SIMULAÇÃO DE FALHAS RANDÔMICAS

Ir para o diretório de simulação de falhas randômicas:

```
cd xtea-cripto-core/sim/fault/rtl/random_fault_campaign
```

Olhar o script de simulação random_fault_campaign.sh:

```
cat random_fault_campaign.sh
```

O seguinte comando habilita a inserção de falhas no circuito especificado (neste caso, o módulo de criptografia XTEA):

```
xrun -V93 -fault_work fault_db -fault_overwrite -fault_file fault.lst -l logs/fault_elab.log \
-input run_sim.tcl -top tb ../../../../rtl/xtea.vhd ../../../../sim/xtea_tb.vhd
```

Olhar o arquivo fault.lst:

```
cat fault.lst
```

O comando fault_target tb.core especifica um módulo / célula / hierarquia onde será inserido a(s) falha(s). Os proximos comandos no script "random_fault_campaign.sh" geram os valores de referência para a inserção de falhas:

```
fault_type="SA1"

xrun -R -fault_good_run -fault_type ${fault_type} -fault_tw 7000ns:10000ns -fault_seed
123 -fault_num_nodes 2 \
-fault_work fault_db -input run_good_sim.tcl -l logs/xrun_good.log
```

Neste caso, o database é gerado levando em conta o modelo de falha *stuck-at 1* e dois pontos de falha simultâneos. Olhar o arquivo run_good_sim.tcl:

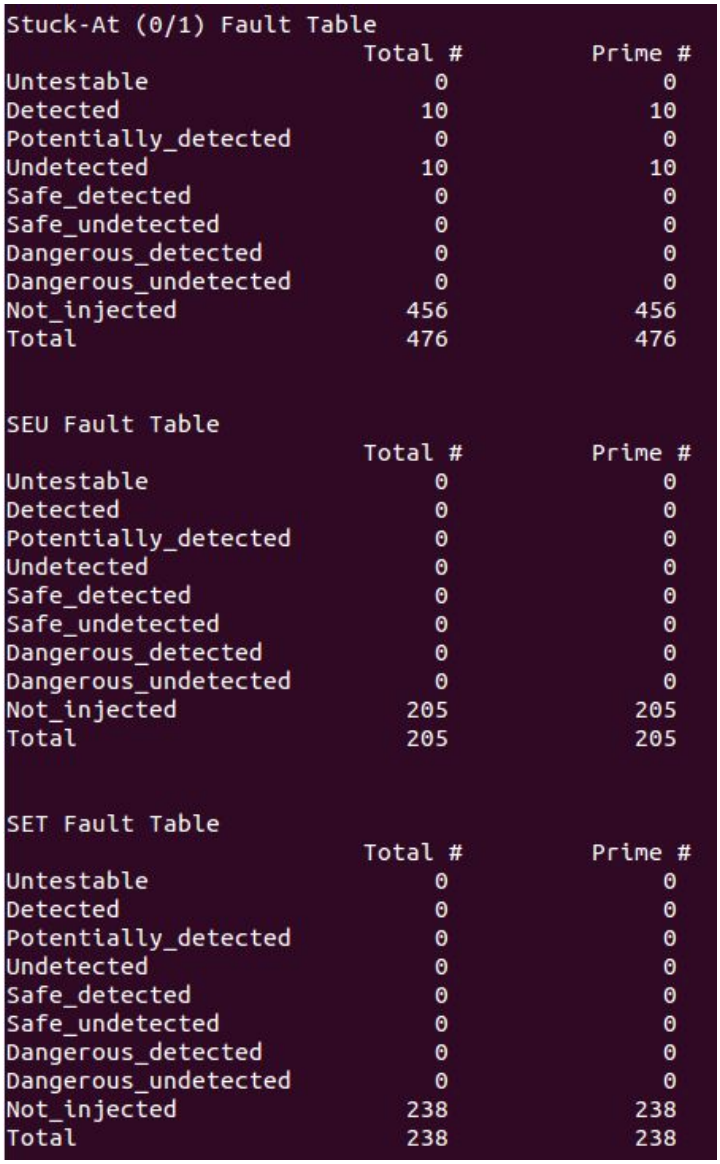
```
cat run_good_sim.tcl
```

O comando fs_strobe tb.core.output cria um ponto de observabilidade. Os próximos comandos inserem falhas do tipo *stuck-at 1* em pontos aleatórios do circuito:

```
n_fault=10

for (( i=1; i<=${n_fault}; i++ ))
do
        xrun -R -input run_fault_sim.tcl -fault_sim_run -fault_timeout 2ms \
        -fault_random_id $i -fault_work fault_db -l logs/xrun_fault_$i.log
done
```

O comando xfr -fault_work fault_db -logfile rpt.log gera o seguinte relatório:



```
Stuck-At (0/1) Fault Table
                        Total #         Prime #
Untestable                    0               0
Detected                     10              10
Potentially_detected          0               0
Undetected                   10              10
Safe_detected                 0               0
Safe_undetected               0               0
Dangerous_detected            0               0
Dangerous_undetected          0               0
Not_injected                456             456
Total                       476             476


SEU Fault Table
                        Total #         Prime #
Untestable                    0               0
Detected                      0               0
Potentially_detected          0               0
Undetected                    0               0
Safe_detected                 0               0
Safe_undetected               0               0
Dangerous_detected            0               0
Dangerous_undetected          0               0
Not_injected                205             205
Total                       205             205


SET Fault Table
                        Total #         Prime #
Untestable                    0               0
Detected                      0               0
Potentially_detected          0               0
Undetected                    0               0
Safe_detected                 0               0
Safe_undetected               0               0
Dangerous_detected            0               0
Dangerous_undetected          0               0
Not_injected                238             238
Total                       238             238
```

O relatório mostra que foram injetadas 10 falhas do tipo *stuck-at 1*, gerando um total de 20 falhas, uma vez que são inseridas em dois nodos simultaneamente. Neste caso, 10 falhas são detectadas, e 10 não são. Para rodar o script de simulação:

```
./random_fault_campaign.sh
```

SIMULAÇÃO DE FALHAS EM NODOS ESPECÍFICOS

Ir para o diretório de simulação de falhas:

```
cd xtea-cripto-core/sim/fault/rtl/targeted_fault_campaign
```

Olhar o script de simulação targeted_fault_campaign.sh:

```
cat targeted_fault_campaign.sh
```

O seguinte comando habilita a inserção de falhas no circuito especificado (neste caso, o módulo de criptografia XTEA):

```
xrun -V93 -fault_work fault_db -fault_overwrite -fault_file fault.lst -l logs/fault_elab.log \
-input run_sim.tcl -top tb ../../../../rtl/xtea.vhd ../../../../sim/xtea_tb.vhd
```

Olhar o arquivo fault.lst:

```
cat fault.lst
```

O comando fault_target tb.core especifica um módulo / célula / hierarquia onde será inserido a(s) falha(s). Os proximos comandos no script "random_fault_campaign.sh" geram os valores de referência para a inserção de falhas:

```
xrun -R -fault_good_run -fault_type ${fault_type} -fault_tw 7000ns:10000ns -fault_seed
123 -fault_num_nodes 1 \
-fault_work fault_db -input run_good_sim.tcl -l logs/xrun_good.log
```

Neste caso, o database é gerado levando em conta todos os modelos de falhas suportados pela ferramenta (*stuck-at 0*, *stuck-at 1*,SEU,SET). Olhar o arquivo run_good_sim.tcl:

```
cat run_good_sim.tcl
```

O comando fs_strobe tb.core.output cria um ponto de observabilidade. Os próximos comandos inserem falhas do tipo *stuck-at 1* em pontos aleatórios do circuito:

```
xrun -R -fault_sim_run -fault_work fault_db \
-input run_fault_sim.tcl -fault_timeout 2ms -l logs/ncsim_fault.log
```

Olhar o arquivo run_fault_sim.tcl:

```
cat run_fault_sim.tcl
```

Neste arquivo estão especificados os tipos de falhas, o tempo, e o sinal em que serão inseridas as falhas. Neste caso, são 5 falhas *stuck-at 0,* 5 do *stuck-at 1,* 10 do tipo SEU e 10 do tipo SET.

O comando xfr -fault_work fault_db -logfile rpt.log gera o seguinte relatório:

```
Stuck-At (0/1) Fault Table
                        Total #         Prime #
Untestable                  0               0
Detected                   10              10
Potentially_detected        0               0
Undetected                  0               0
Safe_detected               0               0
Safe_undetected             0               0
Dangerous_detected          0               0
Dangerous_undetected        0               0
Not_injected              466             466
Total                     476             476


SEU Fault Table
                        Total #         Prime #
Untestable                  0               0
Detected                   10              10
Potentially_detected        0               0
Undetected                  0               0
Safe_detected               0               0
Safe_undetected             0               0
Dangerous_detected          0               0
Dangerous_undetected        0               0
Not_injected              195             195
Total                     205             205


SET Fault Table
                        Total #         Prime #
Untestable                  0               0
Detected                   10              10
Potentially_detected        0               0
Undetected                  0               0
Safe_detected               0               0
Safe_undetected             0               0
Dangerous_detected          0               0
Dangerous_undetected        0               0
Not_injected              228             228
Total                     238             238
```

O relatório mostra que foram injetadas 10 falhas do tipo *stuck-at*(5 do tipo 0 e 5 tipo 1), 10 falhas do tipo SEU e dez falhas do tipo SET, gerando um total de 30 falhas. Neste caso, todas as falhas são detectadas. Para rodar o script de simulação:

```
./targeted_fault_campaign.sh
```

REFERÊNCIAS

[1] /soft64/cadence/ferramentas/XCELIUM183/doc/functionalsafety/Modeling_Faults.html

[2] /soft64/cadence/ferramentas/XCELIUM183/doc/functionalsafety
/Injecting_and_Classifying_Faults.html#pagetop

[3] /soft64/cadence/ferramentas/XCELIUM183/doc/functionalsafety
/Appendix_A__Functional_Safety_Verification_Analysis.html#pagetop

[4] /soft64/cadence/ferramentas/XCELIUM183/doc/functionalsafety
/Reporting_on_Simulation_Results.html#pagetop