrete time intervals.

② Generate stimulus based on the state of the entity i.e. based on the o/p ~~responses~~ of the entity.

* Ex:

i) Reset <= '0', '1' after 100ns, '0' after 180ns, '1' after 210ns;

ii) Infinite loop waveform

```
process
    clk1 <= 'U' after 5ns, '1' after 10ns, 'U' after 20ns, '0' after 25ns;
    wait for 30ns;
end process;
```

✱ Test bench example : Full Adder

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

entity TB_FA is
end TB_FA;

architecture beh of TB_FA is
    component full_adder
        port(a, b, cin: in std_logic;
                s, cout: out std_logic);
    end full_adder;
    signal a, b, c: std_logic := '0';
    signal cout, s: std_logic;
begin
   UUT: full_adder port map(~~a=>~~ a, b, c, s, cout);
   stim_process: process
   begin
      wait for 10ns;
      A <= '1';
      b <= '0';
      Cin <= '0';
       wait for 10ns;



       --write few conditions
         :
         :
         :

   end process;
end beh;
```