DESIGN OF A ROBOT

- Homework-1
- Designed by Shivoo Koteshwar and Team.

OBJECTIVE

To design a robot which loads and unloads an object to the stations depending on the request. The sensors connected to the robot will sense the request and initiate the correct sequence of operation.

PROJECT TEAM

- Shivoo Koteshwar
- Seyda Mohsina
- Farjana Ahad
- Morsheda Khatun

SPECIFICATION

The robot under design has 5 degree of movement -

- Base movement (A)
- Shoulder movement (B)
- Arm movement (C)
- Wrist Movement (D)
- Claw movement (E)

Fig 1.1: Classic Picture of the Robot under Design

The loading station exists vertically up and keeps loading the Robot after each execution of unloading. The unloading stations are located at either side of the Robot-two on each side.

Figures in the subsequent chapters and theory of operations will give more insight to the flow of execution:

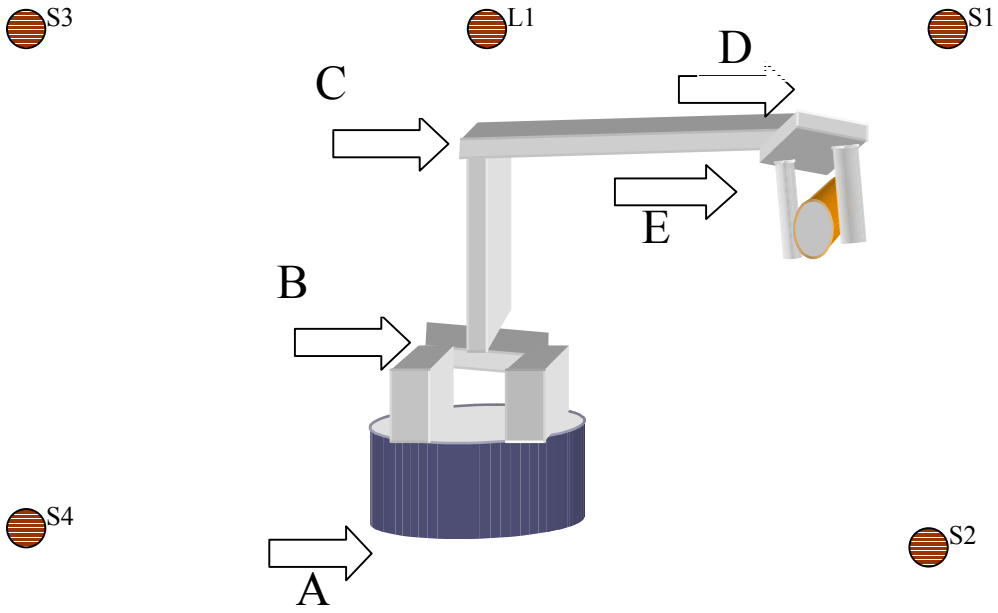


Fig 1.2: Generalized Picture of the ROBOT holding an object:

THEORY OF OPERATION

The setup contains two stations in right (S1 and S2) and two stations on left (S3 and S4). There exists one Loading station (L1) at the top, which is basically used for loading object to the arm of the Robot. The sensors in the Robot sense the request made by the stations and unloads the data to the respective station.

As defined earlier there are 5 degrees of freedom to the Robot and each has two possible orientations. The control signals to the motor define the movement of these parts.

Each of the joints performs 3 actions (Except the claw E) - Stay, Obtuse angle turn and acute angle turn. So two bits are required to represent each of the movement.

A	B	C	D	E
00 01 10	00 01 10	00 01 10	00 01 10	0 1

For the Base A:

00 => Stay (Don't move)
01 => Turn right by 45 Degrees
10 => Turn left by 45 Degrees

For the Shoulder B:

00 => Stay (Don't move)
01 => Turn down by 45 degrees (Acute angle)
10 => Turn up by 45 degrees (Obtuse angle)

For the Arm C:

00 => Stay (Don't move)
01 => Turn down by 45 degrees (Acute angle)
10 => Turn up by 45 degrees (Obtuse angle)

For the Wrist D:

00 => Stay (Don't move)
01 => Turn down by 90 degrees (Horizontal position- For Loading)
10 => Turn up by 90 degrees (Vertical position - For Unloading)

For the Claw E:

0 => Hold
1 => Release

1. Loading Sequence:

00 00 00 00 1	- Initial State
00 00 00 01 1	- Change the wrist D to horizontal position for loading.
00 00 00 00 0	- Once in horizontal pos., load and close Claw E.
00 00 00 10 0	- Change the wrist D to vertical position for unloading
00 00 00 00 0	

Timing Requirement:
As it takes some time for loading, the robot waits 1 clock cycle more before it closes the claw. (3 step in the Load sequence above)

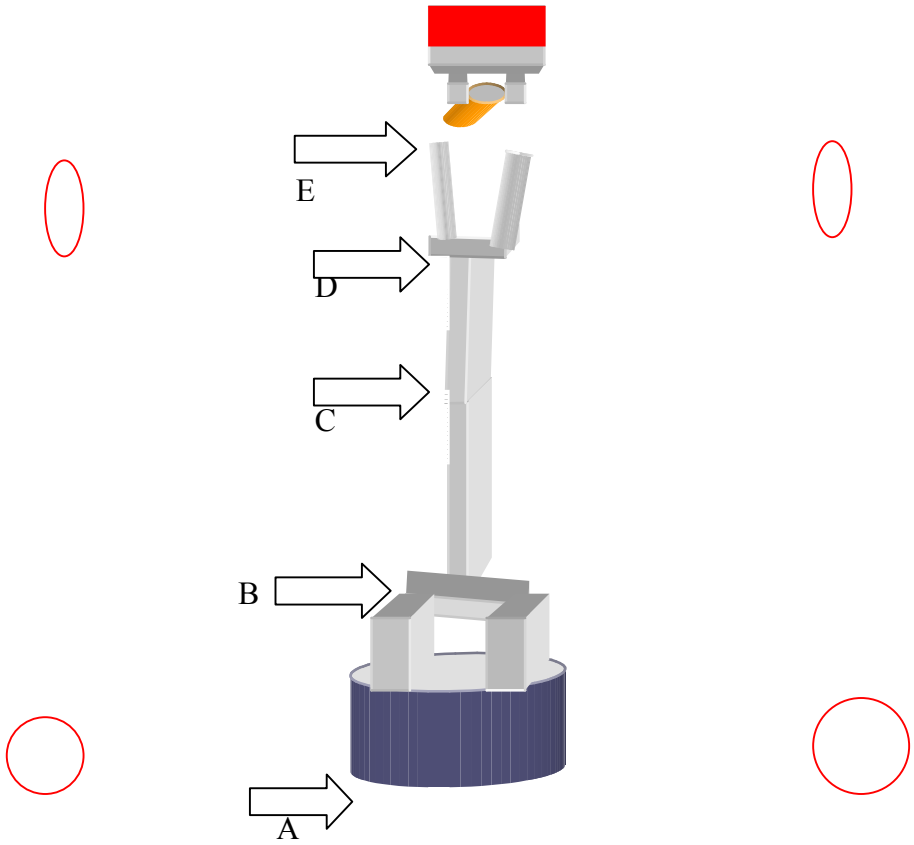


Fig 1.3: Initial State of the ROBOT just before Load sequence:

2. Unloading to S1 Station Sequence:

00 00 00 00 0	- Initial State after loading
01 01 00 00 0	- Turn A right by 45 and B down by 45 degrees.
00 00 00 00 1	- Release claw E.
10 10 00 00 1	- Turn A left 45 and B up by 45 degrees.
00 00 00 00 1	- Repeat the load sequence

Timing Requirement:
None- It assumes that when S1 makes a request, it is ready to take the object.

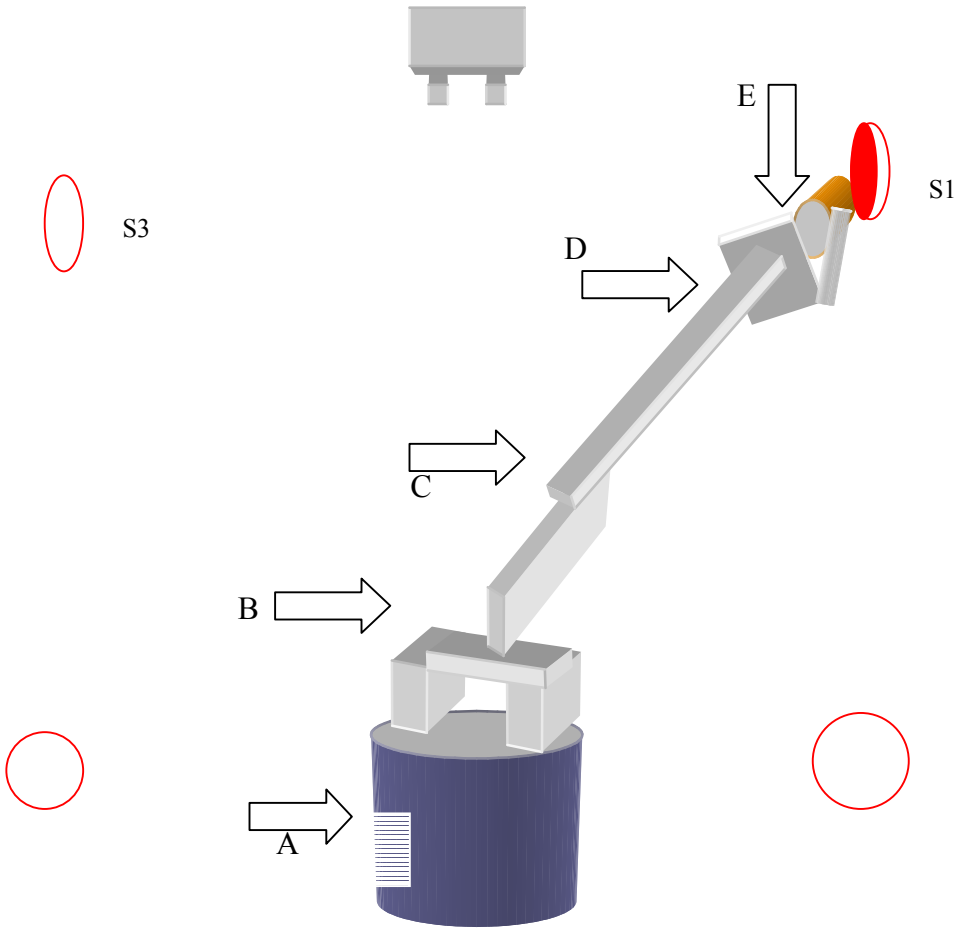


Fig 1.4: Position of Robot at S1 station:

3. Unloading to S2 Station Sequence:

00 00 00 00 0	- Initial State after loading
01 01 00 00 0	- Turn A right by 45 and B down by 45 degrees.
00 00 01 00 0	- Turn C down by 45 degrees.
00 00 00 00 1	- Release claw E.
00 00 10 00 1	- Turn C up by 45 degrees.
10 10 00 00 1	- Turn A left 45 and B up by 45 degrees.
00 00 00 00 1	- Repeat the load sequence

Timing Requirement:
Here though A, B and C could have been moved simultaneously, we are moving A and B simultaneously and then moving C in the next clock edge just to decrease the mechanical tensions on the robot.

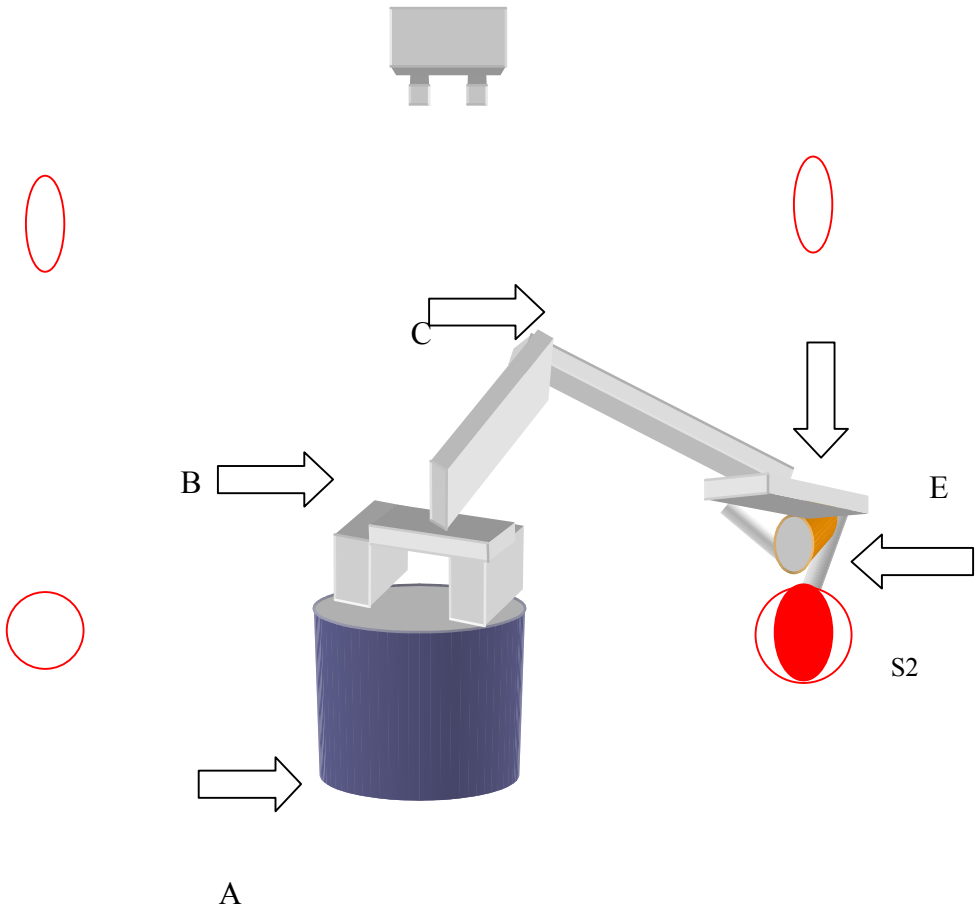


Fig 1.5: Position of Robot at S2 station:

4. Unloading to S3/S4 Stations Sequence:

Note that S3 unloading is symmetrical to S1 unloading and S4 unloading is similar to S2 unloading, we just have to change the sequence of base A, so the sequence would be:

S3 Sequence

00 00 00 00 0	- Initial State after loading
10 01 00 00 0	- Turn A left by 45 and B down by 45 degrees.
00 00 00 00 1	- Release claw E.
01 10 00 00 1	- Turn A right 45 and B up by 45 degrees.
00 00 00 00 1	- Repeat the load sequence

S4 Sequence

00 00 00 00 0	- Initial State after loading
10 01 00 00 0	- Turn A left by 45 and B down by 45 degrees.
00 00 01 00 0	- Turn C down by 45 degrees.
00 00 00 00 1	- Release claw E.
00 00 10 00 1	- Turn C up by 45 degrees.
01 10 00 00 1	- Turn A right 45 and B up by 45 degrees.
00 00 00 00 1	- Repeat the load sequence

IMPLEMENTATION

We have chosen VHDL as the HDL language to represent our design. The flow chart used to write the code uses the above sequence for each of the operation. For debugging purposes, we will generate S1Done, S2Done, S3Done and S4Done after each of the unload operation at S1, S2, S3 and S4 respectively. Similarly LDone is generated after the load sequence. S1Start, S2Start, S3Start, S4Start and LStart are generated when respective sequence is initiated.

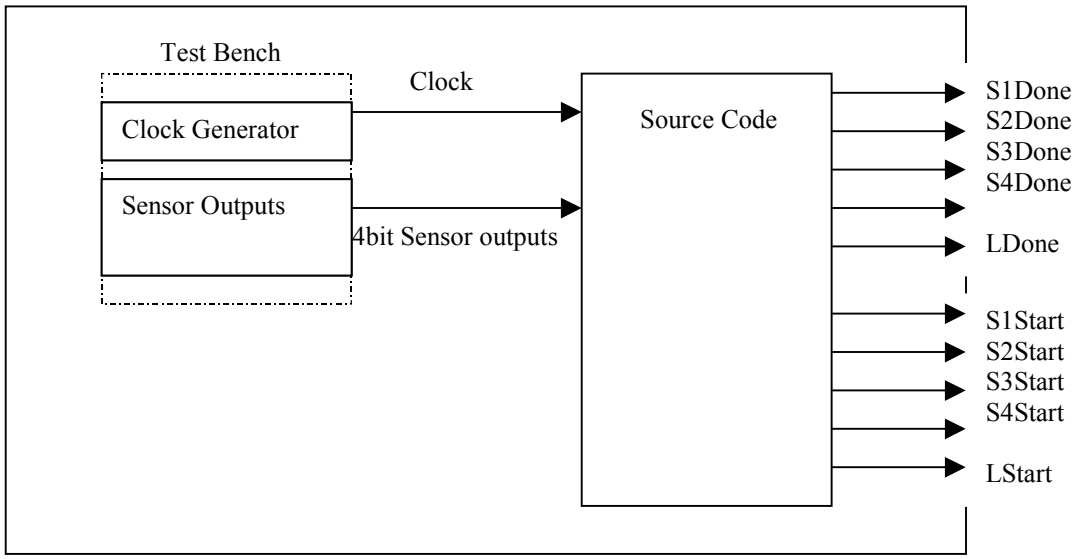
Each of the unloading stations (S1, S2, S3 and S4) will send out a signal if it makes a request. The sensor present in the Robot will pick this up. The convention we have adopted is:

Sensor output is 4 bits and each bit represents a request from one of the stations.

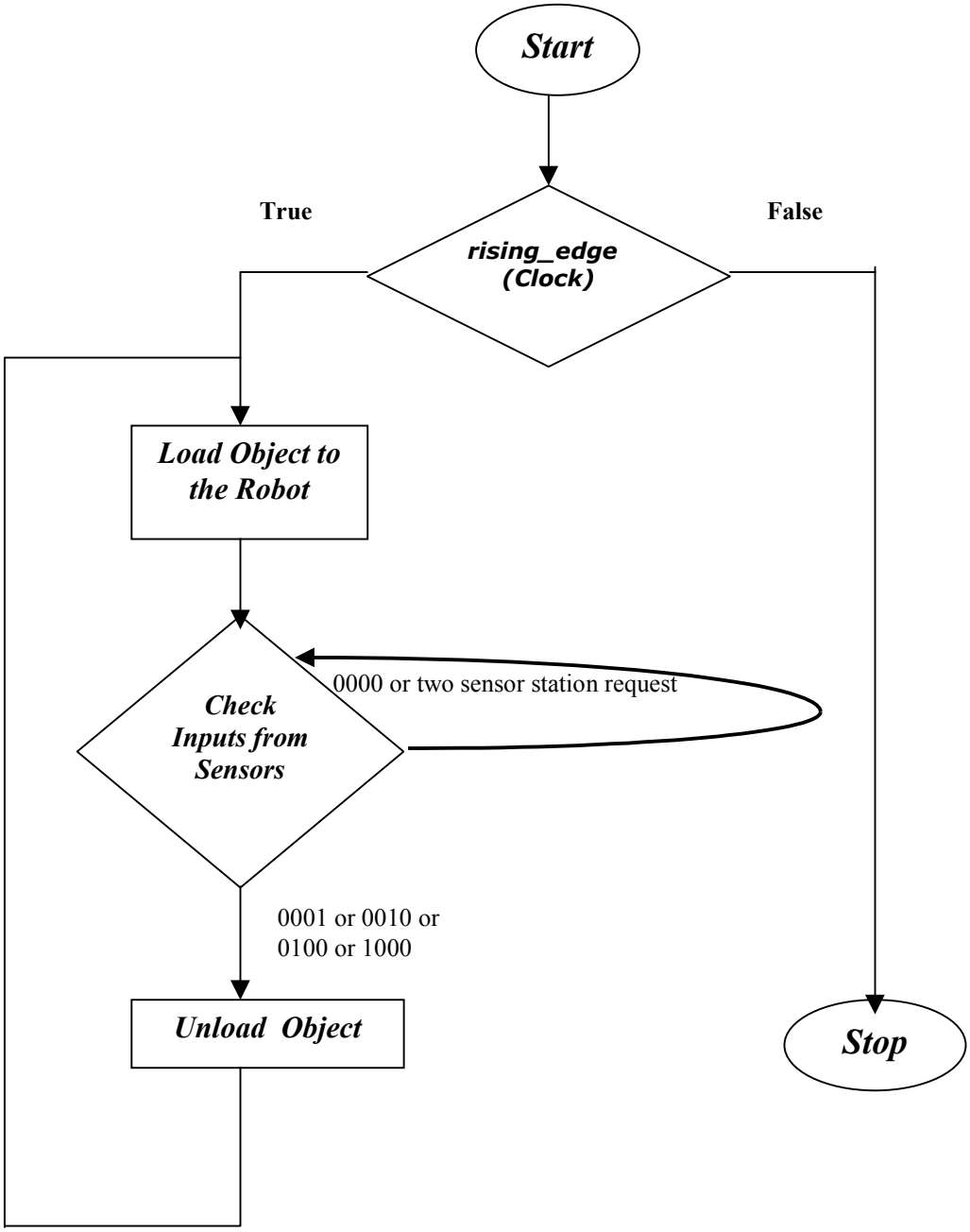
C3C2C1C0 => 0001 => Request made by S1 station.
0010 => Request made by S2 station.
0100 => Request made by S3 station.
1000 => Request made by S4 station.

Note: If the sensor senses two requests at a same time or senses 0000, the robot waits in the decode state until it sees a single request from one of the stations.

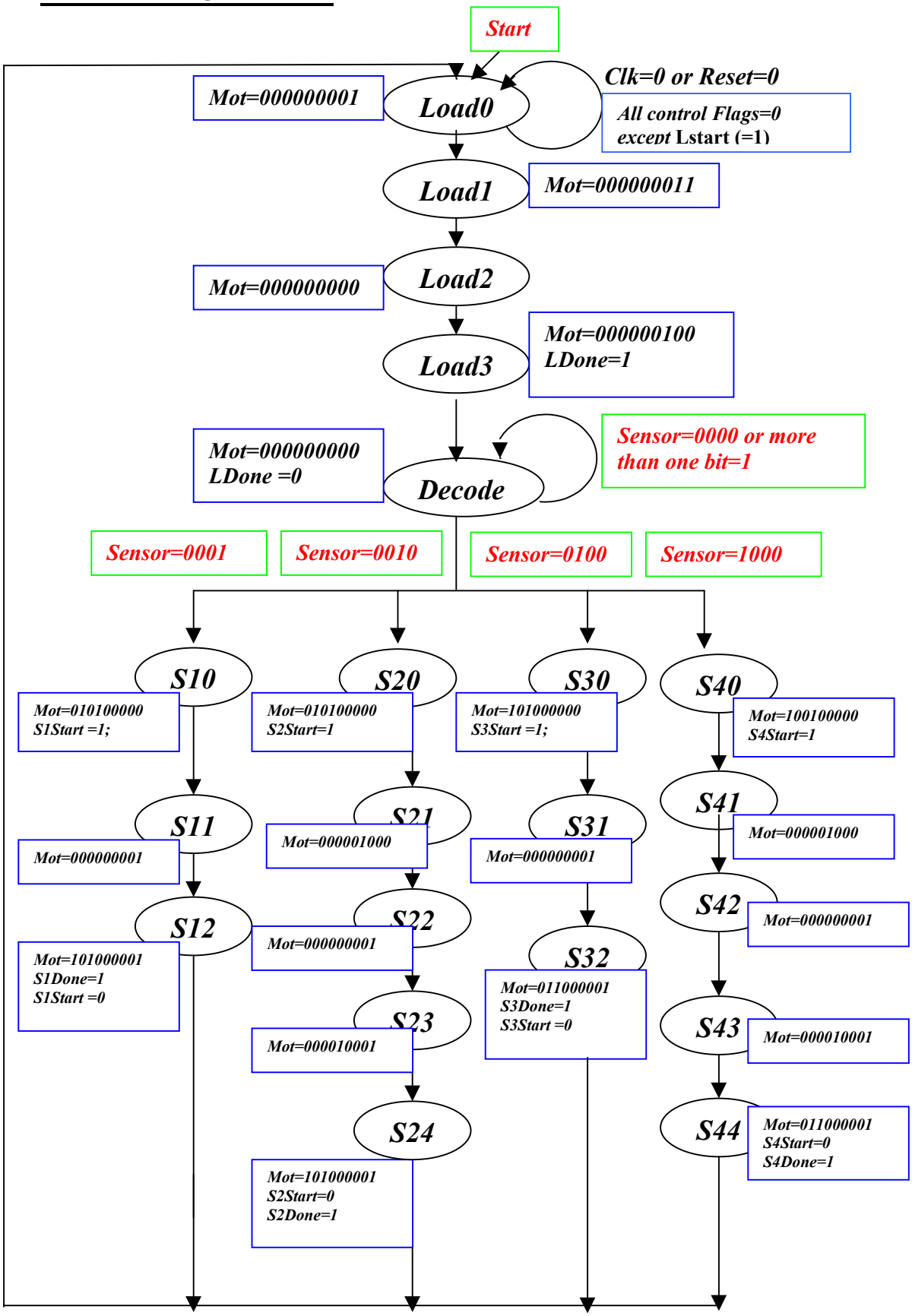
The toplevel VHDL code will look like:



FLOWCHART


Start
True
False
rising_edge
(Clock)
Load Object to
the Robot
Check
Inputs from
Sensors
0000 or two sensor station request
0001 or 0010 or
0100 or 1000
Unload Object
Stop

DETAILED FLOWCHART:



VERIFICATION:

We will generate the clock and the sensor output signal, for the simulation model and check the output flags -S1Done, S2Done, S3Done, S4Done , LDone, S1Start, S2Start, S3Start, S4Start and LStart for each of the operations.

Please refer the SOURCE CODE and Simulation Results section for more details.

SOURCE CODE-VHDL CODE

```
--VHDL source code for ROBOT Design
--Generated By Shivoo Koteshwar
--Functionality of the Program : This is the state machine for the
--                               robot under consideration. It
--                               basically does the load
--                               operation and waits for the signal
--                               from the sensor before it decides
--                               where to unload the object.
--Inputs : Sensor(4 bit), Clock and asynchronous Reset.
--Outputs: Mot(9 bit) and 10Flags which signatures the action.
-------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

PACKAGE myrobot is
  COMPONENT robot port(Mot : OUT std_logic_vector(8 downto 0) ;
                       Sensor : IN std_logic_vector(3 downto 0);
                       Clock : IN std_logic;
                       Reset : IN std_logic;
         S1Done, S2Done, S3Done, S4Done, LDone : OUT std_logic ;
         S1Start, S2Start, S3Start, S4Start, LStart : OUT std_logic ) ;
  end COMPONENT;
end myrobot;
-------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY robot IS
    PORT (
        Mot : OUT std_logic_vector(8 downto 0) ;
        Sensor : IN std_logic_vector(3 downto 0);
        Clock : IN std_logic;
      Reset : IN std_logic;
        S1Done, S2Done, S3Done, S4Done, LDone : OUT std_logic ;
        S1Start, S2Start, S3Start, S4Start, LStart : OUT std_logic ) ;
end robot ;
-------------------------------------------------------------------------
ARCHITECTURE state_machine OF robot IS
    TYPE  state IS (
        Load0, Load1, Load2, Load3,
      Decode,
      S10, S11, S12,
      S20, S21, S22, S23, S24,
      S30, S31, S32,
      S40, S41, S42, S43, S44) ;

   -- SIGNAL present_state : state_type:= Load0;
   -- SIGNAL next_state  : state_type:= Load0;
    SIGNAL present_state : state;
    SIGNAL next_state    : state;

BEGIN
state_comb : process(present_state, Sensor ) begin
   case present_state is
```

```
--Load Sequence is
--00 00 00 00 1    - Initial State
--00 00 00 01 1    - Wrist D to horizontal position for loading.
--00 00 00 00 0    - Once in horizontal pos,load and close Claw E.
--00 00 00 10 0    - Wrist D to vertical position for unloading
--00 00 00 00 0

when Load0  => Mot         <= "000000001" ;
               LDone       <= '0' ;
               LStart      <= '1' ;
               S1Start     <= '0' ;
               S2Start     <= '0' ;
               S3Start     <= '0' ;
               S4Start     <= '0' ;
               S1Done      <= '0' ;
               S2Done      <= '0' ;
               S3Done      <= '0' ;
               S4Done      <= '0' ;

               next_state  <= Load1;

when Load1  => Mot         <= "000000011" ;
               next_state  <= Load2;

when Load2  => Mot         <= "000000000" ;
               next_state  <= Load3;

when Load3  => Mot         <= "000000100" ;
               LDone       <= '1';
               LStart      <= '0';
               next_state  <= Decode;


when Decode => Mot         <= "000000000" ;
               LDone       <= '0';
               if (Sensor = "0000") then
                  next_state   <= Decode;
               elsif (Sensor = "0001") then
                  next_state   <= S10;
               elsif (Sensor = "0010") then
                  next_state   <= S20;
               elsif (Sensor = "0100") then
                  next_state   <= S30;
              elsif (Sensor = "1000") then
                  next_state   <= S40;
               else next_state <= Decode;
               end if;

--S1 UnLoad Sequence is
--00 00 00 00 0    - Initial State after loading
--01 01 00 00 0    - Turn A right by 45 and B down by 45 degrees.
--00 00 00 00 1    - Release claw E.
--10 10 00 00 1    - Turn A left 45 and B up by 45 degrees.
--00 00 00 00 1    - Repeat the load sequence

when S10    => S1Start     <= '1' ;
```

```
            Mot             <= "010100000" ;
            next_state      <= S11;

when S11    => Mot          <= "000000001" ;
               next_state   <= S12;

when S12    => Mot          <= "101000001" ;
               S1Start      <= '0';
               S1Done       <= '1';
               next_state   <=  Load0;

--S2 UnLoad Sequence is
--00 00 00 00 0    - Initial State after loading
--01 01 00 00 0    - Turn A right by 45 and B down by 45 degrees.
--00 00 01 00 0    - Turn C down by 45 degrees.
--00 00 00 00 1    - Release claw E.
--00 00 10 00 1    - Turn C up by 45 degrees.
--10 10 00 00 1    - Turn A left 45 and B up by 45 degrees.
--00 00 00 00 1    - Repeat the load sequence

when S20    => S2Start      <= '1' ;
               Mot          <= "010100000" ;
               next_state   <= S21;

when S21    => Mot          <= "000001000" ;
               next_state   <= S22;

when S22    => Mot          <= "000000001" ;
               next_state   <= S23;

when S23    => Mot          <= "000010001" ;
               next_state   <= S24;

when S24    => Mot          <= "101000001" ;
               S2Start      <= '0';
               S2Done       <= '1';
               next_state   <=  Load0;

--S3 UnLoad Sequence is
--00 00 00 00 0    - Initial State after loading
--10 01 00 00 0    - Turn A left by 45 and B down by 45 degrees.
--00 00 00 00 1    - Release claw E.
--01 10 00 00 1    - Turn A right 45 and B up by 45 degrees.
--00 00 00 00 1    - Repeat the load sequence

when S30    => S3Start      <= '1' ;
               Mot          <= "100100000" ;
               next_state   <= S31;

when S31    => Mot          <= "000000001" ;
               next_state   <= S32;

when S32    => Mot          <= "011000001" ;
               S3Start      <= '0';
               S3Done       <= '1';
               next_state   <=  Load0;
```

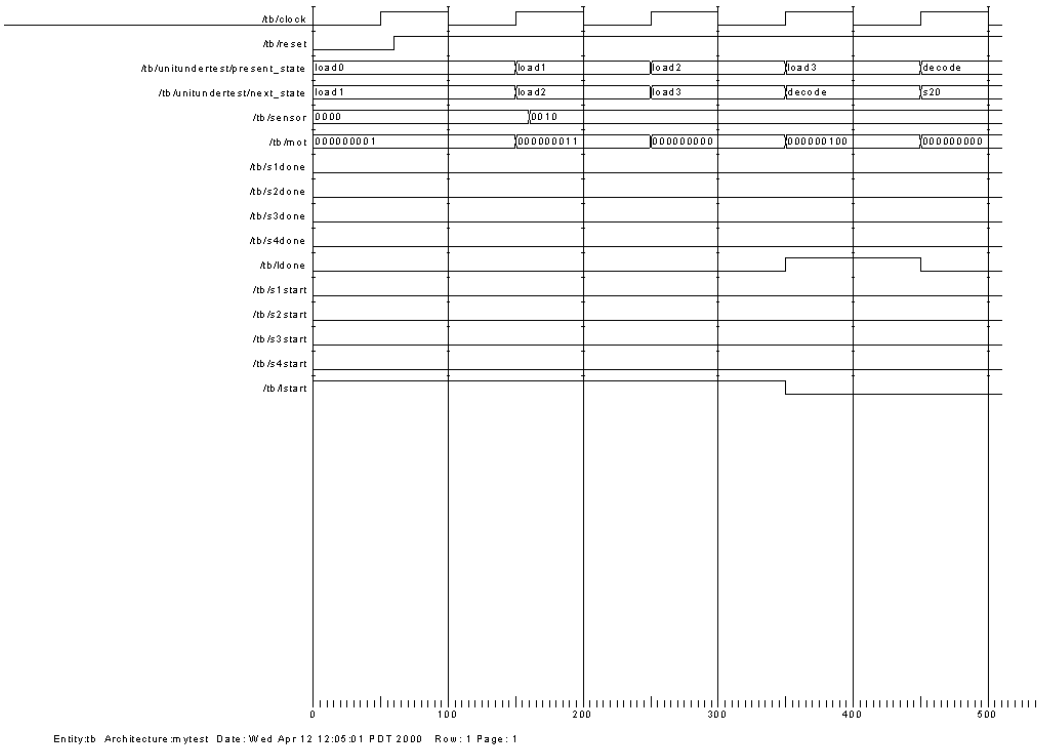
```
     --S4 UnLoad Sequence is
     --00 00 00 00 0    - Initial State after loading
     --10 01 00 00 0    - Turn A left by 45 and B down by 45 degrees.
     --00 00 01 00 0    - Turn C down by 45 degrees.
     --00 00 00 00 1    - Release claw E.
     --00 00 10 00 1    - Turn C up by 45 degrees.
     --01 10 00 00 1    - Turn A right 45 and B up by 45 degrees.
     --00 00 00 00 1    - Repeat the load sequence

     when S40    => S4Start     <= '1' ;
                    Mot         <= "100100000" ;
                    next_state  <= S41;

     when S41    => Mot         <= "000001000" ;
                    next_state  <= S42;

     when S42    => Mot         <= "000000001" ;
                    next_state  <= S43;

     when S43    => Mot         <= "000010001" ;
                    next_state  <= S44;

     when S44    => Mot         <= "011000001" ;
                    S4Start     <= '0';
                    S4Done      <= '1';
                    next_state  <=  Load0;

   end case;

end process state_comb;

state_seq :  process(Clock, Reset) begin
   --if (Clock'event and Clock ='1') then
   if  (Reset ='0') then
       present_state <=Load0;
   elsif  rising_edge(Clock)  then
      present_state <= next_state;
   end if;
end process state_seq;

--end ARCHITECTURE state_machine;  --FOR SOMEREASONS THIS IS NOT
WORKING
end  state_machine;
-------------------------------------------------------------------------
```

TESTBENCH-VHDL CODE

```
--Testbench for ROBOT Design
--Generated By Shivoo Koteshwar
--Mooremachine Design with Two process Blocks.

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY tb is
end tb;

use work.myrobot.all;

ARCHITECTURE mytest OF tb IS
   signal Clock  : std_logic := '0' ;
   signal Reset  : std_logic := '1' ;
   signal Sensor : std_logic_vector(3 downto 0) := "0000";
   signal Mot    : std_logic_vector(8 downto 0) ;
   signal S1Done, S2Done, S3Done, S4Done, LDone : std_logic ;
   signal S1Start, S2Start, S3Start, S4Start, LStart : std_logic ;
begin
unitundertest :  robot  port map( Mot => Mot, Sensor =>Sensor,
                          Clock => Clock, Reset =>Reset,
                          S1Done =>S1Done, S2Done =>S2Done,
                          S3Done =>S3Done, S4Done =>S4Done,
                          LDone =>LDone , S1Start =>S1Start,
                          S2Start=>S2Start, S3Start=>S3Start,
                          S4Start=>S4Start, LStart=>LStart);

Clock <=not(Clock) after 50 ns ;

verify : process
begin
  Reset <='0';
  wait for 60 ns ;
  Reset <='1';
  wait for 100 ns ;
  Sensor <= "0010" ;
  wait for 750 ns;
  Sensor <= "0001" ;
  wait for 750 ns;
  Sensor <= "0100" ;
  wait for 750 ns;
  Sensor <= "1000" ;
  wait for 950 ns;
  Sensor <= "0000" ;
  wait for 750 ns;
  Sensor <= "0001" ;
  wait for 950 ns;
  Sensor <= "0011" ;
  wait for 750 ns;
  Sensor <= "0100" ;
end process;

end;
```

SIMULATION RESULTS

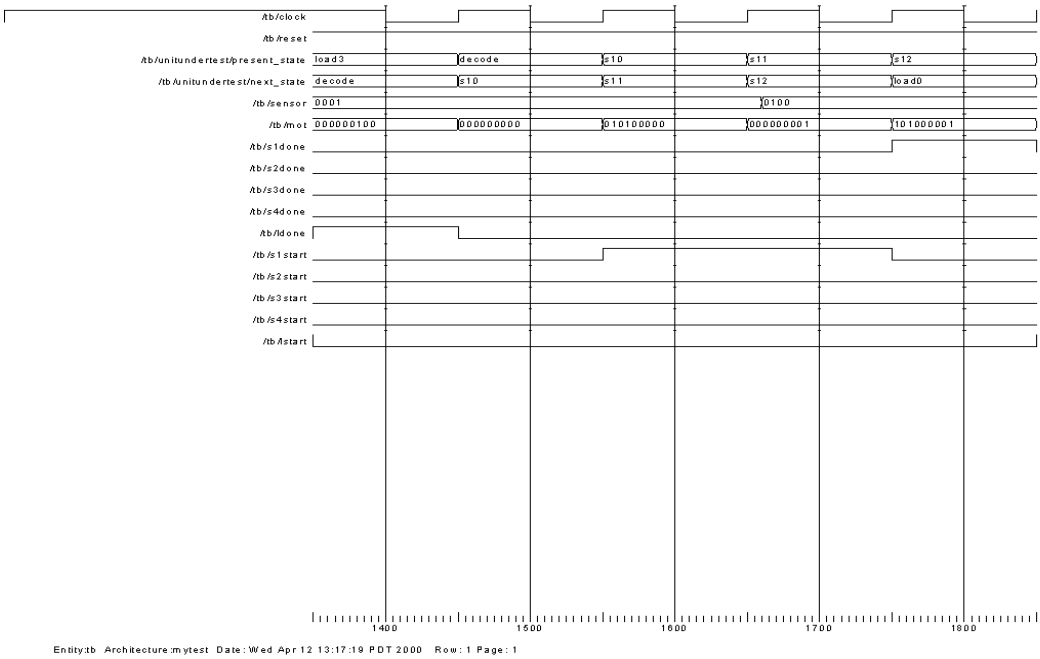
1. LOAD-DECODE SEQUENCE



Please refer the testbench.vhd file to view the input vectors. This graph verifies the load sequence , asynchronous Reset signal operation and Decode sequence.

The charts following this will verify the unload operations at each of the stations depending on the sensor output. When the sensor output is 0000 or any combination where there are multiple requests(0011, 0101, 1110….etc), the machine just waits at Decode state until a valid output is given out by the sensor. The following simulation results are self explanatory.

2. S1 STATION UNLOAD SEQUENCE:

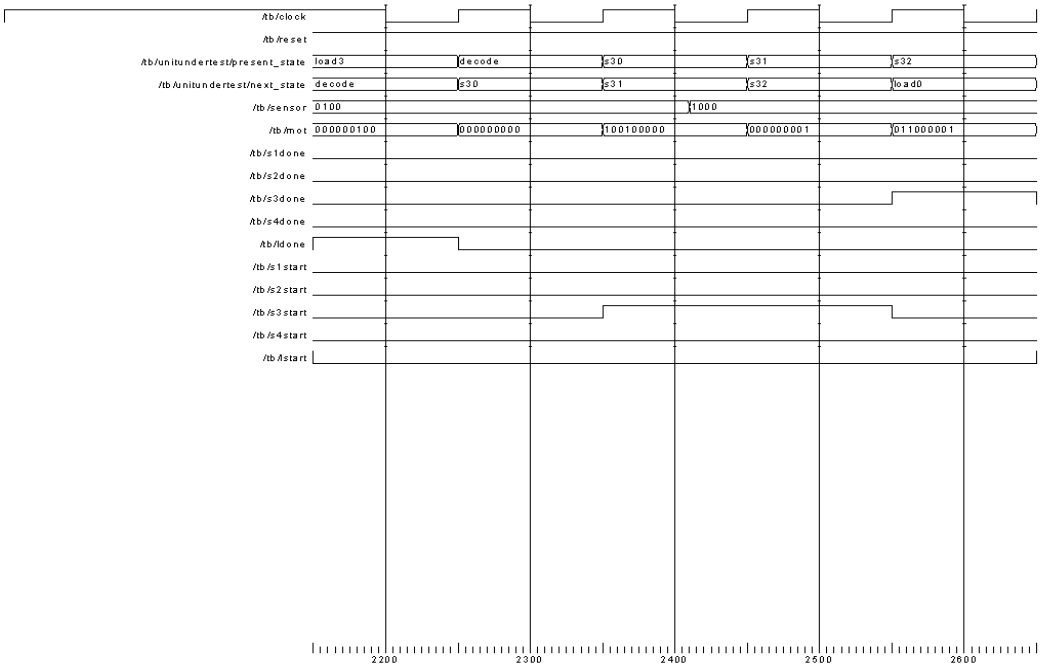


Entity:tb Architecture:mytest Date: Wed Apr 12 13:17:19 PDT 2000 Row: 1 Page: 1

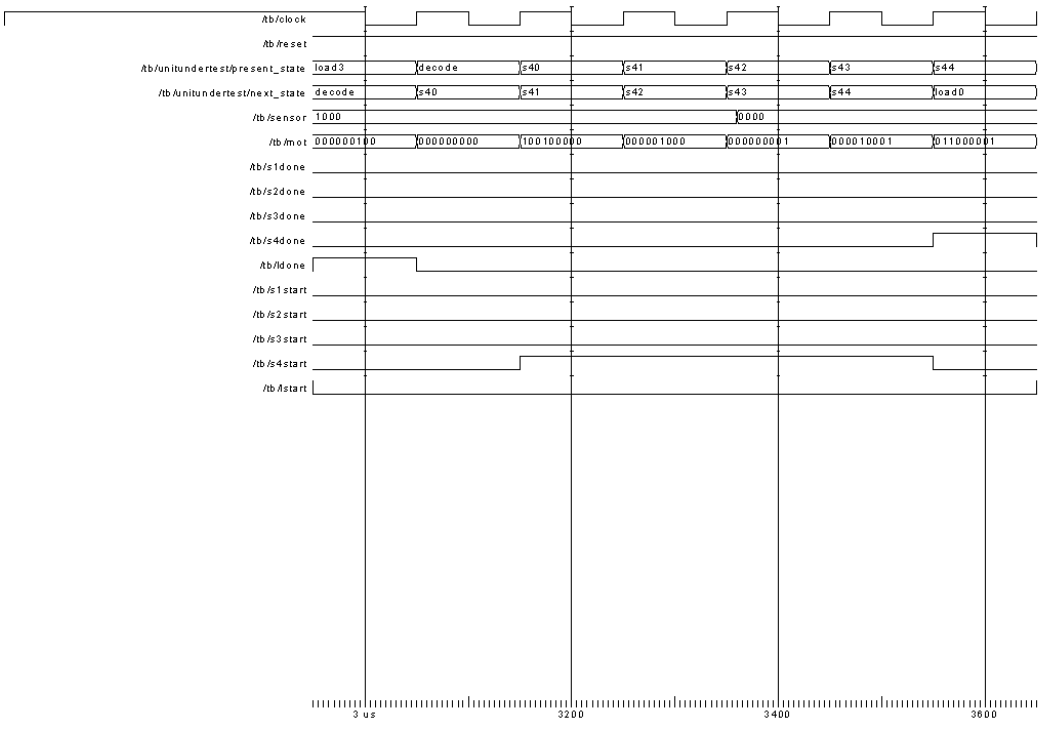
3. S2 STATION UNLOAD SEQUENCE:

Entity:tb Architecture:mytest Date: Wed Apr 12 13:15:11 PDT 2000 Row: 1 Page: 1

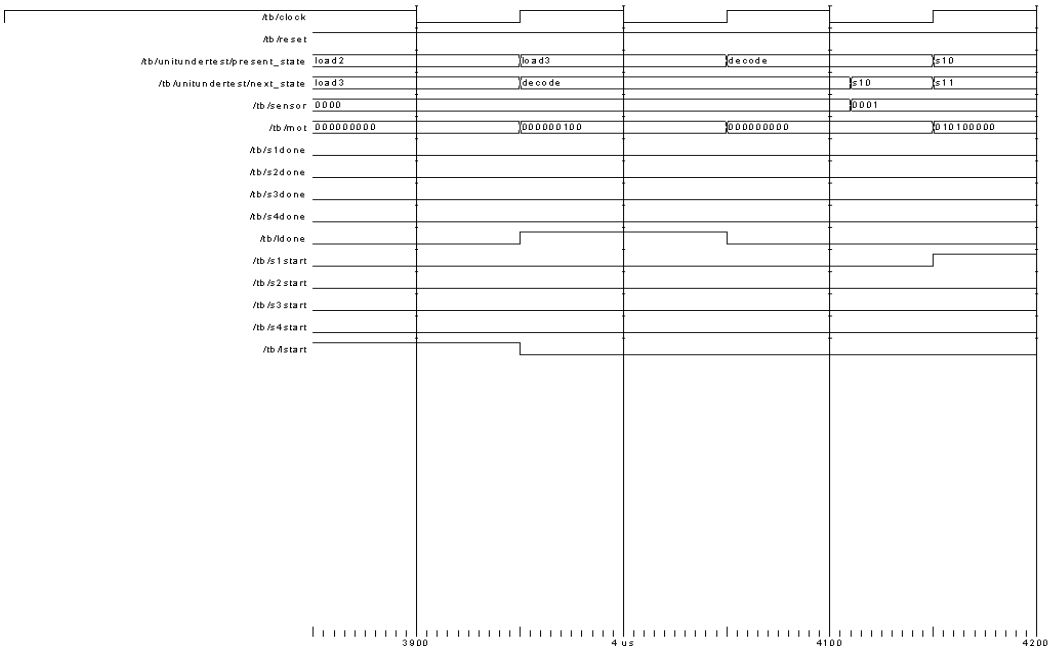
S3 STATION UNLOAD SEQUENCE



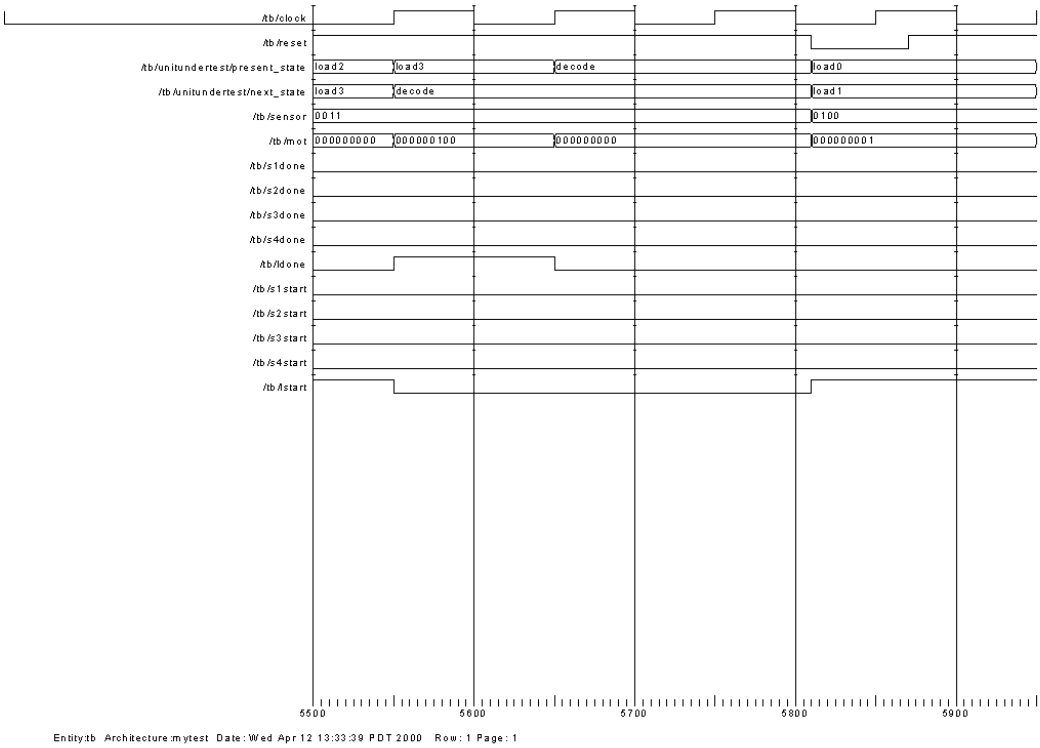
S4 STATION UNLOAD SEQUENCE



RESULTS WHEN SENSOR OUTPUT is 0000



RESULTS WHEN SENSOR OUTPUT is 0011 or WHEN ANY 2 BITS ARE 1



LIMITATIONS

The present design of the Robot entertains only one request at a time. Later this design can be improved by adding a priority encoder and assigning priorities to the requests if they occur simultaneously. The loading station is only at the top of the Robot, this can be changed to make each and every station as loading/unloading stations.

IMPROVEMENTS

Couple of improvements can be made to the present design:

- Design with Priority to the unloading stations can be implemented.
- Interrupt service can be added, so that it stops if the user wants to stop the execution of unloading at one station and initiate unloading at another station.
- Loading at each station instead of central loading station can also be implemented, so that this Robot can be used to transfer objects from one station to another.

We are planning to improve the present design of the Robot and submit it as a part of Homework 2. Any suggestions to the present design will also be incorporated.