TOOLFLOW
Flow

- Run System Generator; generate netlist
- Copy base system package
- Take XSG result netlist, create pcore
- Instantiate XSG design in XPS project
- Instantiate interface pcores in XPS project based on yellow blocks
- Write UCF constraints
- Generate project software
- Run XPS
“Yellow Blocks”

- Wrapper for XSG Gateway Blocks
- Each GW name should contain full block hierarchy & port name
- Each type of yellow block corresponds to an interface pcore by “Tag” in Block Properties
Yellow blocks used to create Matlab objects

*xps_block* parent class

All interface types have associated child class

Each yellow block creates object instantiation
  - Stores parameters in object fields
  - Uses member functions to aid in project generation

Matlab OOP reference:

XPS_Block class reference:
http://casper.ssl.berkeley.edu/wiki/XPS_Block_Class_Reference
Building the EDK Project
gen_xps_files.m acts as primary interface to all parts of toolflow

Combs through all blocks in design and looks for \texttt{xps:*} tag

Uses tag to call appropriate class constructor to create \texttt{xps\_object} for each yellow block
Building the EDK Project

- XSG Core Config yellow block sets parameters in Xilinx System Generator Token
  - Sets part based on hardware platform
  - Output directory from design name
  - All other static setting

- gen_xps_files.m issues call to XSG Token’s Generate

- Compiled NGC netlist at fixed path in output directory
Building the EDK Project

- Checks XSGCC block for HW platform; copies appropriate base system from xps_lib (XPS_LIB_PATH environment variable)
- Creates a pcore in XPS_hw_platform_base\pcores\design_name_v1_00_a
- Copies netlist from XSG output directory
- Writes BBD to point to netlist
- Uses `gen_mpd` method of each xps_object to write MPD for pcore
Building the EDK Project

- New for ROACH: selects an XMP file based on \textit{hw_subsys}
Backup and preprocess `system.mhs`
- Used for non-core-instance-specific structures
- `elseMHSLine #IF# Matlabconditional# ifMHSLine #`

Instantiate XSG design pcore using `gen_mhs_xsg`

Initialize buses & address space usage based on base system & skeleton infrastructure

Cycles through all `xps_objects` to instantiate interface pcores
- Calls `gen_mhs` method for each `xps_object`; each instantiation increments bus address space
- Uses `probe_bus_usage` to determine bus usage
Building the EDK Project

- Backup and preprocess `system.mss`
- Cycles through all `xps_objects` to instantiate interface `pcore` drivers
  - Calls `gen_mss` method for each `xps_object`

- Backup and preprocess `system.ucf`
- Write clock & timing constraints for `usr_clk`
- Cycles through all `xps_objects` to write interface `pcore` constraints
  - Calls `gen_ucf` method for each `xps_object`
Building the EDK Project

- If using TinySH, incorporates TinySH code
- Parses custom code
  - main.c needs to be aware of all user-accessible functions, looping functions, init functions
- Modifies system.xmp to include software source in project
Running EDK on Project

- Runs EDK in command line mode to build project