DELCPG016 – Tutorial *Modus*

Objetivo: executar o fluxo básico de ATPG no Modus.

O Modus é integrado ao Genus (Figura 1), de maneira que os arquivos gerados neste último são utilizados para execução de testes e detecção de defeitos.

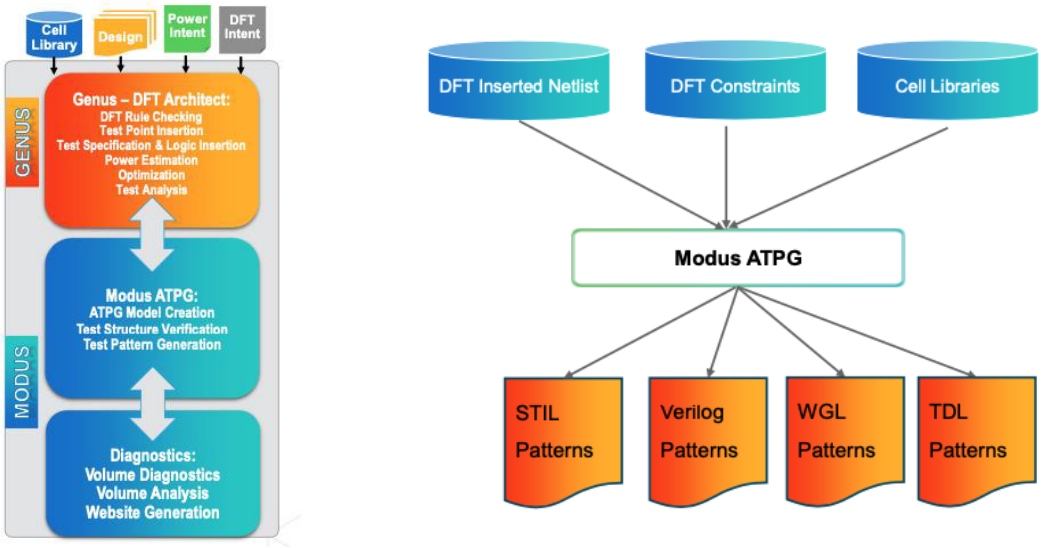

Figura 1 – Integração entre Genus e Modus

Arquivos de entrada:

- *Netlist* com inserção de DFT
- *Constraints*
- Células da biblioteca

Os arquivos de saída são os padrões:

- STIL
- Verilog
- WGL
- TDL

Para abrir o Modus, basta usar um dos comandos a seguir:

```
modus
```

```
modus -gui
```

A Figura 2 mostra a interface gráfica. Os comandos podem ser executados a partir dos menus ou da linha de comando. No lado esquerdo (*Task View*) aparece uma visão da metodologia, que permite ver as tarefas executadas na ordem determinada. É possível também criar metodologias de acordo com os requisitos de um projeto. Da mesma forma, é possível usar a interface gráfica sem necessariamente usar uma metodologia.

Quando uma determinada tarefa é selecionada, um formulário com informações específicas da mesma aparece em uma janela. A linha de comando também pode ser usada para editar e executar os comandos. Após a execução, o histórico dos mesmos pode ser importado usando o ícone do canto inferior esquerdo da interface.

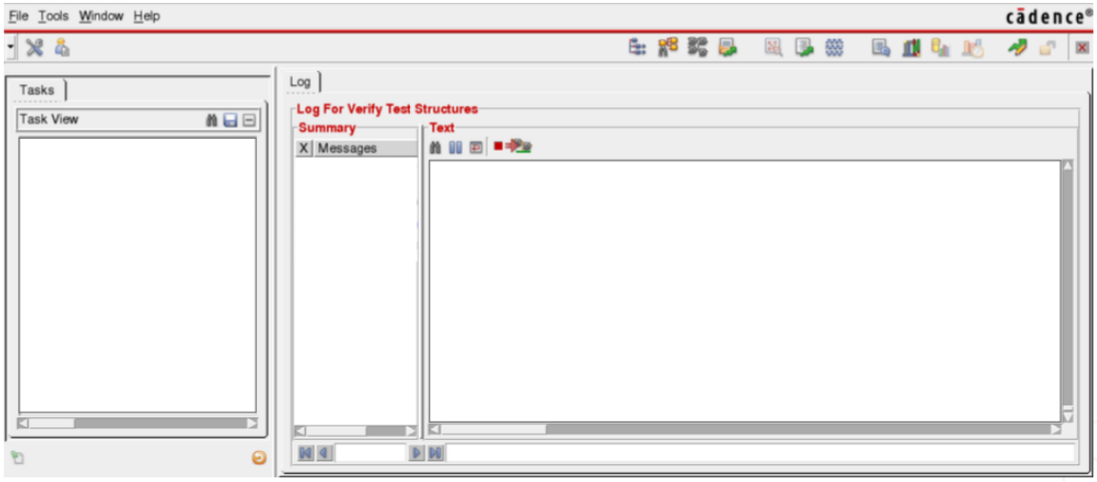

Figura 2

A Figura 3 mostra o fluxo digital e, em detalhe, o fluxo de geração de padrões de teste no Modus. Cada etapa será brevemente descrita a seguir.

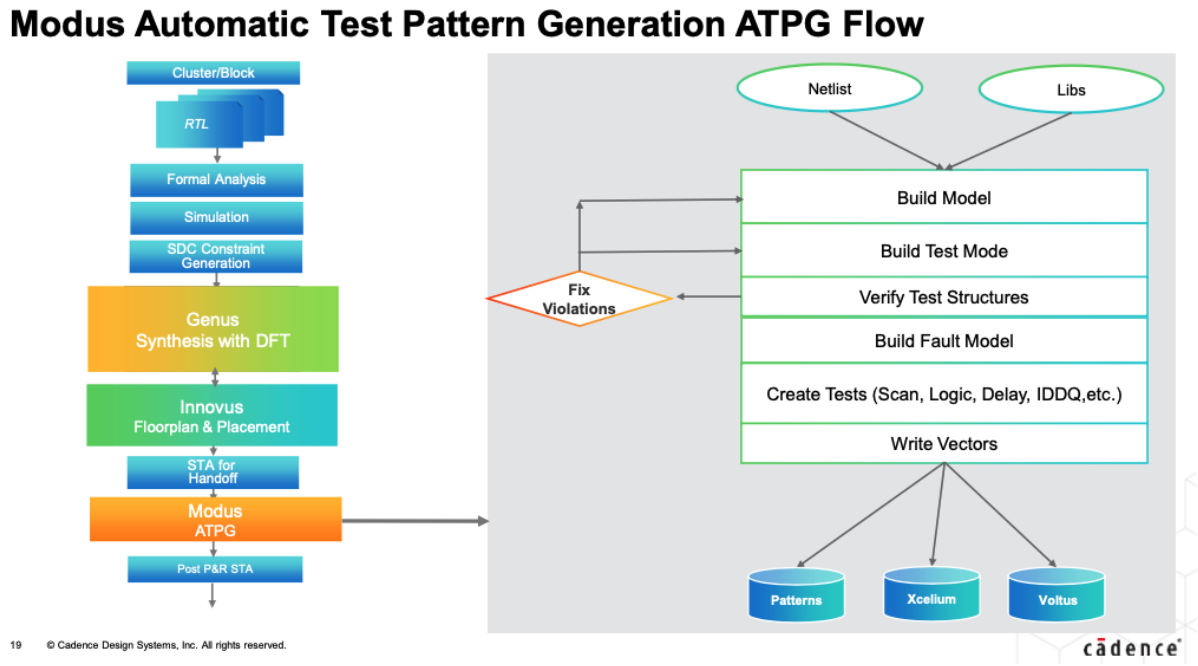

Figura 3

- ***Build Model***: trata-se da criação de um modelo para o Modus a partir da leitura da *netlist* gerada no Genus e dos arquivos da biblioteca da tecnologia. Opcionalmente, pode-se utilizar arquivos de *constraints*. A sintaxe do comando usado é:

```
build_model -workdir <mydir> -cell <to level> -designsource
    <netlist> -techlib <biblioteca> -allowmissingmodules
                        <yes/no>
```

O Modus utiliza a primeira definição da célula/módulo encontrada de acordo com a ordem de busca, que é(são) a(s) *design source(s)* e, em seguida, os arquivos da bilioteca da tecnologia. A saída gerada é uma *binary view* do projeto e um relatório contendo informações do processo executado.

Após a execução do comando, é possível ver um resumo das estatísticas do modelo contendo o número de objetos identificados (entradas e saídas primárias, *tied nets*, *dotted nets*, etc.). Esse resumo também pode ser gerado usando o comando abaixo:

```
report_model_statistics –workdir <directory>
```

- ***Build Test Mode***: o modo de teste especifica como o dispositivo deve ser configurado para teste, incluindo pinos com funções especiais (e.g., *scan clocks*), sequências para inicialização e *scanning*, bem como a lógica que será observável (ativa) durante o teste. Também define os requisitos e restrições do equipamento que vai processar os dados. O modo **FULLSCAN** é o teste tradicional com o par de I/O *scan-in* e *scan-out*. Outros modos incluem compressão, OPCG (*On-Product Clock Generation*) e o protocolo *scan* **IEEE 1149.1**. A sintaxe do comando é:

```
build_testmode -testmode <nome> -assignfile <arquivo
.pinassign> –modedef <nome do arquivo com definição de modo
de teste> -seqdf <arquivo de definição de sequência>
```

O arquivo **.pinassign** contém as informações sobre os pinos e suas funções associadas: nome do pino, funcionalidade e polaridade, respectivamente. A Figura 4 mostra a descrição da funcionalidade e dos valores.

Test Function	Value	Description
SC	0/-, 1/+	System Clock (value is the off state of clock)
EC	0, 1, -, +	Edge triggered Scan Clock (value is the off state of clock)
ES	0, 1, -, +	System Clock and Scan Clock
SI		Scan Input
SO		Scan Output
SE	0, 1, -, + or Z	Scan Enable (constant through scanning)
TI	0, 1, -, +, X or Z	Test Inhibit (constant through scanning and test)
TC	0, 1, -, +, X or Z	Test Constraint (constant through test)

Figura 4

Exemplo: `assign pin=SE test_function= +SE; #shift_enable`

Opcional: `report_test_structures -testmode <nome do testmode> -reportscanchain <all/summary/no>`

- **Verify Test Structure**: este comando analisa as regras de projeto *scan* para assegurar as operações de deslocamento e captura. A sintaxe do comando é:

  `verify_test_structures -testmode <nome do testmode> -workdir mydir`

- **Build Fault Model**: cria uma *database* de falhas para ATPG. Alguns exemplos de tipos de falhas são estáticos, dinâmicos (transição), iddq, etc. A sintaxe é:

  `build_faultmodel -workdir mydir -includedynamic <yes/no>`

- **Create Tests**: cria dois tipos de vetores de teste: *scan chains*, para detectar falhas ao longo da cadeia, e *logic tests*, para detectar falhas na funcionalidade da lógica. A sintaxe do comando é:

  `create_logic_tests -workdir mydir -testmode <nome do testmode> -experiment <nome exp> -effort <low/high>`

  OBS: O comando `create_scanchain_tests` cria testes para a *scan chain*, mas é automaticamente executado nos comandos `create_logic/logic_delay_tests`

  OBS: Para executar o teste de falhas dinâmicas, é necessário usar o comando `create_logic_delay_tests -testmode <nome do testmode> -experiment <nome exp>`. Esse teste consiste em gerar uma transição no local da falha e, em seguida, realizar uma captura. Assim, é mais complexo e requer mais processamento.

  O cálculo da cobertura de teste padrão do Modus é feito de acordo com as equações abaixo:

$$Test\ Coverage = \frac{Number\ of\ detected\ Faults}{Total\ \ Faults}$$

$$Adjusted\ Test\ Coverage = \frac{Number\ of\ detected\ Faults}{Total\ \ Faults - Redundant\ faults}$$

  O teste de cobertura **global** inclui todas as falhas no projeto, subtraindo as falhas ignoradas. Já o teste de cobertura ***testmode*** inclui somente as falhas globais que estão ativas no modo de teste.

- **Write vectors**: escrita dos vetores de teste nos formatos padrão usados na indústria. A sintaxe do comando é:

  ```
  write_vectors -testmode <nome do testmode> -
  inexperiment <nome exp sem commit> -language
  <linguagem> -scanformat <serial/paralelo> -
  outputfilename <nome>
  ```

Para o exemplo do contador, abrir o Modus no diretório work e definir o mesmo como diretório de trabalho:

```
modus -gui
set_db workdir .
```

Compilar a netlist e as bibliotecas em um modelo de teste para o Modus.

```
build_model -workdir . -designsource ../deliverables/counter_dft_dft.v -
techlib <path_GPDK045>/gsclib045/verilog/slow_vdd1v0_basicCells.v -
designtop counter_dft
```

Opcional: Comando para reportar informações estruturais básicas do modelo lógico (e.g., número de portas lógicas, blocos e nós, entradas e saídas primárias):

```
report_model_statistics -workdir .
```

Executar o ***build test mode***:

```
build_testmode -workdir . -testmode FULLSCAN -assignfile
test_scripts/counter_dft.FULLSCAN.pinassign
```

Opcional: Comando para reportar informações do circuito com base em um determinado modo de teste:

```
report_test_structures -testmode FULLSCAN -reportscanchain all
```

Verificar as regras de projeto:

```
verify_test_structures -testmode FULLSCAN -workdir .
```

Criar modelo de falhas:

```
build_faultmodel -workdir . -includedynamic yes
```

Criar vetores de teste:

```
create_logic_tests -testmode FULLSCAN -experiment logic
```