

# Advanced Digital System Design

by

Dr. Lesley Shannon

Email: [lshannon@ensc.sfu.ca](mailto:lshannon@ensc.sfu.ca)

Course Website: <http://www.ensc.sfu.ca/~lshannon/>



*Simon Fraser University*

Tutorial Slide Set: 2

Date: September 22, 2011

# Slide Set Overview

---

- The EDK Tool Set
  - You use Xilinx Platform Studio (the GUI)
    - How does it work?
      - What happens “under-the-hood”
    - What’s the file structure for an EDK project?

# Key EDK Project Files

---

- Key System-Level Files:
  - XMP
  - MHS
  - MSS
  - 2 MakeFiles
  - Also in the main directory there are:
    - LogFiles
    - A Shortcut to iMPACT and its command line

# Key EDK Project Files

---

- XMP:
  - Xilinx Microprocessor Project
  - Automatically generated by XPS
  - Stores Project Information
  - You can open this file with a text editor to look at it

# Key EDK Project Files

---

- XMP:
  - Contains information about the:
    - Version of XPS being used, what synthesis tools and simulator are being used
    - What FPGA Device, Package and Speed Grade are used
    - What the are the names of the MHS, MSS, and peripheral repository directory
    - Info about all the processors in the system and their software projects
  - Do NOT modify it unless you know what you are doing
    - You don't know what you're doing



# Key EDK Project Files

---

- MHS:
  - Microprocessor Hardware Specification
  - Used to describe the system's hardware platform
  - Captures both the system's external ports and internal components
  - Shared net names are used to connect ports on components and/or external ports

# Key EDK Project Files

---

- MHS External Ports:

```
PORT fpga_0_RS232_Uart_1_RX_pin = fpga_0_RS232_Uart_1_RX, DIR = I
PORT fpga_0_RS232_Uart_1_TX_pin = fpga_0_RS232_Uart_1_TX, DIR = O
PORT fpga_0_net_gnd_pin = net_gnd, DIR = O
PORT fpga_0_net_gnd_1_pin = net_gnd, DIR = O
PORT fpga_0_net_gnd_2_pin = net_gnd, DIR = O
PORT fpga_0_net_gnd_3_pin = net_gnd, DIR = O
PORT fpga_0_net_gnd_4_pin = net_gnd, DIR = O
PORT fpga_0_net_gnd_5_pin = net_gnd, DIR = O
PORT fpga_0_net_gnd_6_pin = net_gnd, DIR = O
PORT sys_clk_pin = dcm_clk_s, DIR = I, SIGIS = CLK, CLK_FREQ =
    100000000
PORT sys_rst_pin = sys_rst_s, DIR = I, RST_POLARITY = 0, SIGIS = RST
```

# Key EDK Project Files

---

- MHS MicroBlaze Declaration:

```
BEGIN microblaze
  PARAMETER INSTANCE = microblaze_0
  PARAMETER HW_VER = 5.00.c
  PARAMETER C_USE_FPU = 0
  PARAMETER C_DEBUG_ENABLED = 0
  PARAMETER C_NUMBER_OF_PC_BRK = 1
  BUS_INTERFACE DLMB = dlmb
  BUS_INTERFACE ILMB = ilmb
  BUS_INTERFACE DPLB = mb_plb
  BUS_INTERFACE IPLB = mb_plb
END
```

# Key EDK Project Files

---

- MHS PLB Microprocessor Debug Module (MDM):

```
BEGIN plb_mdm
  PARAMETER INSTANCE = debug_module
  PARAMETER HW_VER = 2.00.a
  PARAMETER C_MB_DBG_PORTS = 0
  PARAMETER C_USE_UART = 1
  PARAMETER C_UART_WIDTH = 8
  PARAMETER C_BASEADDR = 0x41400000
  PARAMETER C_HIGHADDR = 0x4140ffff
  BUS_INTERFACE SPLB = mb_plb
END
```

# Key EDK Project Files

---

- MHS Digital Clock Manager (DCM):

```
BEGIN dcm_module
  PARAMETER INSTANCE = dcm_0
  PARAMETER HW_VER = 1.00.a
  PARAMETER C_CLK0_BUF = TRUE
  PARAMETER C_CLKIN_PERIOD = 10.000000
  PARAMETER C_CLK_FEEDBACK = 1X
  PARAMETER C_DLL_FREQUENCY_MODE = LOW
  PARAMETER C_EXT_RESET_HIGH = 1
  PORT CLKIN = dcm_clk_s
  PORT CLK0 = sys_clk_s
  PORT CLKFB = sys_clk_s
  PORT RST = net_gnd
  PORT LOCKED = dcm_0_lock
END
```

# Key EDK Project Files

---

- MSS:
  - Microprocessor Software Specification
  - Used to describe the system's software platform
  - Describes each processor and its OS along with the Drivers associated with each peripheral

# Key EDK Project Files

---

- MSS Processor and its OS:

```
BEGIN OS
```

```
  PARAMETER OS_NAME = standalone
```

```
  PARAMETER OS_VER = 1.00.a
```

```
  PARAMETER PROC_INSTANCE = microblaze_0
```

```
  PARAMETER STDIN = RS232_Uart_1
```

```
  PARAMETER STDOUT = RS232_Uart_1
```

```
END
```

```
BEGIN PROCESSOR
```

```
  PARAMETER DRIVER_NAME = cpu
```

```
  PARAMETER DRIVER_VER = 1.01.a
```

```
  PARAMETER HW_INSTANCE = microblaze_0
```

```
  PARAMETER COMPILER = mb-gcc
```

```
  PARAMETER ARCHIVER = mb-ar
```

```
  PARAMETER XMDSTUB_PERIPHERAL = debug_module
```

```
END
```

# Key EDK Project Files

---

- Examples of MSS Bus Peripheral Device Driver Declarations:

```
BEGIN DRIVER
```

```
  PARAMETER DRIVER_NAME = plbarb
```

```
  PARAMETER DRIVER_VER = 1.02.a
```

```
  PARAMETER HW_INSTANCE = mb_plb
```

```
END
```

```
BEGIN DRIVER
```

```
  PARAMETER DRIVER_NAME = bram
```

```
  PARAMETER DRIVER_VER = 1.00.a
```

```
  PARAMETER HW_INSTANCE = dlmb_cntlr
```

```
END
```

# Key EDK Project Files

---

- Examples of MSS Peripheral Device Driver Declarations:

```
BEGIN DRIVER
  PARAMETER DRIVER_NAME = uartlite
  PARAMETER DRIVER_VER = 1.01.a
  PARAMETER HW_INSTANCE = debug_module
END
```

```
BEGIN DRIVER
  PARAMETER DRIVER_NAME = uartlite
  PARAMETER DRIVER_VER = 1.01.a
  PARAMETER HW_INSTANCE = RS232_Uart_1
END
```

# Key EDK Project Files

---

- Make Files:
  - system.make
  - System\_incl.make
  - The power behind the XPS tools
  - When you click a command in the GUI, it corresponds to a make file statement

# Key EDK Project Files

---

- 2 Make Files:
  - system.make & system\_incl.make
    - Automatically generated by XPS
  - system\_incl.make
    - Contains all options and settings defined in the form of macros
  - system.make (Main Make File)
    - Contains all the targets and commands in the complete flow
    - Contains: `include system_incl.make`

# Key EDK Project Files

---

- 2 Make Files:
  - Users can generate their own make file
    - Base it on system.make
      - Give it a *different* name
    - Include system\_incl.make
  - Change system.make file after:
    - Adding, deleting, or renaming a processor
    - Adding, deleting, or renaming a software application
    - Changing the implementation tools used: ISE (Project Navigator), Xplorer (in XPS), and Xflow (in XPS)

# Other Project Files in the root directory

---

- Log Files from:
  - XPS itself
  - The Platform Generator
  - The Library Generator
  - The Bitstream Intializer
- iMPACT:
  - Used to download the bitstream to the FPGA
    - XPS includes a shortcut to iMPACT and the appropriate command line
- The Platform Generator's options file is also in the root directory

---

Now let's look at what information XPS puts in the project subdirectories :

---

# EDK Project Directory Structure

---

- Directory Structure:
  - \_\_xps
  - data
  - etc
  - hdl
  - implementation
  - synthesis
  - pcores
  - “project”\_software
  - Processor directories
    - e.g. microblaze\_0
  - Other directories you want
    - e.g. a “code” directory

# EDK Project Directory Structure

---

- What's in `__xps/`:
  - Contains option files for most of the tools in the tool flow:
    - Platform Generator
    - Library Generator
    - Compiler
    - Bitstream Initializer
    - etc

# EDK Project Directory Structure

---

- What's in **data/**:
  - Contains the User Constraint File
    - system.ucf
    - Describes how the external ports of your design connect to the physical pins of the FPGAs
    - Max clock frequency

# EDK Project Directory Structure

---

- Examples of UCF statements describing clk, rst and UART Tx Pins:

```
Net sys_clk_pin LOC=AJ15;
Net sys_clk_pin IOSTANDARD = LVCMOS25;
Net sys_rst_pin LOC=AH5;
Net sys_rst_pin IOSTANDARD = LVTTTL;
## System level constraints
Net sys_clk_pin TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 10000 ps;
Net sys_rst_pin TIG;

Net fpga_0_RS232_Uart_1_TX_pin LOC=AE7;
Net fpga_0_RS232_Uart_1_TX_pin IOSTANDARD = LVCMOS25;
Net fpga_0_RS232_Uart_1_TX_pin SLEW = SLOW;
Net fpga_0_RS232_Uart_1_TX_pin DRIVE = 12;
```

# EDK Project Directory Structure

---

- What's in **etc/**:
  - Contains option files for:
    - the Bitstream Generator
    - connecting with the stub (remember you use MDM)
      - Do you understand the difference between the stub and MDM?
    - option file for Xilinx FPGA Implementation Flow for Fast Runtime

# EDK Project Directory Structure

---

- What's in **hd1/**:
  - Contains the wrapper files generated by the Platform Generator

# EDK Project Directory Structure

---

- What's in **implementation/**:
  - Contains the **output** files from the following tools:
    - Xilinx Synthesis Technology aka XST (.ngc)
    - NGDBuild (.ngd)
    - Mapper, Place & Route aka PAR (.ncd)
    - Bitstream Generator (.bit files and .bmm files)

# EDK Project Directory Structure

---

- What's in **synthesis/**:
  - Used for XST (Xilinx Synthesis Technology)
  - Contains for each IP core:
    - A tool option file (.scr)
    - A list of sub-components to be synthesized (.iso)
      - the core itself
      - The wrapper file from the Platform Generator (in the hdl directory)
    - A Project File listing all the files to be synthesized for each component (.prj)
    - A Log File (.srp)

# EDK Project Directory Structure

---

- What's in **pcores/**:
  - Can be used as a local repository for any IP cores included in the system
  - Remember when creating the project you can make reference to a global repository of IP cores

# EDK Project Directory Structure

---

- What's the structure of an IP core's directory:
  - Each IP core (e.g. `snoopy_v1_00_a`) has three subdirectories
    - `data`
    - `hdl` (`/vhdl` or `/verilog`)
    - `doc`
  - Contents of `data/` directory:
    - Microprocessor Peripheral Definition (MPD) File
    - Peripheral Analyze Order (PAO) File

# EDK Project Directory Structure

---

- MPD File:
  - Defines Peripheral Options
    - HDL, Core Status, provides a core description
  - Defines Bus interface
    - Slave/Master and PLB/LMB/OPB/FLSL
  - Defines customizable parameters (includes defaults)
    - C\_BASEADDR, C\_HIGHADDR, bus width, etc
  - Defines port interfaces
    - I/O, signal width, fixed vs customizable port connections

# EDK Project Directory Structure

---

- PAO File:
  - Defines Project Hierarchy and synthesis order

```
lib snoopy_v1_00_a snoopy_types
lib snoopy_v1_00_a reg
lib snoopy_v1_00_a flipflop
lib snoopy_v1_00_a clk_cntr
lib snoopy_v1_00_a mux
lib snoopy_v1_00_a valid_opb_addr
lib snoopy_v1_00_a valid_pc_addr
lib snoopy_v1_00_a opb_output
lib snoopy_v1_00_a var_instr_cntrs
lib snoopy_v1_00_a snoopy
```

# EDK Project Directory Structure

---

- What's the structure of an IP core's directory:
  - Contents of **hdl/** directory:
    - A VHDL subdirectory and/or
    - A Verilog subdirectory
  - Contents of the **doc/** directory:
    - A .pdf file documenting the core's operations
      - This documentation is accessible through XPS if this core directory structure is followed

# EDK Project Directory Structure

---

- What's in *project\_software/*:
  - The Software project's executable (.elf file)
  - You may want to put your project's source code here, or have a separate “/code” directory

# EDK Project Directory Structure

---

- What's in the Processor Directories:
  - Each processor (e.g. Microblaze\_0) has a directory associated with it for its active software projects and hardware system
  - There are 4 subdirectories:
    - `code`
    - `include`
    - `lib`
    - `libsrc`

# EDK Project Directory Structure

---

- What's in **code/**:
  - The stub's makefile, source assembly, executable file, etc
- What's in **include/**:
  - Header files for all of the included libraries/drivers (recall your **#include** statements)
  - **PLUS** the **xparameters.h** file
    - What does the **xparameters.h** file contain?
      - Hint: You should see this in the lab.

# EDK Project Directory Structure

---

- What's in `lib/`:
  - The precompiled libraries (recall your software platform settings)
  - Includes:
    - `libc.a`
    - `libm.a`
    - `libxil.a`
- What's in `libsrc/`:
  - Recall your `.mss` file
  - This directory contains all the drivers you specified by name and version for your peripherals

---

Now let's look at how XPS uses this information  
to generate a bitstream:

---

# Tools

---

- EDK Tools we're concerned with:
  - Platform Generator
  - Library Generator
  - Compiler and Linker
  - Bitstream Initializer
  - Debugger
- ISE Tools
  - Synthesis (XST)
  - NGDBuild
  - Map
  - Place and Route (PAR)
  - Bitstream Generator
  - iMPACT

# EDK Tools:

---

- Platform Generator
  - Primary Input: MHS file
  - Also reads pcore description files including
    - MPD
    - PAO
  - Generates:
    - Top-level HDL file
    - Wrappers for all of the IP cores instantiated in the system
    - A BRAM Memory Map (BMM) File
      - Contains addresses of various on-chip BRAM memories used
      - Later used for initializing BRAMs with software
  - Invokes XST to synthesize each of the instantiated pcores

# EDK Tools:

---

- Library Generator
  - Primary Inputs: MSS file and MHS file
  - Also reads pcore sw description files including
    - Microprocessor Driver Definition File (kind of a sw version of MPD)
    - Driver code
  - These libraries are include in the:
    - EDK Instantiation libraries
    - Custom libraries (for custom peripherals)
  - Configures:
    - Libraries
    - Device Drivers
    - File Systems (part of an OS)
    - Interrupt Handlers

# EDK Tools:

---

- GNU Compiler Tools (GCC) [Compiler and Linker]
  - Compiler Primary inputs:
    - C Files
    - Header (H) Files
    - Precompiled libraries (.a files)
  - Generates:
    - Object Files
    - .a files
  - Linker Inputs:
    - Linker Script
    - Object files
    - .a files
  - Generates:
    - Executable (.elf files)

# EDK Tools:

---

- Bitstream Initializer:
  - Primary inputs:
    - Bitstream file (without .elf in BRAMs) e.g. system.bit
    - BMM file (where are the BRAMs)
    - .elf (executable to be downloaded)
  - Generates:
    - Bitstream file (includes .elf in the BRAMs) e.g. download.bit
  - Uses the Data2MEM utility from ISE to update the bitstream

# EDK Tools:

---

- Debugging Tools:
  - Debug Configuration Wizard
    - Allows you to instantiate ChipScope
    - Can provide JTAG based virtual input and output
  - Xilinx Microprocessor Debugger (XMD)
    - Can be used as a software stub
    - Can be used with the MicroBlaze Debug Module (MDM)
      - Very powerful
      - Can debug up to 8 MicroBlaze Processors at a time
  - GNU Debugger (GDB)
    - Can be used to debug software on-chip
    - Must be run in conjunction with XMD (it's the GDB server)

# Other EDK Tools:

---

- Simulation Model Generator (Simgen)
- Simulation Library Compiler (CompEDKLib)
  - Need for ModelSim
- Virtual Platform Generator (VPgen)
- Bus Functional Model Compiler (BFM)
- System ACE File Generator (GenACE)
- Flash Memory Programmer
- Format Revision (revup) Tool and Version Management Wizard
- LibXil Memory File System Generator (LibXil MFS)
- Platform Specification Utility

---

For now, just a quick overview of the ISE Tools

---

# ISE Tools used with XPS

---

- Synthesis (XST)
- NGDBuild
- Map
- Place and Route (PAR)
- Bitstream Generator
- iMPACT

# ISE Tools used with XPS

---

- Synthesis (XST)
  - Inputs: HDL Files (VHDL/Verilog)
  - Output: An NGC File (netlist file)
  - Takes HDL designs and creates Xilinx specific netlist files
    - Contains both the logical design data and constraints

# ISE Tools used with XPS

---

- NGDBuild:
  - Inputs: NGC or EDIF Files
  - Output: A Xilinx Native Generic Database (NGD) file
    - Reduced to NGD primitives along with the original hierarchy expressed in the input netlist(s)
  - Converting a netlist to NGD
    - Merge components that reference other files
    - Find appropriate system library components, physical macros (NMC files), behavioural models
    - Run a Logical Design Rule Check

# ISE Tools used with XPS

---

- Map:
  - Inputs: NGD File
  - Output: A Xilinx Native Circuit Description (NCD) file
    - Maps a logical design to a Xilinx FPGA
    - A physical representation of the design mapped to components on the target FPGA
  - Converting an NGD file to an NCD file
    - Performs a logical Design Rule Check
    - Maps design logic to components (logic cells, I/O cells, other components)
    - ***Also produces a mapping report (.mrp): LOOK AT THIS FILE***

# ISE Tools used with XPS

---

- PAR (Place and Route):
  - Inputs: A Mapped NCD File
  - Output: A Placed and Routed (NCD) file
    - A **complete** physical representation of the design mapped to **specific components and routing resources** on the target FPGA
  - This can take a **long** time for big designs
    - Timing Driven option is the default if a clock frequency is specified
    - ***Run trace after this to get a timing report and figure out the critical path (Check out the .twr file)***

# ISE Tools used with XPS

---

- Bitstream Generator (BitGen):
  - Inputs: A Placed and Routed (NCD) file
  - Output: A configuration bitstream (.bit)
    - Contains all the configuration information defining internal logic and interconnections on the FPGA
- iMPACT:
  - Input: A configuration bitstream
  - Output: A programming file (e.g. System ACE) or
  - Enables Device Configuration through different modes:
    - Includes Boundary Scan, and Slave-Serial

---

Finally...

---

# Highlights of this slide set:

---

- EDK Project Directory Structure
- EDK CAD Tools
- Brief Overview of ISE CAD Tools
  
- Now that you have a bit more background ...

# Highlights of this slide set:

---

- EDK Project Directory Structure
- EDK CAD Tools
- Brief Overview of ISE CAD Tools
  
- Now that you have a bit more background ...

Get ready for the Lab Exam/Test 1!