Tutorial de uso do Vitis_HLS

Tabela de Conteúdos

Informações iniciais

- Exemplo utilizado: MatchTemplate_Demo.cpp
- Para facilitar a execução do Vitis_HLS pela CLI, um arquivo Makefile contendo os comandos necessários foi criado.

Relatório de Funcionamento do Makefile - Vitis Vision Project

Descrição Geral

Este Makefile organiza e automatiza o processo de síntese, simulação e implementação de um projeto usando Vitis HLS com as bibliotecas Vision da AMD/Xilinx.

Ele permite ao usuário executar comandos como:

```
make run TARGET=csynth PLATFORM=<nome> PLATFORM_REPO_PATHS=<caminhos>
```

ou

```
make run TARGET=cosim XPART=xcu200-fsgd2104-2-e
```

Variáveis e Caminhos

`XF_PROJ_ROOT`: Caminho raiz para as bibliotecas do Vitis Vision.

`CUR_DIR`: Diretório atual do projeto.

`TARGET`: Define a ação desejada (`csim`, `cosim`, `csynth`, `vivado_syn`, `vivado_impl`).

`XPART`: Define o part number do FPGA. Ex: `xc7s25csga324-1`.

`PLATFORM`: Nome da plataforma para Vitis.

`PLATFORM_REPO_PATHS`: Diretórios onde buscar as plataformas.

`CONFIG_FILE`: Arquivo de configuração usado pelo Vitis.

`CONFIG_TMPL`: Template para gerar o `CONFIG_FILE`.

Alvos Disponíveis

Target	Descrição
`csim`	Simulação em C (C Simulation)
`cosim`	Co-simulação com RTL
`csynth`	Síntese em C
`vivado_syn`	Síntese com Vivado
`vivado_impl`	Implementação com Vivado
`clean`	Remove arquivos gerados
`help`	Mostra instruções de uso

Etapas do Makefile

1. `help`
    - Imprime no terminal instruções de uso do make.
2. Definições de ambiente

- Define `XF_PROJ_ROOT` e atualiza `PATH` com `XILINX_VIVADO`.
- Se presente, executa `ldlibpath.sh` para configurar `LD_LIBRARY_PATH`.

3. Escolha do modo Vivado
   - Se `TARGET=vivado_syn`, define `VIVADO_FLOW=syn`.
   - Senão, define `VIVADO_FLOW=impl`.
4. Verificações de ambiente

   As regras `check_vivado`, `check_vpp` e `check_opencv` verificam se os caminhos corretos estão configurados:

Verificação	O que é validado
`XILINX_VIVADO`	Caminho para o executável do Vivado
`XILINX_VITIS`	Caminho para o compilador Vitis (v++)
`OPENCV_INCLUDE`	Diretório de includes do OpenCV
`OPENCV_LIB`	Diretório das bibliotecas compiladas do OpenCV

   Se algo estiver faltando, o `make` falha com uma mensagem apropriada.

5. Geração de Configuração (`CONFIG_FILE`)

   Gera um arquivo de configuração substituindo variáveis do template (hls_config.tmpl) com os valores de ambiente:

```
import os, string
with open('hls_config.tmpl', 'r') as fr:
  t = fr.read()
with open('hls_config.cfg', 'w') as f:
  f.write(string.Template(t).substitute(**dict(os.environ)))
```

   Executado via interpretador Python fornecido pelo Vitis.

6. Alvo `all`
   - Executa verificações de ambiente.
   - Gera o arquivo de configuração.
   - Executa `v++ -c --mode hls` se o `TARGET_REL` não for `csim`.

```
v++ -c --mode hls --config hls_config.cfg --work_dir hls --part
<XPART>
```

7. Alvo `run`
    - Chama o alvo `all`.
    - Se o `TARGET_REL` não for `csynth`, roda o `vitis-run` correspondente:

```
vitis-run --mode hls --config hls_config.cfg --cosim --work_dir hls --
part <XPART>
```

8. Alvo `clean`

   Remove arquivos temporários e diretório de trabalho:

```
rm -rf hls_config.cfg *_hls.log hls
```

Exemplo de Uso

Simular C++ com modelo `csim`:

```
make run TARGET=csim XPART=xcu200-fsgd2104-2-e
```

Rodar co-simulação RTL:

```
make run TARGET=cosim XPART=xcu200-fsgd2104-2-e
```

Requisitos

Certifique-se de que as seguintes variáveis de ambiente estão configuradas corretamente:

- `XILINX_VIVADO`
- `XILINX_VITIS`
- `OPENCV_INCLUDE`
- `OPENCV_LIB`

Relatório: Explicação do Arquivo `hls_config.tmpl`

O arquivo `hls_config.tmpl` define os parâmetros de configuração usados pelo compilador HLS do Vitis para controlar as etapas de síntese, simulação, compilação e geração de RTL. Ele é processado dinamicamente pelo `Makefile`, substituindo as variáveis de ambiente (como `$CUR_DIR`, `${XF_PROJ_ROOT}`, etc.) por seus valores reais.

Seção [hls]

Essa seção agrupa todas as opções relacionadas ao processo de HLS.

`clock=3.3`

- Define o período do clock em nanosegundos (ns).
- Neste caso: 3.3 ns → ~303 MHz.

`flow_target=vivado`

- Especifica que o fluxo de síntese alvo será o **Vivado HLS**.

`syn.file=$CUR_DIR/src/match_template.cpp`

- Arquivo principal da função a ser sintetizada.
- Usualmente é a **função top-level** do design HLS.

syn.file_cflags=...

- Flags de compilação C++ usadas na síntese.
- Inclui:
  - Includes da Vitis Vision (`${XF_PROJ_ROOT}/L1/include`)
  - Diretório atual (`./`)
  - Define `__SDSVHLS__` para habilitar macros específicas do fluxo HLS.
  - Usa `-std=c++0x` (modo C++11).

`syn.file_csimflags=...`

- Flags de compilação para a simulação C (csim) da função top-level.

`syn.top=match_template`

- Define o nome da função top-level a ser sintetizada.
- Essa função deve estar presente no arquivo `match_template.cpp`.

`tb.file=$CUR_DIR/tb/testbench.cpp`

- Caminho para o testbench usado em simulações.
- Este testbench deve chamar a função match_template com dados de entrada e verificar a saída.

`tb.file_cflags=...`

- Flags de compilação C++ usadas ao compilar o testbench para síntese/simulação.

- Inclui cabeçalhos do OpenCV (`${OPENCV_INCLUDE}`), Vitis Vision e define `__SDSVHLS__`.

`tb.file_csimflags=...`

- Flags de compilação do testbench específicas para simulação em C (`csim`).
- Semelhante ao anterior, mas sem incluir OpenCV diretamente (o que pode ser necessário em `csim`).

Flags de Linkagem

Essas opções definem bibliotecas OpenCV que devem ser linkadas durante diferentes etapas:

`csim.ldflags=...`

- Bibliotecas para linkar durante simulação C.
- Inclui:
    - `opencv_core`, `imgproc`, `imgcodecs`, `highgui`, `flann`, `features2d`.

`cosim.ldflags=...`

- Bibliotecas para co-simulação RTL.

`sim.ldflags=...`

- Flags gerais de linkagem de simulações.

`vivado.flow=${VIVADO_FLOW}`

- Define se o Vivado será usado no modo de síntese (`syn`) ou implementação (`impl`).
- Controlado dinamicamente via Makefile com base na variável `TARGET`.

`vivado.rtl=verilog`

```
Define que a saída RTL será em Verilog (em vez de VHDL, por exemplo).
```

Exemplo de Variáveis Substituídas

Durante a execução, o `Makefile` substitui as variáveis como:

```
$CUR_DIR => /home/usuario/projeto_hls
${XF_PROJ_ROOT} => /tools/Xilinx/Vitis/2024.2/Vitis_Libraries/vision
```

Observações Finais

- Este arquivo é essencial para a integração com o `Makefile`, pois define todos os caminhos e parâmetros que o compilador HLS utilizará.
- Erros nesse arquivo, como caminhos inválidos ou bibliotecas faltantes, causarão falhas nas etapas `csim`, `cosim` ou `csynth`.