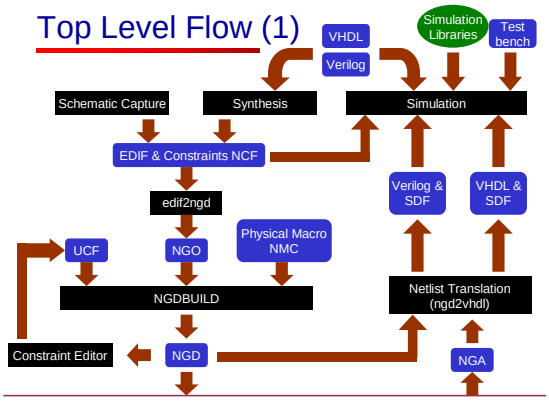
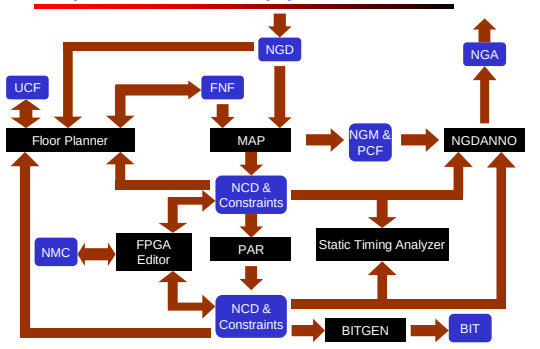
Logic Synthesis with Synplify

Top Level Flow (1)



Top Level Flow (2)



Synplify

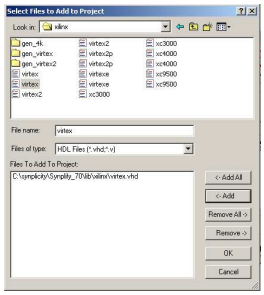
- A synthesis tool from Synplicity
- Interprets high-level HDL description
  - Converts HDL into small, high-performance, design netlists optimized for popular FPGAs
  - Writes results to standard EDIF file
- Can write post-synthesis VHDL or Verilog netlists for simulation

Supported Standards

- Synthesizable subset of VHDL93 and following packages
  - std_logic_1164
  - numeric_std
  - numeric_bit
  - std_logic_unsigned
  - std_logic_signed
  - std_logic_arith
- Synthesizable subset of Verilog95

Starting New Project (1)

- First add Virtex library file from Synplify folder

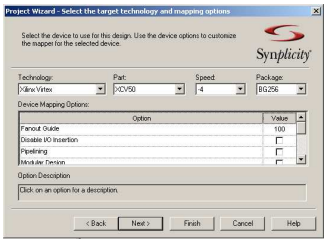


Starting New Project (2)

- Next add all other library files that are not built into Synplify
- Add your files at the end
- Order of files in the project matters and should reflect hierarchy of the design

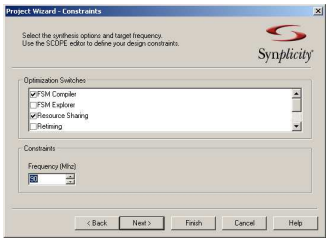
Starting New Project (3)

- Set synthesis options
  - Target technology
  - Fanout guide
  - I/O insertion
  - Pipelining
  - Modular Design
  - Retiming



Starting New Project (4)

- Set target clock frequency and other options
  - FSM Compiler & Explorer
  - Resource Sharing



Starting New Project (5)

- Choose top level entity and encoding for FSMs

Fanout Guide

- Sets the maximum limit for number of inputs driven by one output
  - Not a hard limit. To set a hard limit use syn_maxfan attribute
- Large fanouts can cause large delays and routability problems
- Low fanouts result in excessive logic replication and buffering

Disable I/O insertion

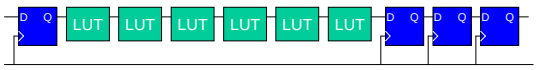
- Useful when the synthesized circuit will later be instantiated in another circuit
- If you try to implement the circuit with disabled I/O the mapper will optimize all the logic away

Pipelining (1)

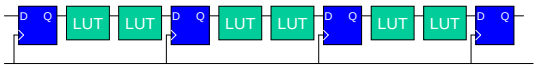
- Automatically pipelines multipliers and ROMs
- Pipeline registers must exist in the RTL code and will be pushed into the module that needs to be pipelined
  - Registers must have the same clock, set/reset (or none) and enable
- ROMs must be at least 512 words

Pipelining (2)

Before



After



Modular Design

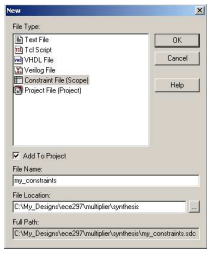
- Support for large designs developed in modules by different groups of people

Retiming

- Moves registers (register balancing) across combinational gates or LUTs ensuring identical behavior
- Does not change the number of registers in the path but may change total number of registers in the design
- Is a superset of Pipelining

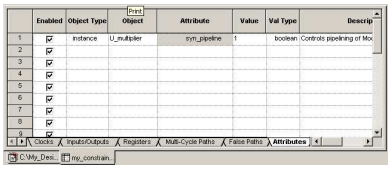
Constraints (1)

- Constraints are stored in .sdc file
- Synplify has convenient built-in editor of constraints
- If you want to edit constraints for the design you should first compile it at least once



Constraints (2)

- Constraints can address
  - Individual clocks
  - Inputs and outputs
  - Registers
  - Multicycle paths
  - False paths
  - Attributes



Attributes

- Attributes can be edited either in constraints editor or added in VHDL code
- For the list of allowed attributes refer to Synplify PRO Reference Manual

Synthesis

- After all options and constraints are set simply push the RUN button
- Results of synthesis are available to view in the form of schematics (RTL and Gate Netlists) and report file .srr
- Simulation netlists are created only when appropriate implementation options are checked
- EDIF file is generated automatically

Log File (1)

- To view log file push 

button
- Log file contains detailed information about your implementation
  - errors, warnings and notes
  - estimated performance
  - critical path(s)
  - circuit size

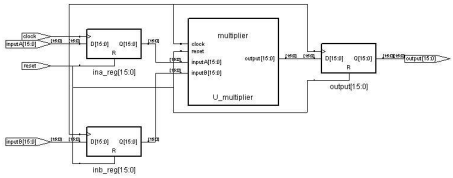
Log File (2)

- Errors, warnings and notes start with @E, @W and @N symbols
- Some of the implementation results can be viewed in log watch window

Log Parameter	rev_1
clock - Estimated Frequency	105.1 MHz
clock - Slack	1.596
multiplier_impl Total Luts	180 (11%)
multiplier_impl Register bits (Non I/O)	151 (9%)
multiplier_impl I/O Register bits	29
multiplier_impl I/O primitives	49

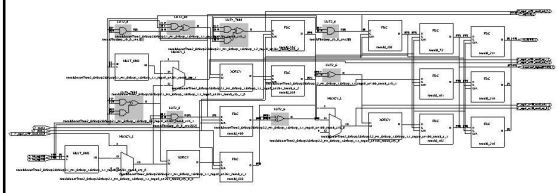
RTL Schematic

- Represents the logic in RTL level
  - Shows all identified components in easy to recognize way



Gate Schematic

- Shows details of implementation at the LUT level

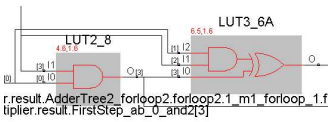


Viewing Schematics

- Operations allowed on schematics
  - Moving across hierarchy
  - Zooming
  - Flattening
  - Filtering to selected components
  - Viewing critical path

Viewing Critical Path

- Open Gate Netlist (schematic)
- Highlight critical path
- Flatten schematic
- Filter schematic



- Numbers on top of components represent timing info: delay up to the output of the component and slack in the entire path

Tips and Tricks

- NGDBulid
  - If some constraints cannot be forward annotated by Synplify add them in UCF file
- MAP
  - Do not map to 5-input functions. Do not use *–k* option at all.
- PAR
  - Do not use effort level of 5. Set it to 4 at most using *–l4* option