DELCPG016 – Tutorial *Genus Desing for Test*

Objetivo: executar o fluxo de síntese lógica com DFT no Genus para inserir *scan cells* e permitir a geração de padrões de teste (ATPG). Para isso, são necessários os arquivos mostrados na Figura 1.

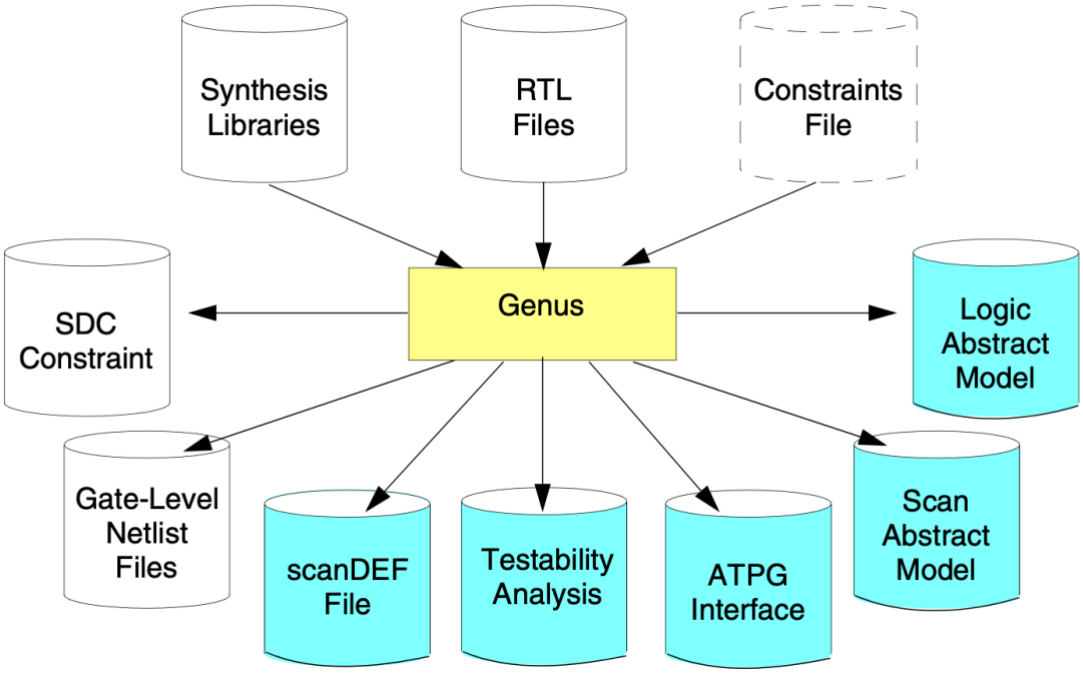

Figura 1

- **Synthesis Libraries**: biblioteca(s) contendo informações de timing e das scan cells
- **RTL Files**: arquivos HDL
- **Constraints File (opcional)**: arquivo com restrições do projeto

A Figura 1 também contém os arquivos gerados:

- **Gate-Level Netlist**
- **Constraint SDC**, para processamento da lógica DFT inserida nesta etapa na ferramenta de P&R
- **scanDEF File**: descrição da configuração da *scan chain*
- **Testability Analysis**
- **ATPG Interface**
- **Scan Abstract Model**
- **Logic Abstract Model**

O fluxo a ser executado está mostrado na Figura 2. Cada uma das etapas será brevemente descrita a seguir.

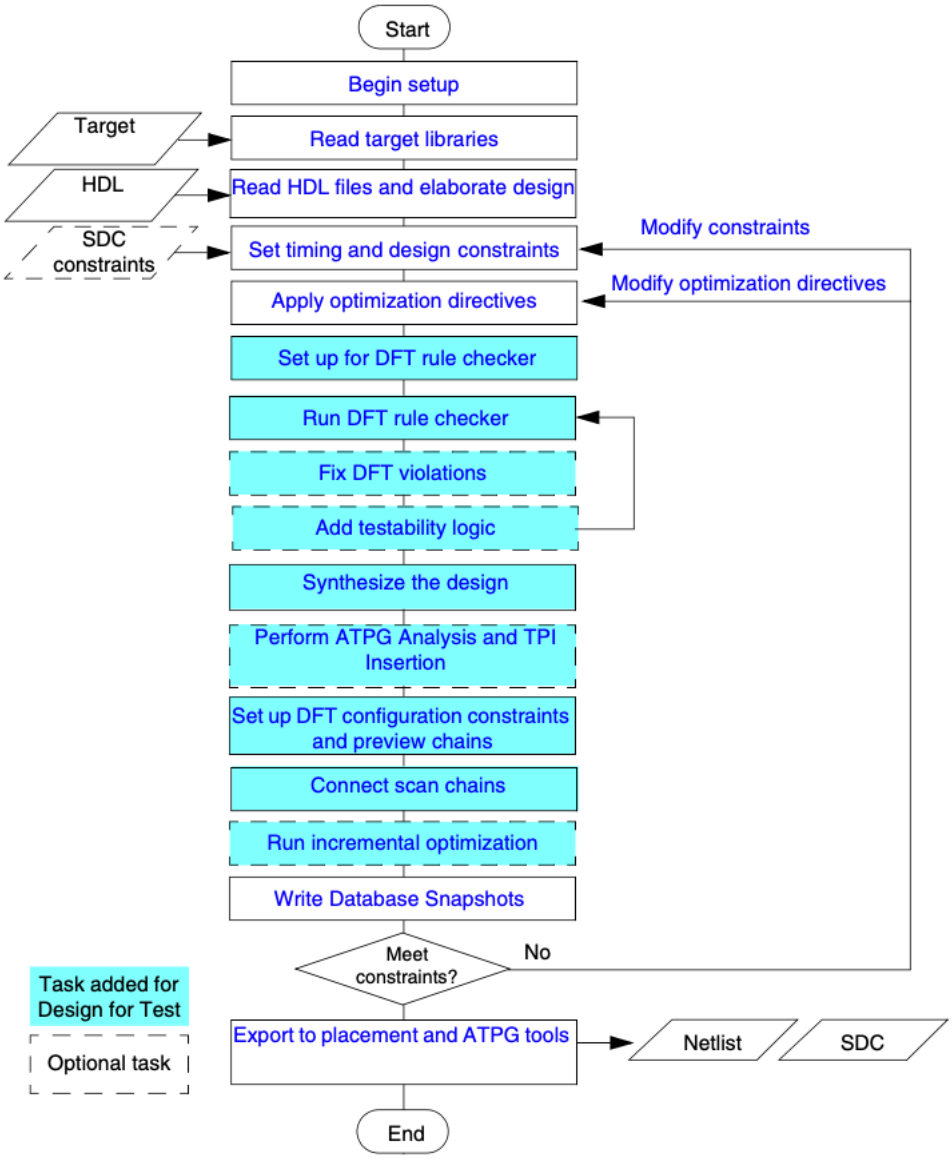

Figura 2

1. ***Begin Setup***: nesta etapa são especificados os caminhos para arquivos de bibliotecas, scripts e arquivos HDL. Se não forem especificados, o Genus utiliza o diretório em que foi aberto.

```
set_db init_lib_search_path ../lib
set_db init_hdl_search_path ../rtl
```

2. ***Read Target Libraries***: especificar a(s) biblioteca(s) que será(ão) usada(s) na síntese.

```
read_libs slow_vdd1v0_basicCells.lib
```

3. ***Read HDL Files and Elaborate Design***

```
read_hdl counter.v
```

Opcional: Para possibilitar o rastreamento de violações de regras DFT (arquivo RTL e número da linha):

```
set_db hdl_track_filename_row_col true
```

Em seguida, transformar a descrição RTL em uma forma adequada para a síntese com o comando `elaborate`.

4. ***Set Timing and Design Constraints***: essas informações podem ser inseridas no *shell* do Genus manualmente ou em um arquivo no formato SDC.

```
read_sdc ../constraints/constraints_top.sdc
```

5. ***Apply Optimization Directives***: várias estratégia de otimização podem ser usadas para atender os requisitos de desempenho após a síntese. Alguns exemplos são: preservar, agrupar ou desagrupar instâncias e módulos, controlar a otimização de instâncias hierárquicas, especificar nets ideais, etc.

6. **Set Up for DFT Rule Checker**:

   Escolha do estilo: o estilo ***muxed_scan*** é o mais comum. Nesse caso, um flip-flop tipo D é substituído por uma célula scan composta por um multiplexador e um flip-flop D. O sinal scan_enable seleciona a origem dos dados (sistema ou scan). O comando a ser usado é:

```
set_db dft_scan_style muxed_scan
```

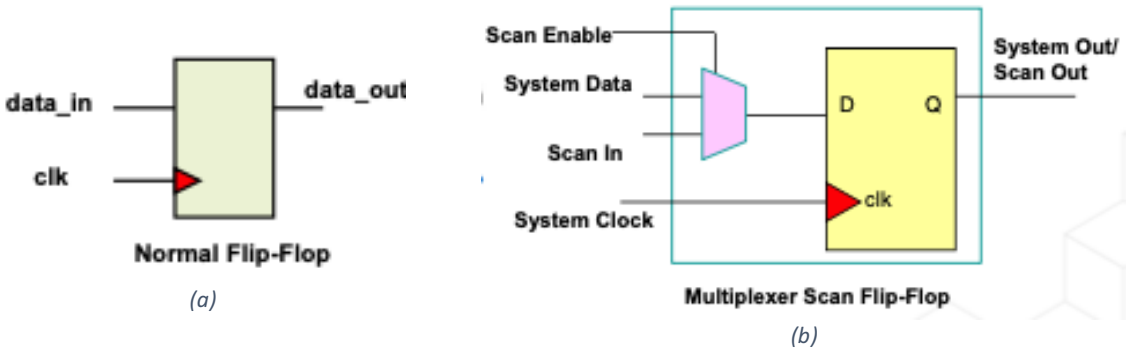

Figura 3 – Flip-flops.(a): Normal; (b) Scan

Opcional: usar prefixo **dft**:

```
set_db dft_prefix dft
```

Opcional: Definição do sinal ***test_mode*** com a opção *-create-port:*

```
define_test_mode -name TM -active high -create_port TM
```

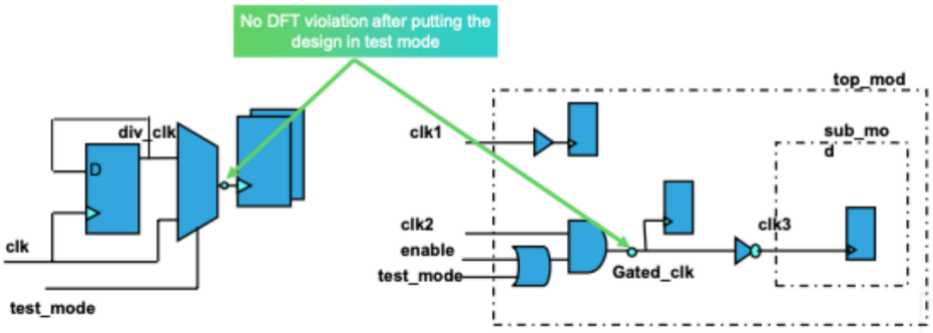

Figura 4 – Definição do sinal test_mode

Definição do sinal ***shift_enable*** com a opção *-create-port:* habilita o modo de deslocamento nos flip-flops scan.

```
define_shift_enable -name SE -active high -create_port SE
```

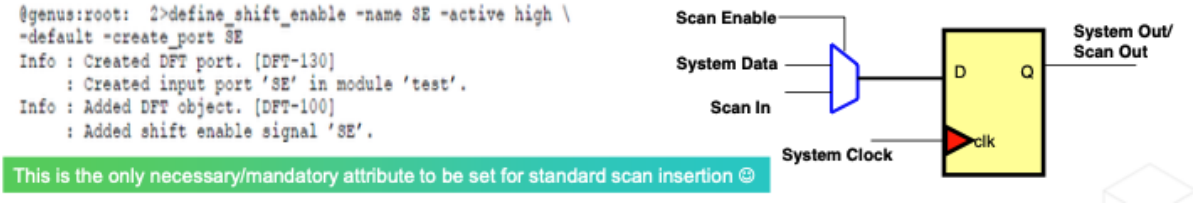

Figura 5

7. Run DFT Rule Checker

```
check_dft_rules
```

Verificação de regras básicas:

- Assegurar que os elementos das scan chains podem deslocar os dados
- Verificar a controlabilidade dos clocks dos registradores, e também se os sinais set/reset assíncronos podem assumir os valores adequados durante o deslocamento
- Verificar se todos os segmentos do registrador de deslocamento podem ser adequadamente controlados para deslocar os dados através de seus elementos

Os flip-flops que não passam na verificação são marcados para exclusão da scan chain. Assim, quanto menor o número de flip-flops com violações, maior é a cobertura de falhas.

É importante lembrar que a síntese DFT é limitada a cumprir os requisitos para inserção de scan chains, de maneira que as mesmas operem adequadamente no circuito sintetizado.

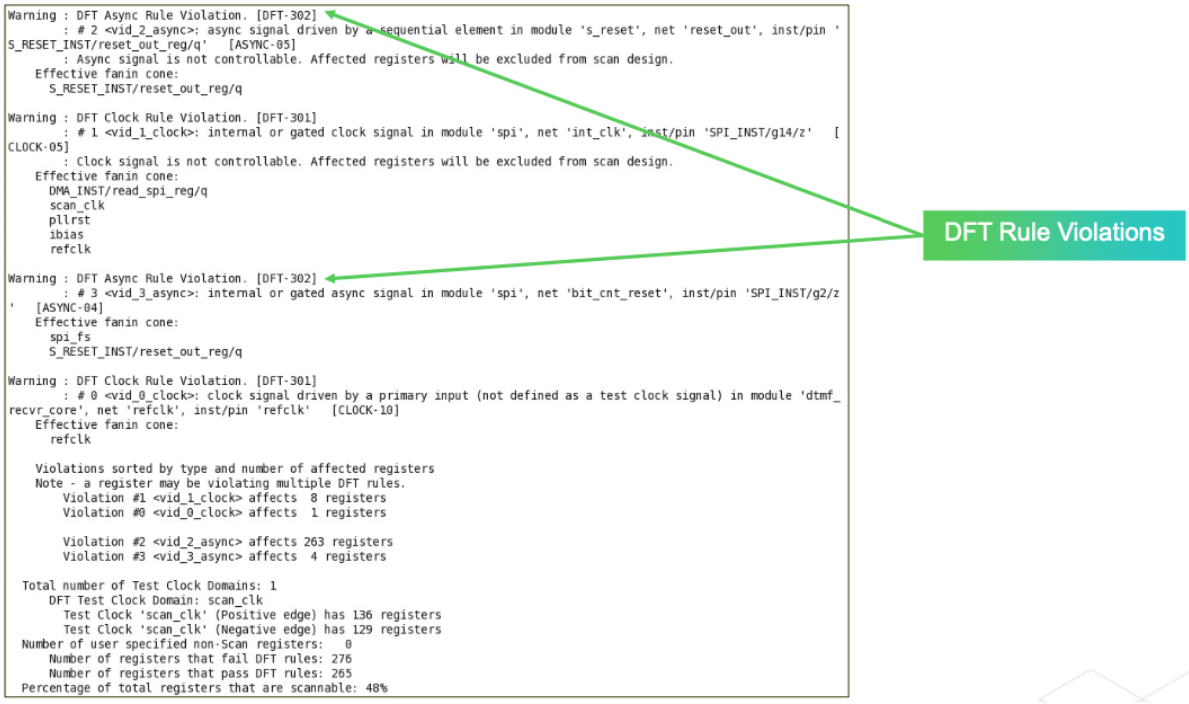

Figura 6

8. ***(Opcional) Fixing DFT Violations***
9. ***(Opcional) Add testability logic***
10. **Synthesize design and map to scan**: todos os registradores que passaram na verificação de regras DFT e que não estão marcados com dft_dont_scan ou preserve são mapeados para flip-flops scan durante a síntese.
11. **(Opcional) Testability analysis**
12. **Set up DFT configuration constraints, preview and connect scan chains**

Para especificar o número mínimo de *scan chains* a serem criadas, usar o comando abaixo. Por padrão, uma scan chain é inserida por borda ativa (subida ou descida) de cada domínio de clock.

```
set_db design:counter .dft_min_number_of_scan_chains 1
```

Não há limite para o comprimento da cadeia, mas o mesmo pode ser controlado por:

```
set_db <design> .dft_max_length_of_scan_chains integer
```

O comando para conectar as *scan chains* deve ser feito somente **após o *syn_map***.

```
connect_scan_chains -auto_create_chains
```

Após a conexão, realizar a síntese incremental para otimização:

```
syn_opt -incremental
```

Para mostrar as *scan chains*, basta usar o comando:

```
report_scan_chains
```

13. Netlist, SDC, ScanDEF, ATPG

Para gerar a netlist e os arquivos SDF e SDC:

```
write_hdl > outputs/counter_netlist_dft.v
write_sdf -timescale ns -nonegchecks -recrem split -edges check_edge -
setuphold split > output/def_delays.sdf
write_sdc > outputs/counter_sdc_dft.sdc
```

Para criar um arquivo de interface scanDEF para reordenar a scan chain no Innovus, usar:

```
write_scandef > outputs/counter_scanDEF.scandef
```

Para gerar os arquivos necessários um o template do script para executar o ATPG no Modus, usar o comando:

```
write_dft_atpg -library ../lib/slow_vdd1v0_basiccells.v
```

Após esse comando, será criado um diretório chamado **test_scripts** com os arquivos listados abaixo:

- **counter.test_netlist.v**: netlist em Verilog
- **runmodus.atpg.tcl**: script para executar no Modus
- **counter.FULLSCAN.pinassign** : arquivo para especificar o comportamento dos pinos I/O de teste
- **run_fullscan_sim_sdf**: arquivo para executar simulações *back annotated* ou *non_zero_delay*