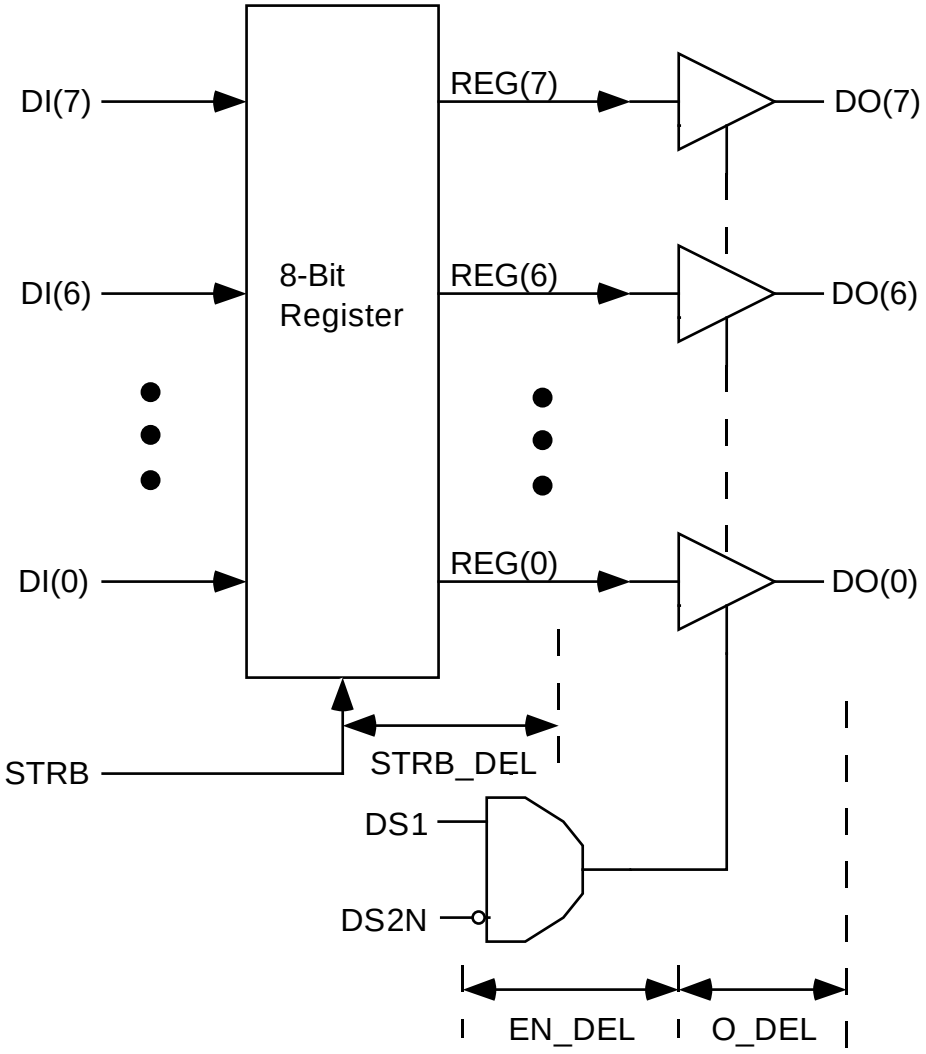
# 11.0 Chip Level Modelling

**Chip Level Model** is a behavioral model of a block of logic in which signal path delays are accurately modeled without resorting to lower-level descriptions.




(a)

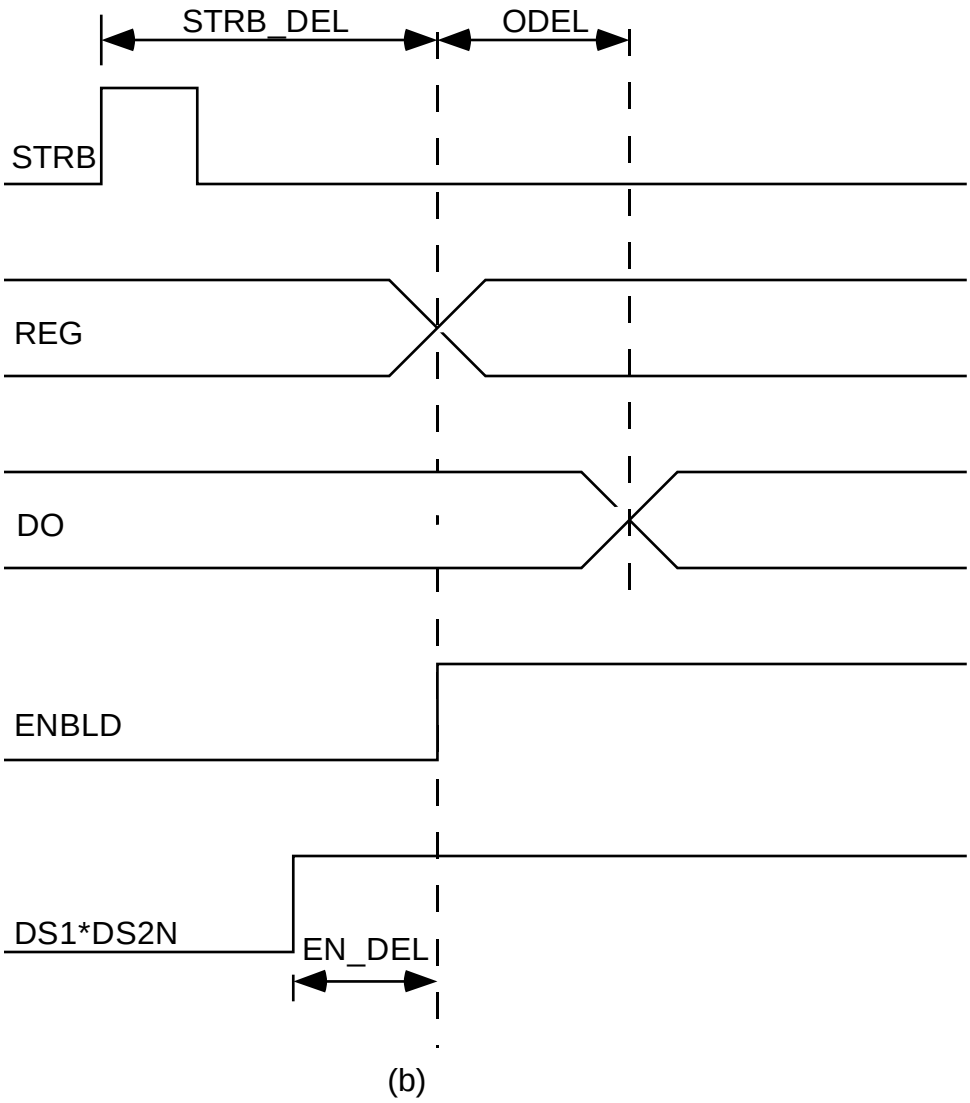



Figure 1. Chip Level Modelling (a) Chip Block Diagram, (b) Timing Diagram.

## Modelling without delay

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY BUF_REG
        PORT(DI: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                DO: OUT STD_LOGIC_VECTOR(7 DOWNTO 0) BUS;
                STRB, DS1, DS2N : IN STD_LOGIC);
END BUF_REG;

ARCHITECTURE lev1 OF BUF_REF IS
        SIGNAL REG: STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
        PROCESS(STRB, DS1, DS2N)
        BEGIN
                IF STRB='1' and STRB'EVENT THEN
```

```
                    REG <= DI;
                    IF DS1='1' and DS2N='0' THEN
                        DO <= REG; ELSE
                        DO <="ZZZZZZZZ";
                    END IF;
                ELSIF DS1'EVENT or DS2N'EVENT THEN
                    IF DS1='1' and DS2N='0' THEN
                        DO <= REG; ELSE
                        DO <= "ZZZZZZZZ";
                    END IF;
                END IF;
        END PROCESS;
END levl1;
```

## Modelling Including High Level Cause and Effect Delays

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY BUF_REG
        GENERIC(STRB_DEL, EN_DEL, ODEL: TIME);
        PORT(DI: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
             DO: OUT STD_LOGIC(7 DOWNTO 0) BUS;
             STRB, DS1, DS2N : IN STD_LOGIC);
END BUF_REG;

ARCHITECTURE data_path OF BUF_REF IS
        SIGNAL REG : STD_LOGIC_VECTOR(7 DOWNTO);
        SIGNAL ENBLD: STD_LOGIC;
BEGIN
A:      PROCESS(STRB)
        BEGIN
            IF(STRB='1') THEN
                REG <= DI AFTER STRB_DEL;
            END IF;
        END PROCESS A;

B:      PROCESS(DS1, DS2N)
        BEGIN
            ENBLD <= DS1 and not DS2N AFTER EN_DEL;
        END PROCESS B;

C:      PROCESS(REG, ENBLD)
        BEGIN
            IF(ENBLD='1') THEN
                DO <= REG AFTER ODEL; ELSE
                DO <= "ZZZZZZZZ" AFTER ODEL;
            END IF;
        END PROCESS C;
END data_path;

ARCHITECTURE data_flow OF BUF_REG IS
        SIGNAL REG : STD_LOGIC_VECTOR(7 DOWNTO);
        SIGNAL ENBLD: STD_LOGIC;
BEGIN
```

```
blk1:    BLOCK(STRB='1' and STRB'EVENT)
         BEGIN
                  REG <= guarded DI AFTER STRB_DEL;            --process A
                  ENBLD <= DS1 and not DS2N AFTER EN_DEL;      --process B
                  DO <= REG AFTER ODEL WHEN ENBLD='1'          --process C
                           ELSE "ZZZZZZZZ" AFTER ODEL;
         END BLOCK blk1;
END data_flow;
```

# Modeling Digital Signals

## Single-bit signals

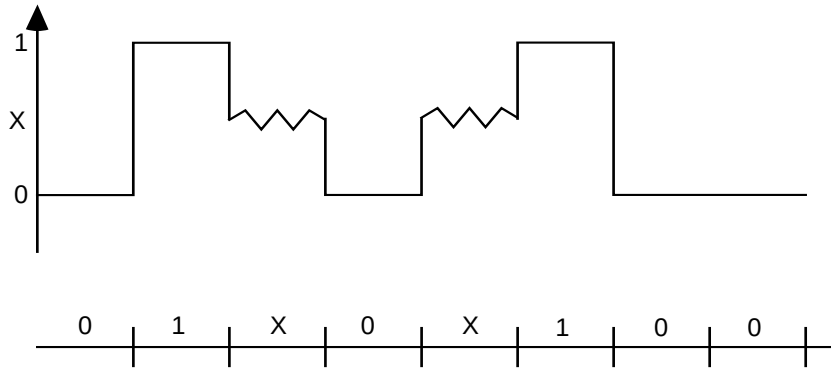



Figure 2. Single-bit Signal Representation.

Single-bit signals is represented by a sequence of 0, 1, X value.

## N-tuple Signals

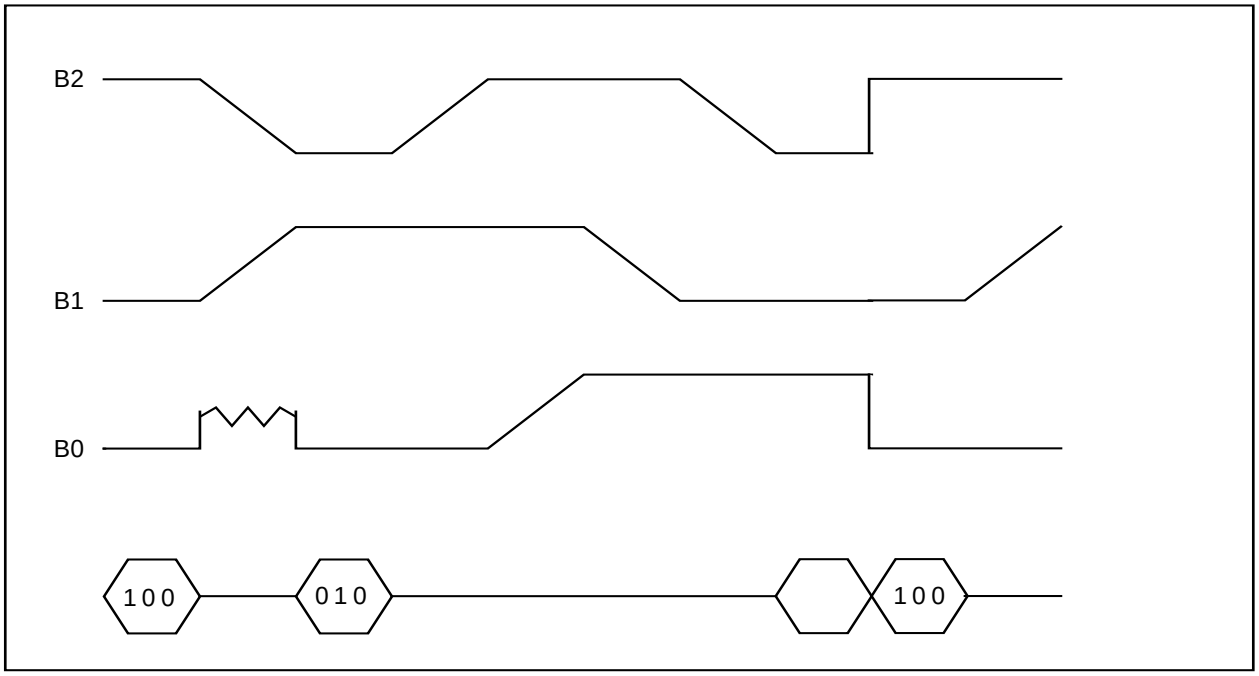


Figure 3. N-tuple Signal Representation.

Representation of n-tuple signal S:

***Double lines*** indicate that S is stable, the stable value is written between the double lines.
*Single line* at ½ indicates that S is unstable, (i.e. some bit of S is in transition).
*Crossing double lines* indicate a sharp transition of signal S, i.e. all bits involved in the transition change sharply at the same time.

## Signal Types and Events

There are two functional classes of digital signals:

Level signal - is a signal that is used to represent information (either data or control information) by the value (or level) of the signal. A level signal carries information only when it is stable at 0 or 1.

Edge signal – is a signal that is used to identify specific points in time. It does so by making sharp transitions, either from 0 to 1 or from 1 to 0 .

There are five different types of events: two types for edge signal and three types for level signal.

The two edge signal events are:

S↑ for positive transition
S↓ for negative transition.

The three level signal events are:

S< transition from unstable to stable.
S> transition from stable to unstable.
S| a sharp transition

**Interval types**

[t1, t2] denotes the time interval between t1 and t2.

ex. [X<, C↑] denotes the time interval between a transition of X to a stable value and a positive transition of C.

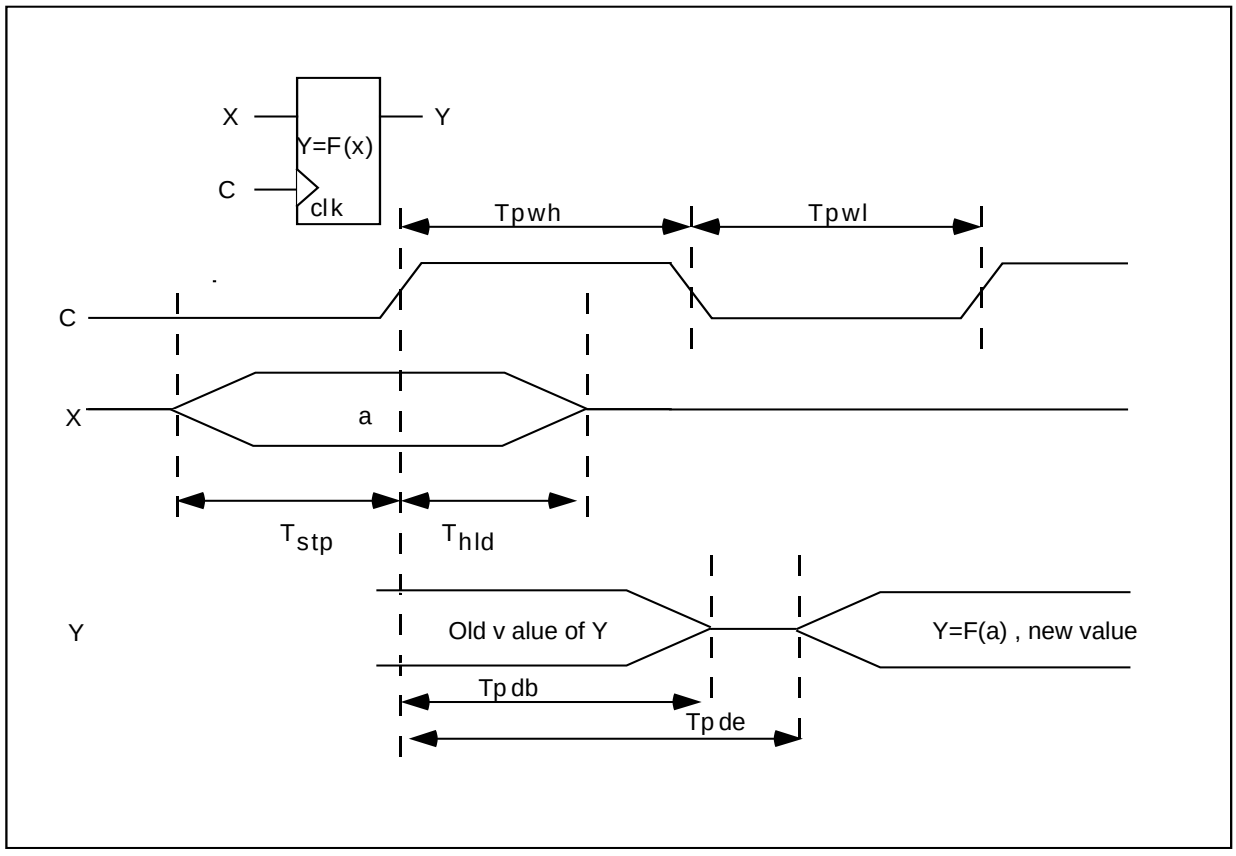



Figure 4. Setup, Hold, and Propagation Delay Intervals.

**Setup interval** – is an interval bounded on the left by the time a level signal word becomes stable and on the right by the time that an edge signal makes a transition. There are four possible combinations:

[X<, C↑], [X<, C↓], [X|, C↑], [X|, C↓]

It is understood that X remains stable throughout the setup interval.

**Hold interval** – is an interval that is bounded on left by a transition of an edge signal and on the right by a transition from stable to unstable of a level signal. There are four possible combinations:

[C↑, X>], [C↓, X>], [C↑, X⏐ ], [C↓, X⏐]

**Propagation delay** – is an interval that is bounded on the left by an event that causes, or contributes to causing, the event that bounds the interval on the right.

The event C↓ causes Y to change value. There are two time specifications:

**Tpdb** – is the propagation delay to the beginning of the transition (some signals start changing). Example, min Tpdb=min [ C↓, Y>]. The specification is given by its minimum value, that is Y> is guaranteed not to occur until at least (Tpdb)min after C↓.

**Tpde** – is the propagation delay to the end ( or completion) of the transition (all signals stabilizes to new value). Example, max Tpde=max [C↓, Y<]. The specification is given by its maximum value, that is Y< is guaranteed to occur before (Tpde)max.

Usually distinction between the beginning and end of the transition will not be made explicitly and one propagation delay (Tpd) will be mentioned, typically it will beTpd=(Tpde) max.

For some devices the propagation delay of a single-bit output for transitions from high to low may be significantly different from that for transitions from low to high. Separate specification may then be given. When several such outputs are grouped together into a word, the worst of the individual specifications should be used. Example given the bit-wise specification:

$5 \le T_{HL} \le 10$
$7 \le T_{LH} \le 15$

The bit-wise is combined to obtain the word specification:

min (5,7)=5 ≤ Tpd ≤ 15 =max (10, 15)

Maximum propagation delays specification for outputs always have an associated load capacitance. If the specified load capacitance is exceeded, the specified propagation may be exceeded.

For **Clock Edge Controlled Register**, the following timing specifications are given:

Input requirements:
Tstp ≥ _____
Thld ≥ _____
Tpwh ≥ _____
Tpwl ≥ _____

Output quarantees
Tpdb ≥ _____
Tpde ≤ _____

where Tpwh, Tpwl are the clock pulse width high and low specifications.

For **combinational circuits**, there is no clock involved. The only relevant specification is the propagation delay. The event in this case is the change in input signal X that cause the change in the output signal Y.

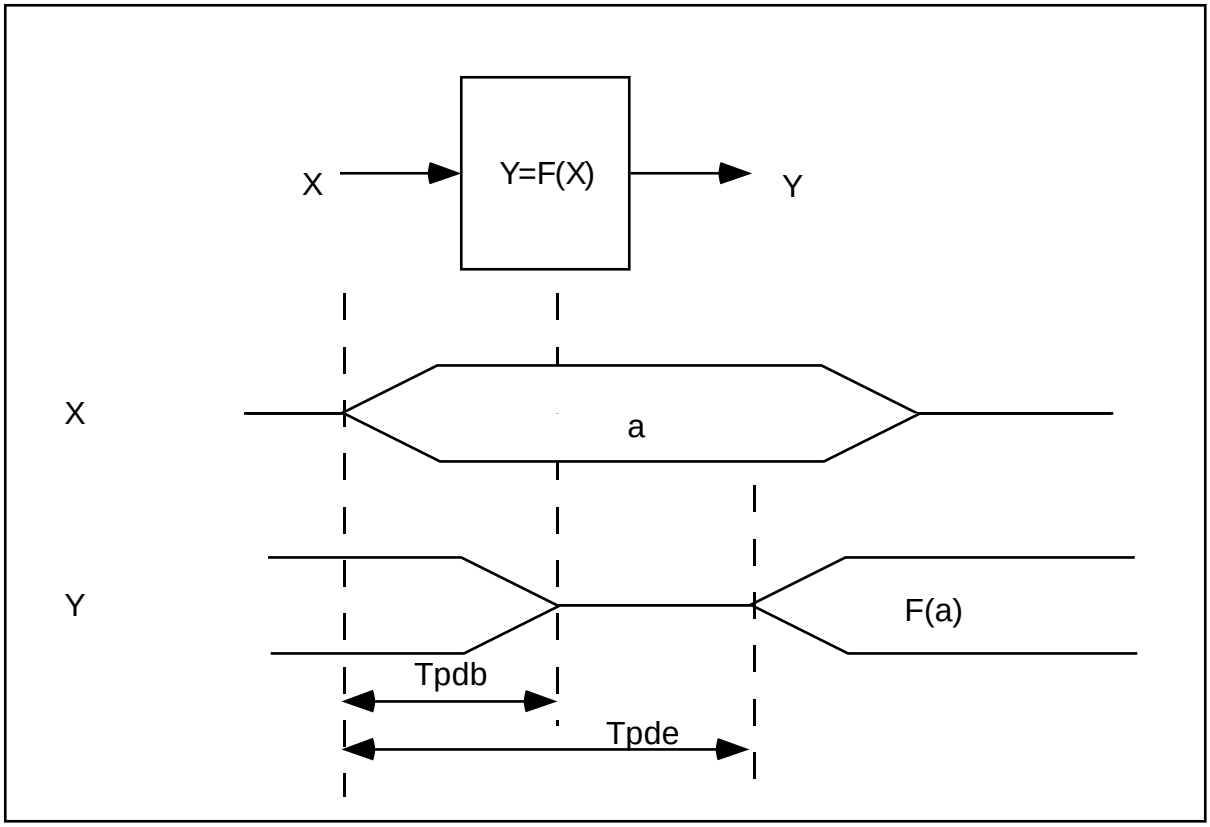



Figure 5. Propagation Delays of Combinatorial Network.

Timing specifications:

$T_{pdb}^{X-Y} \geq$ _____(usually not given)

$T_{pde}^{X-Y} \leq$ _____

For **Level Controlled Register** – the difference between edge control and level control is reflected in the propagation delay specifications. For level control, propagation delay is measured both from the transition of C to its asserted level, and from X<. If X changes while C is asserted, the register will follow the change and load the new value. An edge-controlled register, by contrast, would load only at its transition edge.

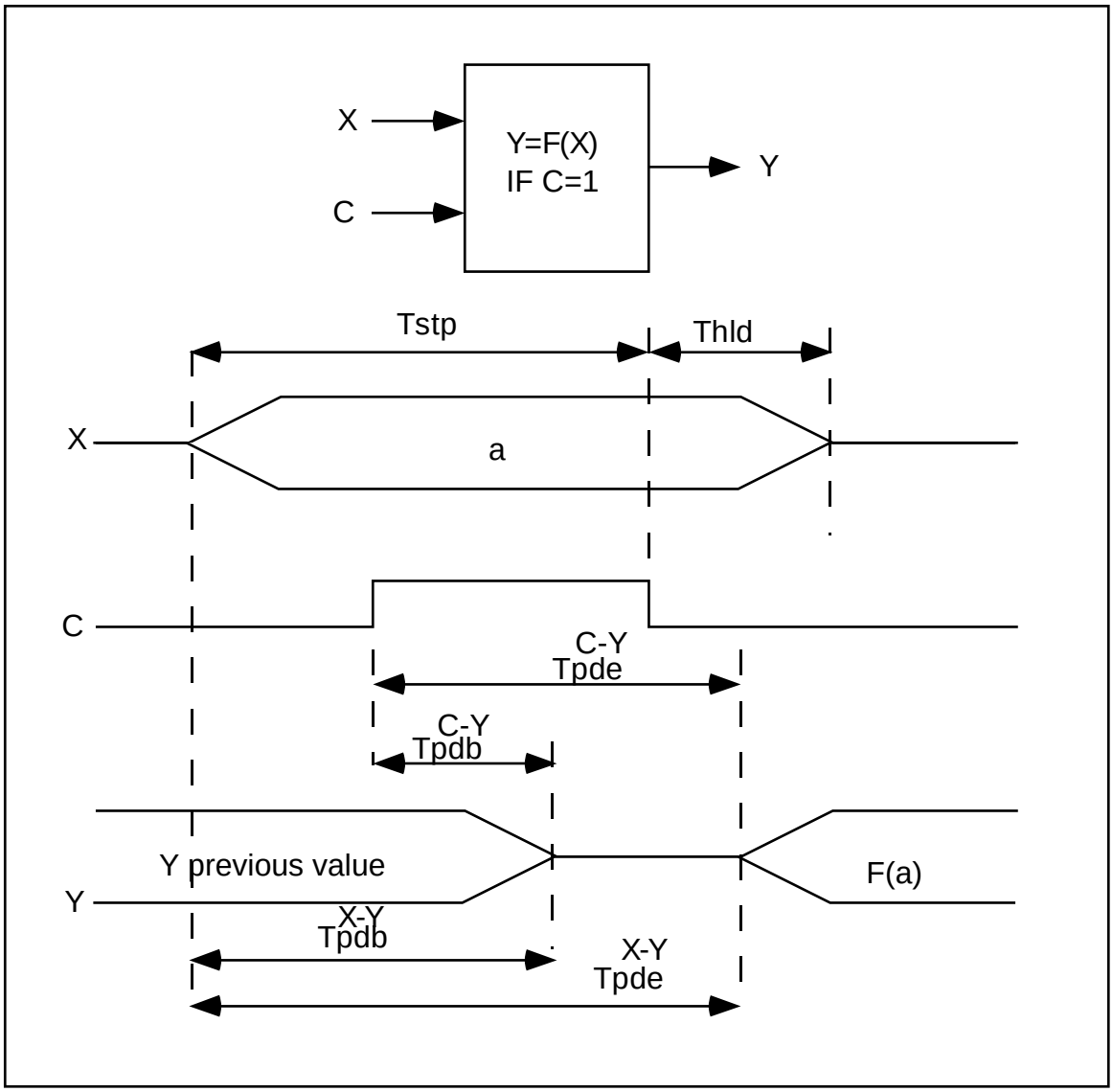



Figure 6. Level Controlled Register Delays.

The timing specifications are:

input requirements:
Tstp ≥ _____
Thld ≥ _____

Output quarantees:

$T_{pde}^{C-Y} \leq$ _____

$T_{pde}^{X-Y} \leq$ _____

$T_{pdb}^{C-Y} \geq$ _____

$T_{pdb}^{X-Y} \geq$ _____

## RAM Modelling

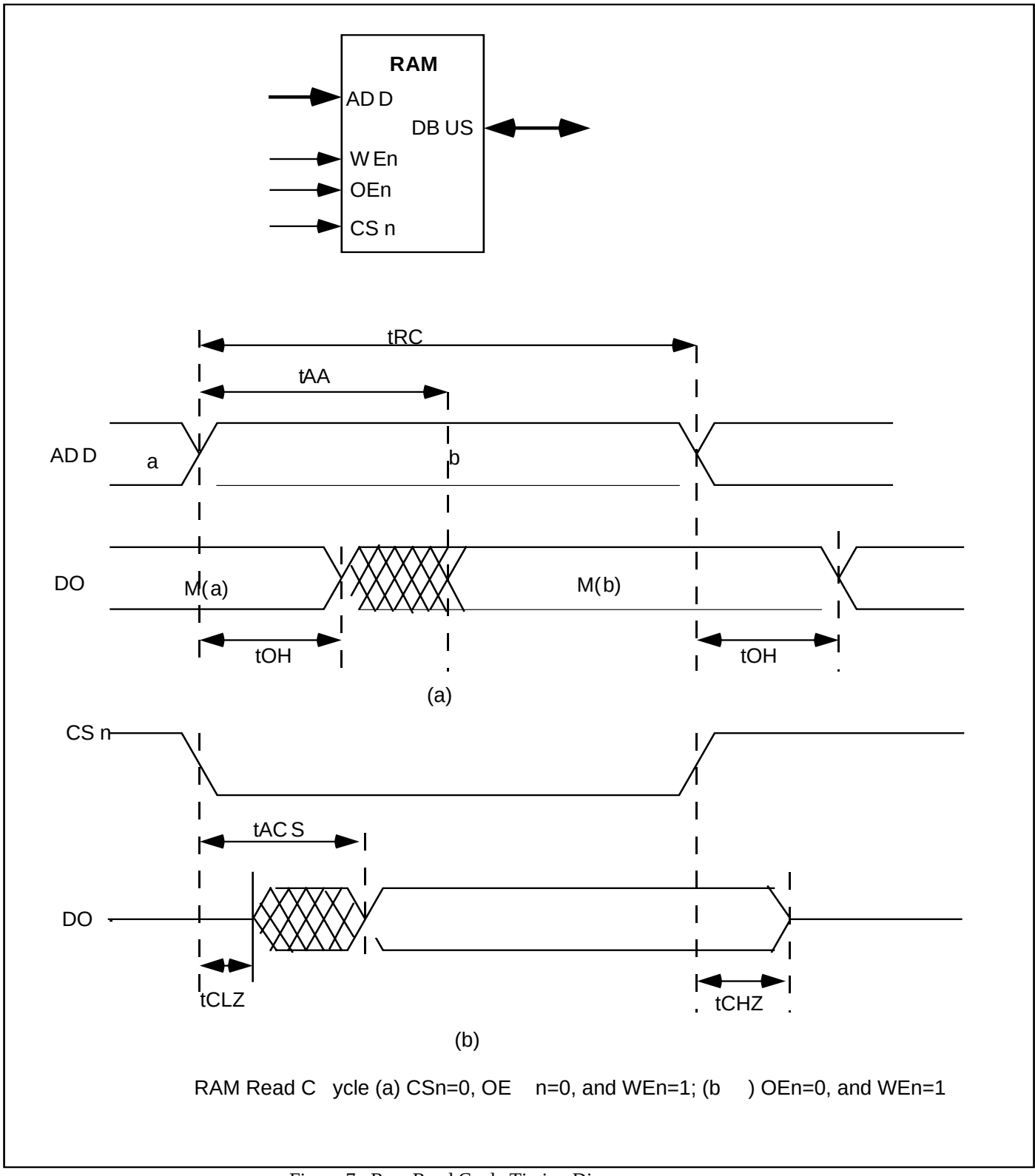



Figure 7. Ram Read Cycle Timing Diagram.

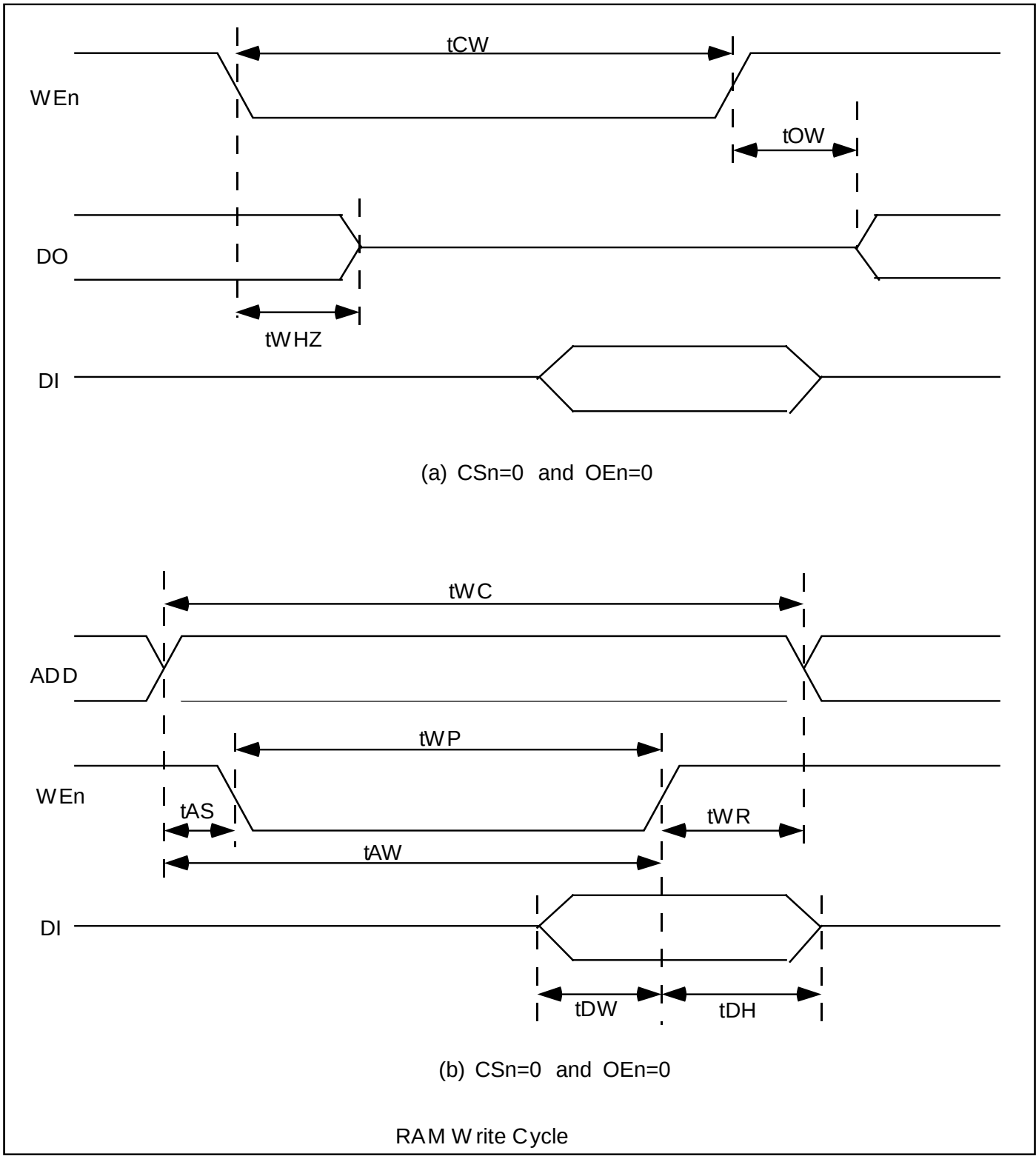



Figure 8. Ram Write Cycle Timing Diagram.

DO = DBUS in output mode.
DI = DBUS in input mode.

tRC – Read cycle time= [ADD|, ADD|]=120ns min .
tAA – Address access time= [ADD<, DO<]=120 ns max
tACS – Chip select access time= [ CS↓, DO<]=120 ns max
tCLZ – Chip selection to output in low-Z =[CS↓, DO|]=10 ns min (beginning of tACS)
tCHZ – Chip deselection to output in high-Z =[CS↑, DO>]= 40 ns max
tOE – Output enable to output valid = [OE↓, DO<] = 80 ns max
tOLZ – Output enable to output in low-Z= [OE↓, DO>]= 10 ns min
tOH – Output hold from address change = [ADD|, DO>]=10 ns min

tWC – Write cycle time = tRC=[ADD|, ADD|]=120 ns min
tCW – Chip selection to end of write = [CS↓, WE↑] = 70 ns min
tAW – Address valid to end of write = [ADD|, WE↑] = 105 ns min
tAS – Address setup time = [ADD|, WE↓]= 0 ns min
tWP – Write pulse wide = [WE↓, WE↑]= 70 ns min
tWR – Write recovery time =[WE↑, ADD|]= 0 ns min
tWHZ – Write enable to output in high-Z = [WE↓, DO>]= 35 ns max
tDW – Data valid to end of write = [DI<, WE↑]= 35 ns min
tDH – Data hold from end of write =[WE↑, DI>] = 0 ns min
tOW – Output active from end of write =[WE↑, DO<] = 10 ns min

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY RAM IS
   GENERIC(
         CONSTANT tAA:       TIME:=120 ns;
         CONSTANT tACS:      TIME:=120 ns;
         CONSTANT tCLZ:      TIME:=10  ns;
         CONSTANT tCHZ:      TIME:=10  ns;
         CONSTANT tOH:       TIME:=10  ns;
         CONSTANT tWC:       TIME:=120 ns;
         CONSTANT tAW:       TIME:=105 ns;
         CONSTANT tWP:       TIME:=70  ns;
         CONSTANT tWHZ:      TIME:=35  ns;
         CONSTANT tDW:       TIME:=35  ns;
         CONSTANT tDH:       TIME:=0   ns;
         CONSTANT tOW:       TIME:=10  ns);
   PORT(CSn, WEn, OEn:IN STD_LOGIC;
         ADD: IN BIT_VECTOR(7 DOWNTO 0);
         DBUS: INOUT STD_LOGIC_VECTOR(7 DOWNTO 0):="ZZZZZZZZ");
END RAM;

ARCHITECTURE beh OF RAM IS
   TYPE RAMarray IS ARRAY(0 TO 255) OF BIT_VECTOR(7 DOWNTO 0);
   SIGNAL RAMS: RAMarray;
```

```
BEGIN
RAM0: PROCESS(CSn, WEn, ADD)
   BEGIN
        IF (rising_edge(WEn) AND CSn'DELAYED='0') OR                    --write cycle Fig 8. (a)
          (rising_edge(CSn) AND WEn'DELAYED='0') THEN
                RAMS(TO_INTEGER(ADD'DELAYED,0))<=TO_BITVECTOR(DBUS'DELAYED,0);
                DBUS <=TRANSPORT DBUS'DELAYED AFTER tOW; --read back after write
        END IF;

        IF falling_edge(WEn) AND CSn='0' THEN --enter write mode
                DBUS <= TRANSPORT "ZZZZZZZZ" AFTER tWHZ;
        END IF;

        IF CSn'EVENT AND OEn='0' THEN                                   --read cycle Fig 7. (b)
           IF CSn='1' THEN        -- RAM is deselected
                DBUS<= TRANSPORT "ZZZZZZZZ" AFTER tCHZ;
           ELSIF WEn='1' THEN
                DBUS <= "XXXXXXXX AFTER tCLZ;
                DBUS <= TRANSPORT To_StdLogicVector(RAMS(TO_INTEGER(ADD,0))) AFTER tACS;
           END IF;
        END IF;

        IF ADD'EVENT AND CSn='0' AND OEn='0' AND WEn='1' THEN  --read cycle Fig 7. (a)
           DBUS<="XXXXXXXX" AFTER tOH;
           DBUS<=TRANSPORT To_StdLogicVector(RAMS(TO_INTEGER(ADD,0))) AFTER tAA;
        END IF;

   END PROCESS RAM0 ;

CHK: PROCESS(WEn, ADD, CSn)
   BEGIN
        IF CSn'DELAYED='0' AND NOW /=0 ns THEN      --Fig 8. (b)
           IF ADD'EVENT THEN
                ASSERT (ADD'DELAYED'STABLE(tWC))  --tRC=tWC assumed
                  REPORT "Address cycle time too short"
                  SEVERITY WARNING;
           END IF;

           IF rising_edge(WEn) THEN
                ASSERT (ADD'DELAYED'STABLE(tAW))
                  REPORT "Address not valid long enough to end of write"
                  SEVERITY WARNING;
                ASSERT (WEn'DELAYED'STABLE(tWP))
                  REPORT "Write pulse too short"
                  SEVERITY WARNING;
                ASSERT (DBUS'DELAYED'STABLE(tDW))
                  REPORT "Data setup time too short"
                  SEVERITY WARNING;
                WAIT FOR tDH;
                ASSERT (DBUS'LAST_EVENT >= tDH)
                  REPORT "Data hold time too short"
                  SEVERITY WARNING;
            END IF;
        END IF;
   END PROCESS CHK;
END beh;
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY ram_test IS
END ram_test;


ARCHITECTURE test OF ram_test IS
   COMPONENT RAM IS
         PORT(CSn, WEn, OEn: IN STD_LOGIC;
            ADD: IN BIT_VECTOR(7 DOWNTO 0);
            DBUS: INOUT STD_LOGIC_VECTOR(7 DOWNTO 0));
   END COMPONENT test;

   FOR U0: RAM USE ENTITY WORK.RAM(beh);

   SIGNAL CSn, WEn: STD_LOGIC:='1';
   SIGNAL DBUS: STD_LOGIC_VECTOR(7 DOWNTO 0):="ZZZZZZZZ";
   SIGNAL ADD: BIT_VECTOR(7 DOWNTO 0);
BEGIN
   U0: RAM PORT MAP(CSn, WEn, '0', ADD, DBUS);
   PROCESS
   BEGIN
         CSn <= ‘1’,  ‘0’ AFTER 100 ns, ‘1’ AFTER 300 ns, ‘0’ AFTER 500ns, ‘1’ AFTER 900 ns ;
         WEn <= ‘1’, ‘0’ AFTER 120 ns, ‘1’ AFTER 280 ns;
         ADD <= “00000000”, “00001000” AFTER 100 ns, “00011000” AFTER 300 ns,
                  “00001000” AFTER 500 ns, “00010000” AFTER 700 ns, “00011000” AFTER 900 ns;
         DBUS <= “11100011” AFTER 240 ns;
   END PROCESS;
END test;
```

| **TIME( ns)** | **CSn** | **WEn** | **ADD** | **DI** | **DO** | **Remark** |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 00000000 | | ZZZZZZZZ | |
| 100 | 0 | 1 | 00001000 | | ZZZZZZZZ | ADD change, read |
| 110* | 0 | 1 | 00001000 | | XXXXXXXX | X => DBUS tOH=10ns later |
| 120 | 0 | 0 | 00001000 | | XXXXXXXX | Switch to write |
| 155* | 0 | 0 | 00001000 | | ZZZZZZZZ | Z => DBUS tWHZ=35ns later |
| 240 | 0 | 0 | 00001000 | 11100011 | | Data to be written on DBUS |
| 280 | 0 | 1 | 00001000 | 11100011 | | tDH=35<280-240, setup time satisfied |
| 290* | 0 | 1 | 00001000 | | 11100011 | Data written read back tOW=10ns later |
| 300 | 1 | 1 | 00011000 | | 11100011 | ADD changeand Ram deselected |
| 310* | 1 | 1 | 00011000 | | ZZZZZZZZ | Ram deselected Z => DBUS tCHZ=10ns later |
| 500 | 0 | 1 | 00001000 | | | Ram selected, ADD set up to read data written eralier |

| | | | | | | |
|---|---|---|---|---|---|---|
| 510* | 0 | 1 | 00001000 | | XXXXXXXX | X => DBUS<br>tOH=10ns after ADD change |
| 620* | 0 | 1 | 00001000 | | 11100011 | read Data=>DBUS<br>tAA=120ns later |
| 700 | 0 | 1 | 00010000 | | 11100011 | ADD change |
| 710* | 0 | 1 | 00010000 | | XXXXXXXX | X=>DBUS<br>tOH=10ns later |
| 820 | 0 | 1 | 00010000 | | 00000000 | read Data=>DBUS<br>tAA=120ns, assume zero |
| 900 | 1 | 1 | 00011000 | | 00000000 | ADD change, & Ram deselected |
| 910* | 1 | 1 | 00011000 | | ZZZZZZZZ | Z=>DBUS<br>tCHZ=10ns after deselected |