



A proposed test plan along with expected outputs for a 16 bit rising edge triggered register with increment and having tristate control of the register output: The register has a synchronous reset, load has priority over inc.

- 1) Drive reset low (active) to reset the register, drive Datain with a value (ABCD)
This should remain low for several clock cycles.
Dataout should remain undefined as Enable is low (inactive).
- 2) Drive reset high (inactive), no change in output.
- 3) Drive enable high (active)
The output should show 0000 after the next rising edge of the clock as the register was reset and no Load has yet taken place.
- 4) Drive enable low, output should show as undefined (technically it's in high impedance state but alliance show it as undefined)
- 5) Drive Load high
no change in output as enable is low, register should now contain value on Datain
- 6) Drive Load low then drive Enable high,
the value that was on Datain when Load was active should now appear on Dataout while Enable is active.
- 7) Drive Enable low.
- 8) Drive Datain with a new value (1234), this should have no effect until the next active Load.
- 9) Drive Inc high (active), after a rising edge drive Inc low else value will increment on each rising edge
- 10) Drive Enable high, Dataout should show ABCE, (last loaded value + 1)
- 11) Repeat Load → Enable → Inc → Enable with the new Datain value

Once we have designed our test plan we need to write the pattern file: A partial pattern file is shown here , up to just before the first increment test.

```
in vdd B;
in vss B;
in reset B;
in clk B;
in inc B;
in load B;
in data_in(15 downto 0) X;
in enable B;
out data_out(15 downto 0) X;

begin
<0 ns> rstl : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<0 ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<0 ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<0 ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<+5ns> rsth : 10 1 0 0 0 ABCD 0 ****;
<+5ns>      : 10 1 0 0 0 ABCD 0 ****;
<+5ns>      : 10 1 1 0 0 ABCD 0 ****;
<+5ns>      : 10 1 1 0 0 ABCD 0 ****;
<+5ns> enh  : 10 1 0 0 0 ABCD 1 ****;
<+5ns>      : 10 1 0 0 0 ABCD 1 ****;
<+5ns>      : 10 1 1 0 0 ABCD 1 ****;
<+5ns>      : 10 1 1 0 0 ABCD 1 ****;
<+5ns>      : 10 1 0 0 0 ABCD 1 ****;
<+5ns>      : 10 1 0 0 0 ABCD 1 ****;
<+5ns>      : 10 1 1 0 0 ABCD 1 ****;
<+5ns>      : 10 1 1 0 0 ABCD 1 ****;
<+5ns>      : 10 1 0 0 0 ABCD 1 ****;
<+5ns>      : 10 1 0 0 0 ABCD 1 ****;
<+5ns> enl  : 10 1 1 0 0 ABCD 0 ****;
<+5ns>      : 10 1 0 0 0 ABCD 0 ****;
<+5ns>      : 10 1 0 0 0 ABCD 0 ****;
<+5ns>      : 10 1 1 0 0 ABCD 0 ****;
<+5ns>      : 10 1 1 0 0 ABCD 0 ****;
<+5ns> ldh  : 10 1 0 0 1 ABCD 0 ****;
<+5ns>      : 10 1 0 0 1 ABCD 0 ****;
<+5ns>      : 10 1 1 0 1 ABCD 0 ****;
<+5ns>      : 10 1 1 0 1 ABCD 0 ****;
<+5ns>      : 10 1 0 0 1 ABCD 0 ****;
<+5ns>      : 10 1 0 0 1 ABCD 0 ****;
<+5ns>      : 10 1 1 0 1 ABCD 0 ****;
<+5ns>      : 10 1 1 0 1 ABCD 0 ****;
<+5ns> ldl  : 10 1 1 0 0 ABCD 0 ****;
end;
```

Pattern files can be generated through the use of a C programme, an example and further details can be found on the module webpage. Once the initial setup is done then it becomes mostly copy/paste/edit

```

#include <stdio.h>
#include "genpat.h"
char *inttostr(entier)
    int entier;
{
    char *str;
    str = (char *) mblock(32 * sizeof(char));
    sprintf(str, "%d", entier);
    return(str);
}

main ()
{
    int vect_date = 0; /* this date is an absolute date, in ps */
    int i,j,k; /* loop counter variables */

    DEF_GENPAT("ld_inc_reg_in");
    SETTUNIT("ns");

    /* set up the interface declarations */
    /* name, white_space, format, mode , size, option */

    DECLAR("vdd",":0","B",IN,"","");
    DECLAR("vss",":2","B",IN,"","");
    DECLAR("reset",":0","B",IN,"","");
    DECLAR("clk",":2","B",IN,"","");
    DECLAR("inc",":1","B",IN,"","");
    DECLAR("load",":1","B",IN,"","");
    DECLAR("data_in",":2","X",IN,"15 downto 0","");
    DECLAR("enable",":2","B",IN,"","");
    DECLAR("data_out",":2","X",OUT,"15 downto 0","");
    /* insert a label in the pattern file */

    LABEL("rstl");

    /* AFFECT( date, name, value) */
    /* set up the initial values for the inputs and outputs. */

    AFFECT("0", "vdd", "0b1");
    AFFECT("0", "vss", "0b0");
    AFFECT("0", "reset", "0b0"); /*power-on reset assumes active low logic*/
    AFFECT("0", "clk", "0b0");
    AFFECT("0", "inc", "0b0");
    AFFECT("0", "load", "0b0");
    AFFECT("0", "data_in", "0xabcd");
    AFFECT("0", "enable", "0b0");
    AFFECT("0", "data_out", "0x****");

    /*****
    *
    * Allow everything to settle on power up
    * twice round the loop per clock cycle
    *****/
    for (i = 0; i < 10; i++) /*five clock cycles */
    {
        vect_date += 10;
        /* %2 toggle clock on even/odd value of i */
        if(i%2)
            AFFECT(inttostr(vect_date), "clk", "0b0");
        else
            AFFECT(inttostr(vect_date), "clk", "0b1");
    }
}

```

```

    }

LABEL("rsth");
AFFECT(inttostr(vect_date), "reset", "0b1"); /* Reset now inactive */
for (i = 0; i < 2; i++) /*one clock cycle*/
{
    vect_date += 10;
    /* %2 toggle clock on even/odd value of i */
    if(i%2)
        AFFECT(inttostr(vect_date), "clk", "0b0");
    else
        AFFECT(inttostr(vect_date), "clk", "0b1");
}

LABEL("enh");
AFFECT(inttostr(vect_date), "enable", "0b1"); /* enable now active */
for (i = 0; i < 6; i++) /*three clock cycles*/
{
    vect_date += 10;
    /* %2 toggle clock on even/odd value of i */
    if(i%2)
        AFFECT(inttostr(vect_date), "clk", "0b0");
    else
        AFFECT(inttostr(vect_date), "clk", "0b1");
}

LABEL("enl");
AFFECT(inttostr(vect_date), "enable", "0b0"); /* enable now inactive */
for (i = 0; i < 2; i++) /*one clock cycle*/
{
    vect_date += 10;
    /* %2 toggle clock on even/odd value of i */
    if(i%2)
        AFFECT(inttostr(vect_date), "clk", "0b0");
    else
        AFFECT(inttostr(vect_date), "clk", "0b1");
}

LABEL("ldh");
AFFECT(inttostr(vect_date), "load", "0b1"); /* load now active */
for (i = 0; i < 6; i++) /*three clock cycles*/
{
    vect_date += 10;
    /* %2 toggle clock on even/odd value of i */
    if(i%2)
        AFFECT(inttostr(vect_date), "clk", "0b0");
    else
        AFFECT(inttostr(vect_date), "clk", "0b1");
}

LABEL("ldl");
AFFECT(inttostr(vect_date), "load", "0b0"); /* load now inactive */
for (i = 0; i < 2; i++) /*one clock cycle*/
{
    vect_date += 10;
    /* %2 toggle clock on even/odd value of i */
    if(i%2)
        AFFECT(inttostr(vect_date), "clk", "0b0");
    else
        AFFECT(inttostr(vect_date), "clk", "0b1");
}

LABEL("enh");
AFFECT(inttostr(vect_date), "enable", "0b1"); /* enable now active */
for (i = 0; i < 6; i++) /*three clock cycles*/
{
    vect_date += 10;
    /* %2 toggle clock on even/odd value of i */

```

```

        if(i%2)
            AFFECT(inttostr(vect_date), "clk", "0b0");
        else
            AFFECT(inttostr(vect_date), "clk", "0b1");
    }
LABEL("enl");
AFFECT(inttostr(vect_date), "enable", "0b0"); /* enable now inactive */
for (i = 0; i < 2; i++) /*one clock cycle*/
{
    vect_date += 10;
    /* %2 toggle clock on even/odd value of i */
    if(i%2)
        AFFECT(inttostr(vect_date), "clk", "0b0");
    else
        AFFECT(inttostr(vect_date), "clk", "0b1");
}

LABEL("inch");
AFFECT(inttostr(vect_date), "inc", "0b1"); /* inc now active */
for (i = 0; i < 2; i++) /*one clock cycles*/
{
    vect_date += 10;
    /* %2 toggle clock on even/odd value of i */
    if(i%2)
        AFFECT(inttostr(vect_date), "clk", "0b0");
    else
        AFFECT(inttostr(vect_date), "clk", "0b1");
}
LABEL("incl");
AFFECT(inttostr(vect_date), "inc", "0b0"); /* inc now inactive */
for (i = 0; i < 2; i++) /*one clock cycle*/
{
    vect_date += 10;
    /* %2 toggle clock on even/odd value of i */
    if(i%2)
        AFFECT(inttostr(vect_date), "clk", "0b0");
    else
        AFFECT(inttostr(vect_date), "clk", "0b1");
}

LABEL("enh");
AFFECT(inttostr(vect_date), "enable", "0b1"); /* enable now active */
for (i = 0; i < 6; i++) /*three clock cycles*/
{
    vect_date += 10;
    /* %2 toggle clock on even/odd value of i */
    if(i%2)
        AFFECT(inttostr(vect_date), "clk", "0b0");
    else
        AFFECT(inttostr(vect_date), "clk", "0b1");
}
LABEL("enl");
AFFECT(inttostr(vect_date), "enable", "0b0"); /* enable now inactive */
for (i = 0; i < 2; i++) /*one clock cycle*/
{
    vect_date += 10;
    /* %2 toggle clock on even/odd value of i */
    if(i%2)
        AFFECT(inttostr(vect_date), "clk", "0b0");
    else
        AFFECT(inttostr(vect_date), "clk", "0b1");
}

SAV_GENPAT();
}

```

output from genpat (man genpat for details)

```
in vdd B;
in vss B;
in reset B;
in clk B;
in inc B;
in load B;
in data_in(15 downto 0) X;
in enable B;
out data_out(15 downto 0) X;

begin
<0 ns> rstl : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<0 ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<0 ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 0 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<+5ns>      : 10 0 1 0 0 ABCD 0 ****;
<+5ns> rsth : 10 1 0 0 0 ABCD 0 ****;
<+5ns>      : 10 1 0 0 0 ABCD 0 ****;
<+5ns>      : 10 1 1 0 0 ABCD 0 ****;
<+5ns>      : 10 1 1 0 0 ABCD 0 ****;
<+5ns> enh  : 10 1 0 0 0 ABCD 1 ****;
<+5ns>      : 10 1 0 0 0 ABCD 1 ****;
<+5ns>      : 10 1 1 0 0 ABCD 1 ****;
<+5ns>      : 10 1 1 0 0 ABCD 1 ****;
<+5ns>      : 10 1 0 0 0 ABCD 1 ****;
<+5ns>      : 10 1 0 0 0 ABCD 1 ****;
<+5ns>      : 10 1 1 0 0 ABCD 1 ****;
<+5ns>      : 10 1 1 0 0 ABCD 1 ****;
<+5ns>      : 10 1 0 0 0 ABCD 1 ****;
<+5ns>      : 10 1 0 0 0 ABCD 1 ****;
<+5ns> enl  : 10 1 1 0 0 ABCD 0 ****;
<+5ns>      : 10 1 0 0 0 ABCD 0 ****;
<+5ns>      : 10 1 0 0 0 ABCD 0 ****;
<+5ns>      : 10 1 1 0 0 ABCD 0 ****;
<+5ns>      : 10 1 1 0 0 ABCD 0 ****;
<+5ns> ldh  : 10 1 0 0 1 ABCD 0 ****;
<+5ns>      : 10 1 0 0 1 ABCD 0 ****;
<+5ns>      : 10 1 1 0 1 ABCD 0 ****;
<+5ns>      : 10 1 1 0 1 ABCD 0 ****;
<+5ns>      : 10 1 0 0 1 ABCD 0 ****;
<+5ns>      : 10 1 0 0 1 ABCD 0 ****;
<+5ns>      : 10 1 1 0 1 ABCD 0 ****;
<+5ns>      : 10 1 1 0 1 ABCD 0 ****;
<+5ns> ldl  : 10 1 1 0 0 ABCD 0 ****;
end;
```