Projeto com Circuitos Reconfiguráveis

Vivado – Non project mode

Prof. Daniel M. Muñoz Arboleda

FGA - UnB

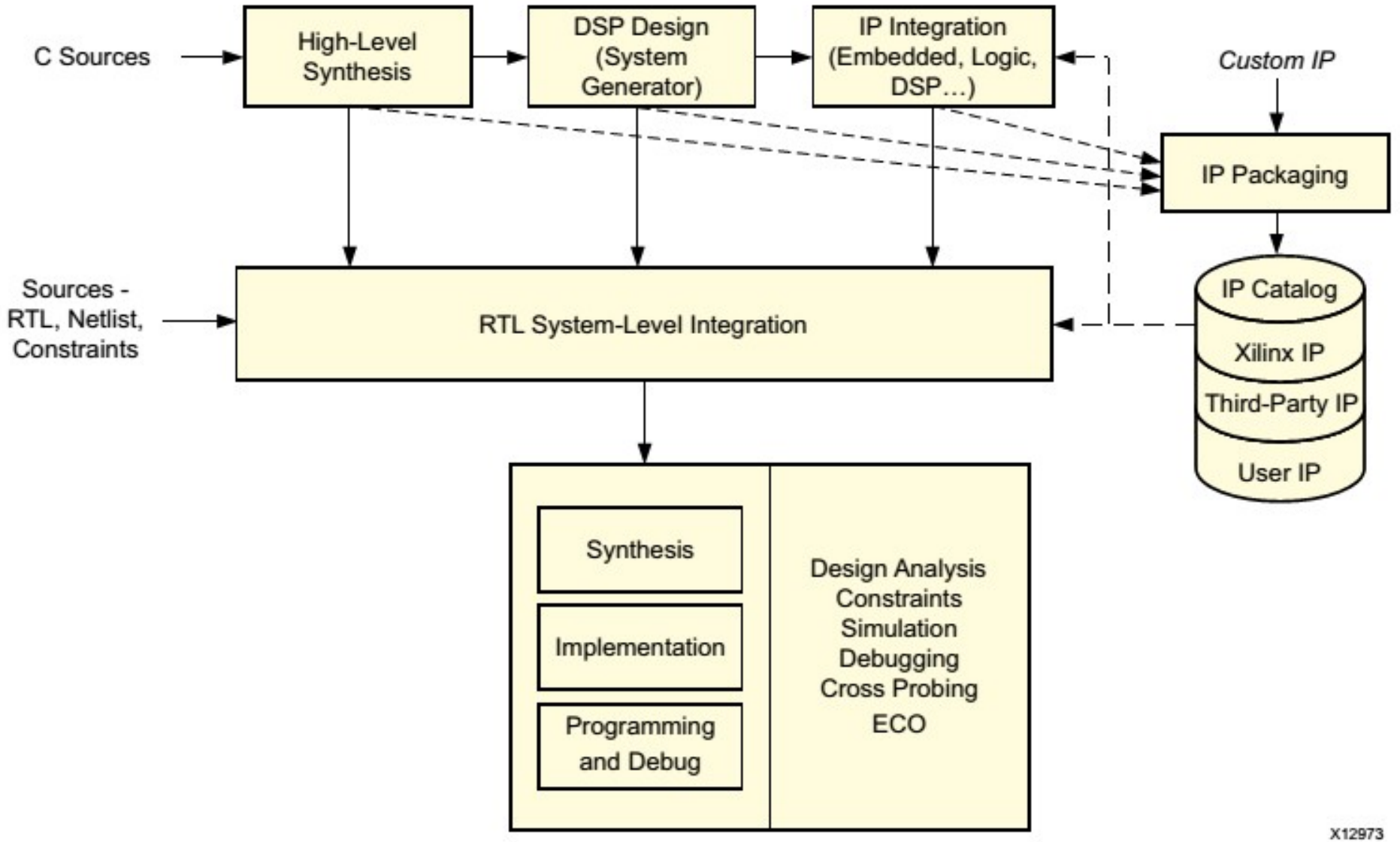
Plano de Aula

- Overview dos fluxos de projeto no Vivado
- Vivado no modo projeto
- Vivado no modo *Non-Project*
- Explicação dos comandos básicos
- Criação de scripts no Vivado
- Exemplos

Fluxo de Projeto: RTL to Bitstream Design Flow

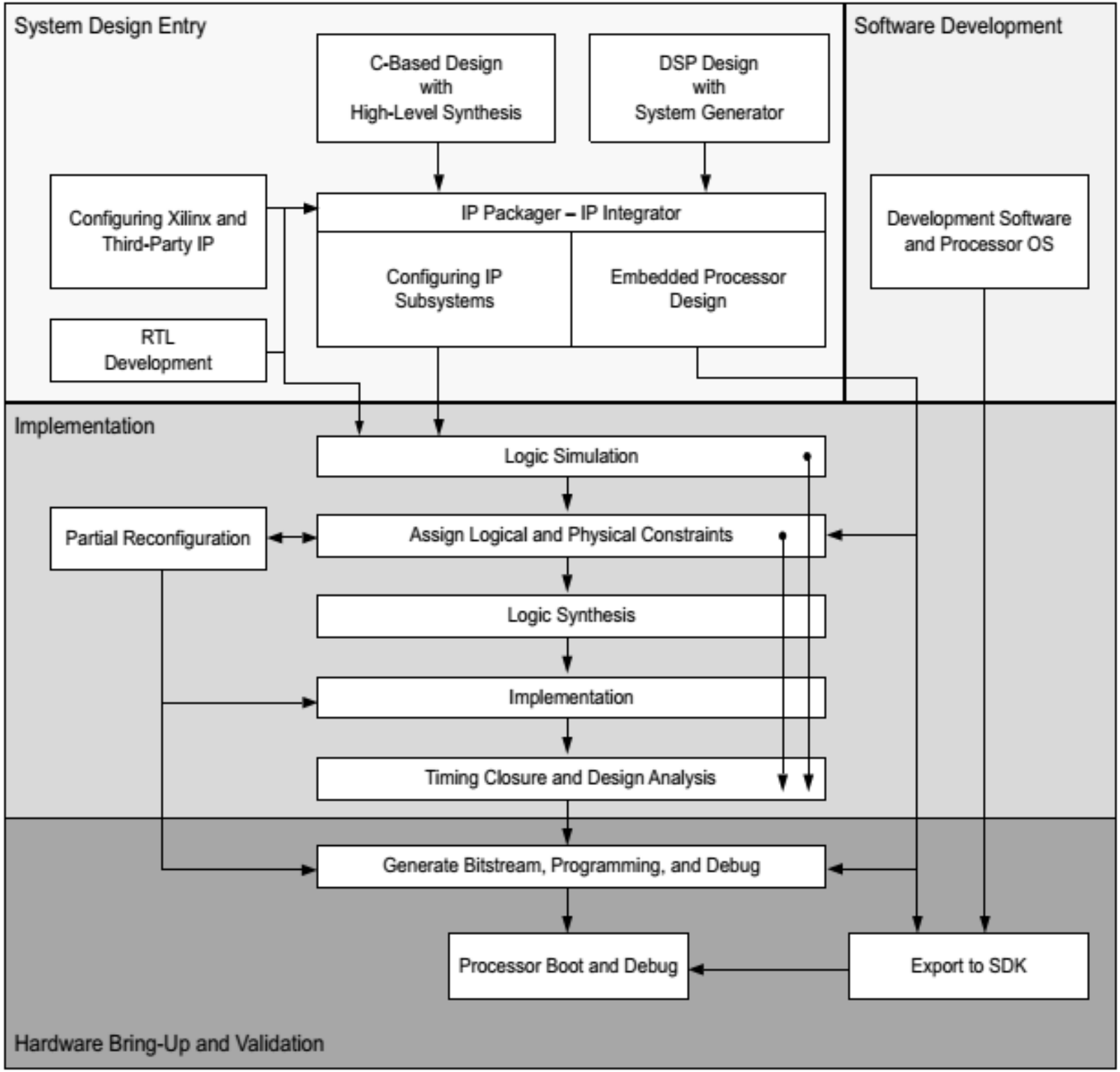
1) Projeto RTL
2) IP design e Integração a nível de sistema
3) IP subsystem design
4) IO e clock planning
5) Synthesis
6) Design analysis e simulação
7) Placement and Routing
8) Hardware debug and validation

Fluxo de Projeto: RTL to Bitstream Design Flow



Fluxos de Projeto Alternativos

- Embedded processor design flow (Vivado IP integrator e SDK)
- Model-based DSP design using System Generator (Matlab)
- High-level synthesis (C-based design)
- Partial reconfiguration design
- Hierarchical design (também conhecido síntese incremental)


System Design Entry
Software Development
C-Based Design with High-Level Synthesis
DSP Design with System Generator
Configuring Xilinx and Third-Party IP
IP Packager – IP Integrator
Configuring IP Subsystems
Embedded Processor Design
RTL Development
Development Software and Processor OS
Implementation
Logic Simulation
Partial Reconfiguration
Assign Logical and Physical Constraints
Logic Synthesis
Implementation
Timing Closure and Design Analysis
Generate Bitstream, Programming, and Debug
Processor Boot and Debug
Export to SDK
Hardware Bring-Up and Validation

Vivado: Modos de Uso

- Vivado tem dois modos de uso: Modo Projeto e Modo Non-Project.
- Ambos podem ser usados usando o Vivado IDE ou através de comandos tcl e scripts.

Vivado: Modo Projeto

- Arquitetura baseada em projeto: Vivado cria uma estrutura de pastas no disco.
- O Vivado gerencia inteiramente o processo de projeto, incluindo as dependências de dados, geração de reportes, armazenamento de dados, etc.
  - Se os arquivos HDL são modificados, o Vivado solicita sintetizar novamente.
  - Se os constraints são modificados, o Vivado solicita sintetizar novamente, reimplementação, ou ambos.
  - Após *routing*, Vivado gera automaticamente reportes de timing, DRC, power, entre outros.
  - O fluxo inteiro pode ser executado com um click.

Vivado: Modo Non-Project

- Total controle sobre cada etapa do fluxo de projeto: o usuário gerencia os arquivos fonte e o processo de projeto através de comandos tcl ou scripts.
- Estilo de compilação direto: source files são facilmente importados e processados desde a síntese até a implementação.
- Através de comandos tcl podem ser configurados parâmetros de design e opções de implementação. O usuário pode criar *design checkpoints* e reportes em cada etapa do fluxo.
- Cada etapa do fluxo pode ser executada individualmente, permitindo analisar os resultados após cada etapa.

Vivado: Modo Non-Project

- *In-memory compilation:* no modo non-project o design é descartado da memória após cada seção e só é armazenado em disco o que o usuário solicita.
- *In-memory compilation*: Não tem sobrecarga computacional relativa ao projeto pois todo o processamento (desde síntese até geração de bitstream) é feito em memória.
- Se necessário permite uso do GUI: *start_gui*, *stop_gui* para realizar *design analysis* ou *constraints assingment*.
- Tcl API: permitem configurar o design, configurar ferramentas de síntese, placement e routing, permitem reporte robusto.

Project Mode

Non-Project Mode

GUI	Tcl Script	Tcl Script
	create_project … add_files … import_files … … launch_run synth_1 wait_on_run synth_1 open_run synth_1 report_timing_summary launch_run impl_1 wait_on_run impl_1 open_run impl_1 report_timing_summary launch_run impl_1 –to_step_write_bitstream wait_on_run impl_1	read_verilog … read_vhdl … read_ip … read_xdc … read_edif … … synth_design … report_timing_summary write_checkpoint opt_design write_checkpoint place_design write_checkpoint route_design report_timing_summary write_checkpoint write_bitstream

Comandos no Modo Non-Project

Command	Description
read_edif	Imports an EDIF or NGC netlist file into the Design Source fileset of the current project.
read_verilog	Reads the Verilog (.v) and System Verilog (.sv) source files for the Non-Project Mode session.
read_vhdl	Reads the VHDL (.vhd or .vhdl) source files for the Non-Project Mode session.
read_ip	Reads existing IP (.xci or .xco) project files for the Non-Project Mode session. For Vivado IP (.xci), the design checkpoint (.dcp) synthesized netlist is used to implement the IP if the netlist is in the IP directory. If not, the IP RTL sources are used for synthesis with the rest of the top-level design. The .ngc netlist is used from the .xco IP project. ***Note:*** The .xco file is no longer supported in UltraScale device designs.

Comandos no Modo Non-Project

Command	Description
read_checkpoint	Loads a design checkpoint into the in-memory design.
read_xdc	Reads the .sdc or .xdc format constraints source files for the Non-Project Mode session.
read_bd	Reads existing IP Integrator block designs (.bd) for the Non-Project session.
set_param set_property	Used for multiple purposes. For example, it can be used to define design configuration, tool settings, and so forth.
link_design	Compiles the design for synthesis if netlist sources are used for the session.
synth_design	Launches Vivado synthesis with the design top module name and target part as arguments.

Comandos no Modo Non-Project

opt_design	Performs high-level design optimization.
power_opt_design	Performs intelligent clock gating to reduce overall system power. This is an optional step.
place_design	Places the design.
phys_opt_design	Performs physical logic optimization to improve timing or routability. This is an optional step.
route_design	Routes the design.
report_*	Runs a variety of standard reports, which can be run at different stages of the design process.
write_bitstream	Generates a bitstream file and runs DRCs.
write_checkpoint	Saves the design at any point in the flow. A design checkpoint consists of the netlist and constraints with any optimizations at that point in the flow as well as implementation results.
start_gui stop_gui	Opens or closes the Vivado IDE with the current design in memory.

Comandos para executar o Vivado no Windows ou Linux

- vivado
- vivado – mode tcl : invoca o Vivado Design Suite Tcl shell
- vivado -mode batch -source <your_Tcl_script>

Comandos para executar o Vivado no Windows ou Linux

- vivado
- vivado – mode tcl : invoca o Vivado Design Suite Tcl shell
- vivado -mode batch -source <your_Tcl_script>: invoca o Vivado Design Suite e executa o script indicado pelo usuário.

Exemplo: Vivado Non-Project Mode

1) Abrir Vivado Tcl Shell
2) Trocar diretório para
   <install_dir>/Vivado/201x.x/examples/Vivado_Tutorial
   Por exemplo:
   cd c:/xilinx/Vivado/2016.2/examples/Vivado_Tutorial
3) Executar script
   source ./create_bft_kintex7_batch.tcl
4) Observar resultados na pasta *Tutorial_Created_Data*

Exemplo: Vivado no Modo Projeto usando script

1) Abrir Vivado Tcl Shell
2) Trocar diretório para
   <install_dir>/Vivado/201x.x/examples/Vivado_Tutorial
   Por exemplo:
   cd c:/xilinx/Vivado/2016.2/examples/Vivado_Tutorial
3) Executar script
   source ./run_bft_kintex7_project.tcl
4) Observar resultados na pasta *Tutorial_Created_Data*

Exercício:

1) Criar script para o exemplo ping-pong-leds usando a placa de desenvolvimento Basys3
2) Crie design checkpoint e reportes de utilização de recuros, timing e consumo de potência após síntese e implementação (placement and routing).
3) Crie o bitstream
4) Incluir comando start_gui para verificação do resultado no Vivado
5) Execute o script no Vivado Modo Projeto.
6) Programe e verifique o comportamento na placa.

Referencias

1) Xilinx User Guide UG892 Vivado Design Flows Overview, 2017.
2) Xilinx User Guide UG894 Vivado Design Suite using Tcl Scripting, 2017.