Minimizing Single Event Upset Effects Using Synplicity

This application note gives an overview of some single event upset (SEU) resistant design techniques and describes how to implement these techniques using Synplicity Synplify 3.0C or later. Familiarity with Synplify is assumed. For additional information about radiation resistant design techniques, refer to the Actel application note *Design Techniques for RadHard FPGAs* available on the Actel Web site (http://www.actel.com).

Background and Terminology

The Actel ACT 2, ACT 3, 3200DX, and 42MX device families contain two types of logic modules, sequential modules (S-modules) and combinatorial modules (C-modules). S-modules contain a sequential logic element and some combinatorial logic. C-modules contain only combinatorial logic. Flip-flops and latches can be implemented using the sequential logic element of an S-module (S-FF) or by using combinatorial logic (CC-FF). The combinatorial logic in a CC-FF may come from the combinatorial portion of an S-module, from a C-module, or from both.

The 54SX device family also contains two types of logic modules, register cells (R-cells) and combinatorial cells (C-cells). R-cells contain only a sequential logic element. C-cells contain only combinatorial logic. Flip-flops and latches currently can only be implemented using the sequential logic element of an R-cell, also referred to as an S-FF in this application note.

The ACT 1 and 40MX device families contain only C-modules. All flip-flops and latches are implemented as CC-FFs.

SEU Resistant Design Techniques

The resistance of a device to SEU effects can be influenced by using certain logic design techniques. The default technique, using S-FFs, produces designs that are the most susceptible to SEU effects. Because ACT 1 and 40MX devices do not have S-modules, S-FFs cannot be implemented in these devices.

In addition to the default there are two SEU resistant design techniques that can be used in Actel devices using Synplicity. In order of increasing resistance to SEU effects the techniques are: using CC-FFs and using triple voting. Synplicity also allows for custom implementations. A single design may use any or all of these design techniques.

Using CC-FFs

Using CC-FFs produces designs that are more resistant to SEU effects than designs using S-FFs. CC-FFs are used by default in ACT 1 and 40MX devices because these devices only contain CC-FFs. CC-FFs cannot currently be implemented in 54SX devices. CC-FFs typically use twice the area resources of S-FFs.

Using Triple Voting

Triple voting, also called triple module redundancy (TMR), produces designs that are the most resistant to SEU effects. Instead of a single flip-flop, triple voting uses three flip-flops leading to a majority gate voting circuit. This way, if one flip-flop is accidentally flipped to the wrong state, the other two out-vote it and the correct value is propagated to the rest of the circuit. Because of the cost (typically three to four times the area and two times the delay required for S-FF implementations), triple voting is usually implemented using S-FFs. However, triple voting can be implemented using only CC-FFs in Synplicity.

Implementing a Technique

Synplify has eight sequential primitives that are used to implement latches and flip-flops. The basic latch and flip-flop each have a variation with set, reset or both. The primitives are shown in Table 1.

Table 1 • *Synplicity Sequential Primitives*

Flip-Flop	Latch
dff	lat
dffr	latr
dffs	lats
dffrs	latrs

Synplicity provides supplemental library files that contain compiler directives to ensure that the intended sequential mapping is implemented. The files, available in both Verilog and VHDL, are as follows:

- cc.v(hd)—for CC-FF implementation
- tmr.v(hd)—for TMR implementation without CC-FFs
- tmr_cc.v(hd)—for TMR implementation using CC-FFs

Although the procedure to implement one of these techniques is simple, the order in which the source files are added is critical. The Actel family library file must appear first, then supplemental library file, then the design file(s). Use the following procedure to implement a technique.

1. Invoke Synplify and open a project or start a new project.
2. Click the Change Target button and select the Actel family, device, and speed grade, then click OK.
3. Click the Add button to add source file(s) to the project. If there are multiple design files, make sure the top level file appears at the bottom of the source files list.
4. Click the Add button to add one of the supplemental library files (cc, tmr, or tmr_cc). The files are located in the "<synplify_install_directory>\lib\actel" directory. Make sure to use the one with the correct extension (.v or .vhd).
5. Click the Add button to add the appropriate Actel family to the top of the source files list. This file is also located in "<synplify_install_directory>\lib\actel" directory.
6. Click the Run button to compile the design.

You can make sure that your sequential logic has been implemented correctly by selecting the Technology View command from the HDL Analyst menu. The Actel *Macro Library Guide* can be used to determine whether each macro used contains S-modules (or R-cells) and/or C-modules (C-cells).

Figure 1 illustrates how Synplify implements a design using the "tmr_cc.vhd" supplemental library file during compilation. S-FFs have been replaced with TMR implementations comprised of only CC-FFs. Notice the use of the "DFP1" macro, which contains only C-modules.

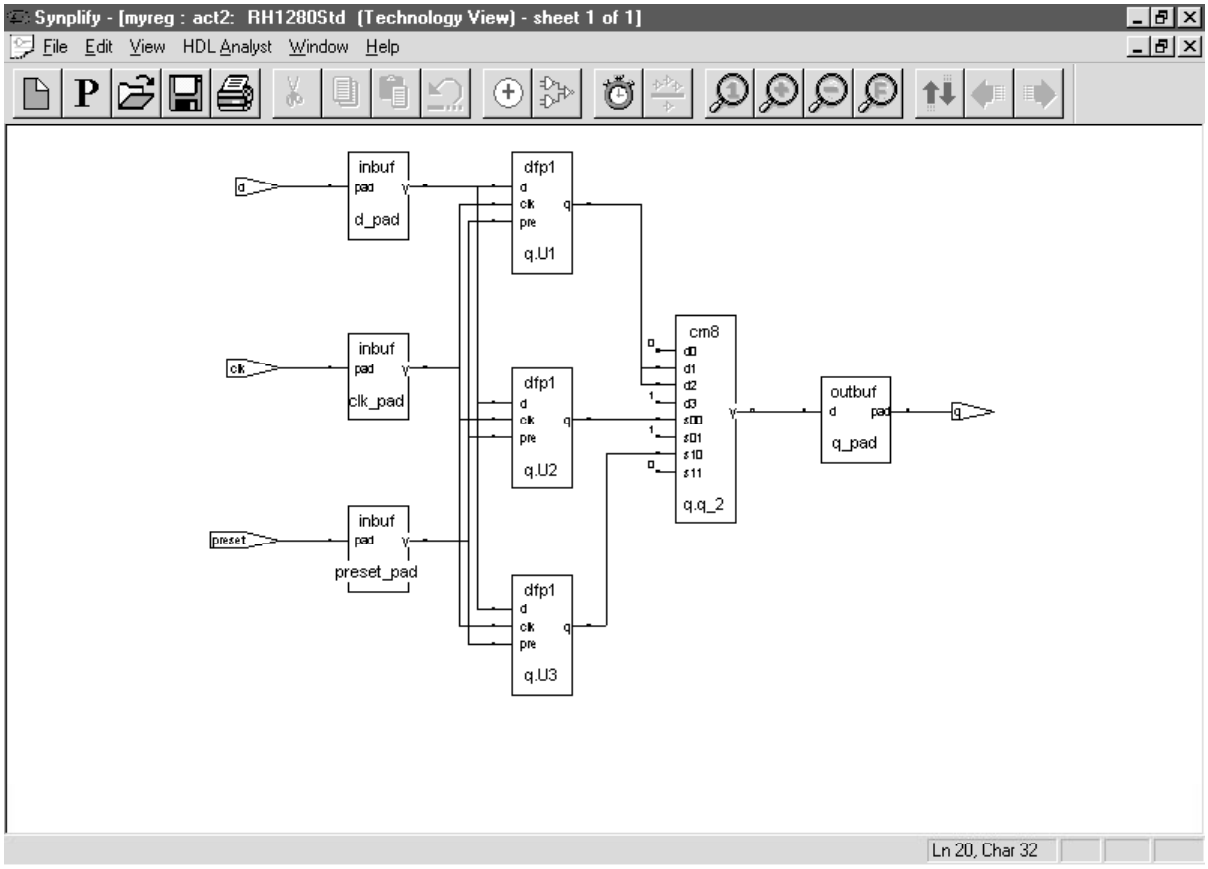

Figure 1 • *Synplify Implementation Using the "tmr.vhd" Supplemental Library File*

Hierarchy

The procedure in the previous section implies that only one design technique can be employed for the entire design. However, it is possible to apply separate techniques to sections of the design. This is achieved through black boxing. For example, if you have a design, A, that has hierarchical blocks B and C (Figure 2) and you wanted to implement CC-FFs in A, TMR in B, and TMR using CC-FFs in C, the procedure would be as follows:

1. Using a text editor, write blocks B and C into separate design files.
2. Synthesize B and C using the appropriate supplemental library files as described in the previous section. The result files will be EDIF (.edn). Remember to click the disable I/O insertion box in the Set Device Options dialog box, so that I/O cells are not inserted into the sub-blocks.
3. Instantiate B and C as black boxes in block A. If you don't already know how to do this, there is a good description in the Synplify on-line help menu - search for "instantiating black boxes." Synthesize A with the "cc.v(hd)" supplemental library file.
4. Run the Actel "edn2adl" utility. For example, type the following command at the DOS or UNIX prompt:

   ```
   edn2adl fam:act2 ednin:B.edn+C.edn+A.edn
   adl:A.adl A
   ```

   "A" is the top level entity or module name in the command line above. The top level EDIF file is listed last in the ednin string.

You can then compile the new design in the Actel Designer software. Make sure you select the output file from step 4 above (A.adl) when compiling the new design.

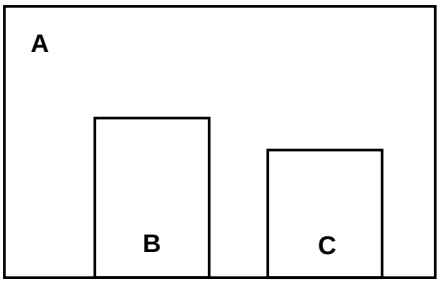

***Figure 2** • Hierarchical Block Diagram*

Customization

The supplemental library files provided by Synplicity use compiler directives to perform the macro substitution and implement the design techniques. The directives, "syn_implement" and "syn_preserve," are used to map the primitive functions to the desired hardware implementation and ensure that the mapping is not overridden by further optimization. For example, in Figure 3, a section of "cc.vhd," the bold lines make sure that the CC-FF macro "DFP1" is actually used at the gate level. Figure 4 illustrates the Verilog versions of the directives.

To generate a custom implementation of the techniques, copy one of the supplemental library files to your local area and modify it for the desired implementation. Then, include this new supplemental library file in the Synplify source file list.

```
entity dfp1_dff is
port (q : out std_logic;
      d,c : in std_logic);
end dfp1_dff;

architecture arch1 of dfp1_dff is
 attribute syn_implement of arch1 : architecture is "dff";
 attribute syn_preserve of arch1 : architecture is true;
begin

U1: DFP1 port map (d=>d,clk=>c,pre =>'0',q=>q);

end arch1;
```

***Figure 3** • Substitution Complier Directives (VHDL)*

```
module dfpc_dffrs(q,d,c,s,r)/* synthesis syn_preserve=1
syn_implement=dffrs */;
output q;
input d,c,s,r;

wire r_i = ~r;
DFPC  u1 (d,c,s,r_i,q);

endmodule
```

Figure 4 • *Substitution Compiler Directives (Verilog)*

Additional Information

For additional information about designing radiation resistant devices, visit the following Web site:

http://www.actel.com/products/radhard.html

If you have any comments or suggestions about Actel's line of radiation resistant devices, e-mail Actel at radhard.designer@actel.com. For technical support contact the Actel Customer Application Center at 1-800-262-1060 or tech@actel.com.

Take it to a higher level.

http://www.actel.com

Actel Europe Ltd.
Daneshill House, Lutyens Close
Basingstoke, Hampshire RG24 8AG
United Kingdom
Tel: +44.(0)1256.305600
Fax: +44.(0)1256.355420

Actel Corporation
955 East Arques Avenue
Sunnyvale, California 94086
USA
Tel: 408.739.1010
Fax: 408.739.1540

Actel Japan
EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ka
Tokyo 150 Japan
Tel: +81.(0)3.3445.7671
Fax: +81.(0)3.3445.7668

5192643-0