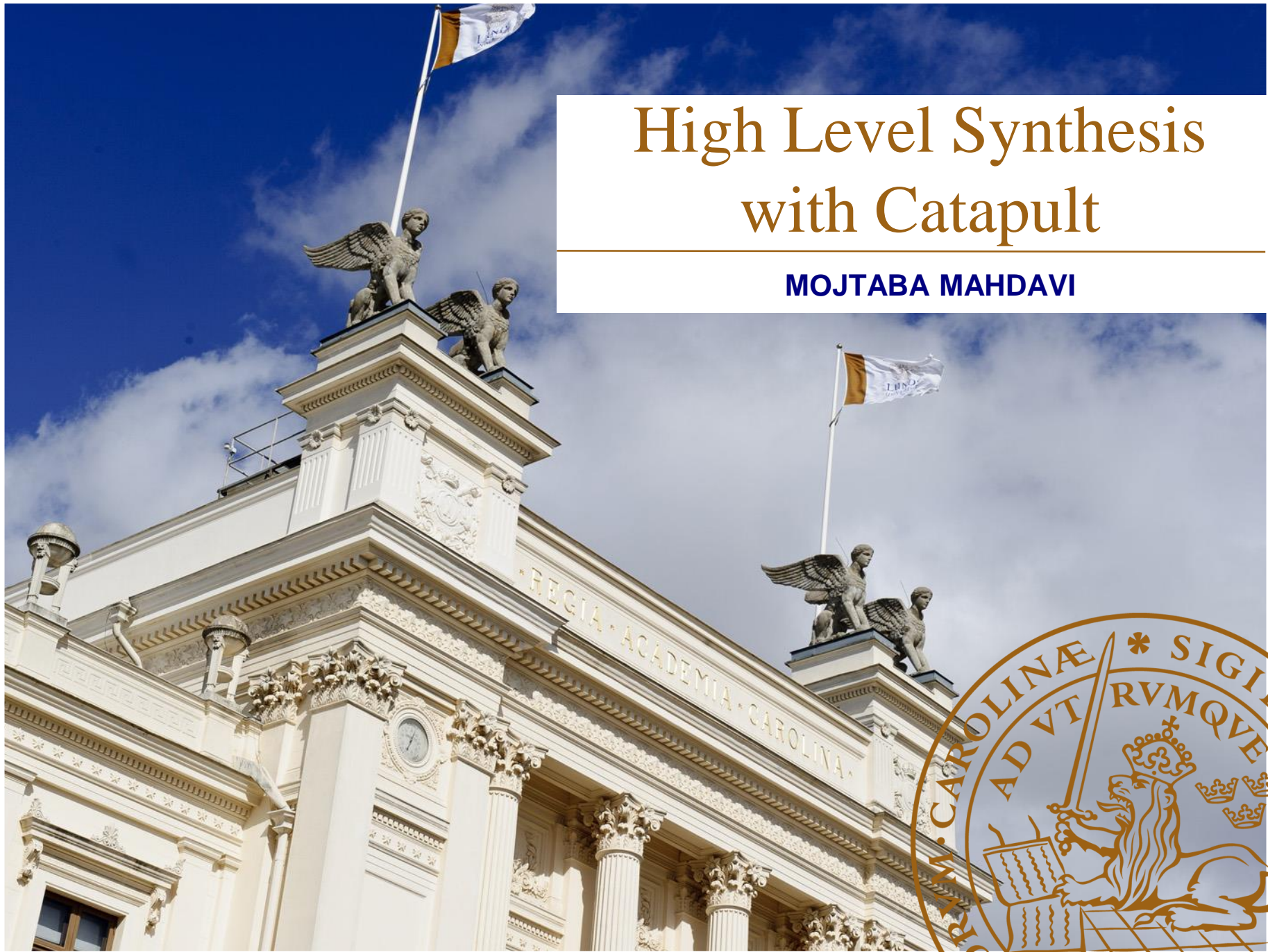


High Level Synthesis with Catapult

MOJTABA MAHDAVI



Outline

- High Level Synthesis (HLS)
- Catapult Basics
- Data Types
- Design Flow in Catapult

Outline

- ❑ High Level Synthesis (HLS)
- ❑ Catapult Basics
- ❑ Data Types
- ❑ Design Flow in Catapult

High Level Synthesis (HLS)

- ❑ Higher level of abstraction
- ❑ High-level synthesis bridges hardware and software domains.
- ❑ Generate hardware from C/C++ code or another high level language.

High Level Synthesis (HLS)

- ❑ A common misconception:
 - Synthesizing hardware from C++ provides users the freedom of describing the algorithms in any desired style of C++ coding.

- ❑ Remember that you are still describing hardware using a high level language !
 - "poor" description leads to a sub-optimal hardware.

High Level Synthesis (HLS)

- ❑ It is important to know HW concepts to get optimal results
- ❑ Successful projects require HW engineers not SW engineers
 - Hardware designers can work at a higher level of abstraction while creating high-performance hardware

High-Level Synthesis Benefits

- ❑ Develop algorithms at high level, e.g. C-level
 - Reduce design time
 - Reduce time for design changes
 - Easy design exploration
 - Faster time to market

- ❑ Verify at the higher level
 - Easier and faster verification

High-Level Synthesis Benefits

- ❑ Easy to explore different optimizations
 - Create multiple hardware implementation alternatives from the C source code using optimization directives

- ❑ Easy to maintain, easy to reuse
 - Create readable and portable C source code

Some Limitations

- ❑ Not all algorithms are suited for hardware implementations using HLS
 - Sequential code
 - Control logic
 - Large loops with function calls

Outline

- High Level Synthesis (HLS)
- Catapult Basics
- Data Types
- Design Flow in Catapult

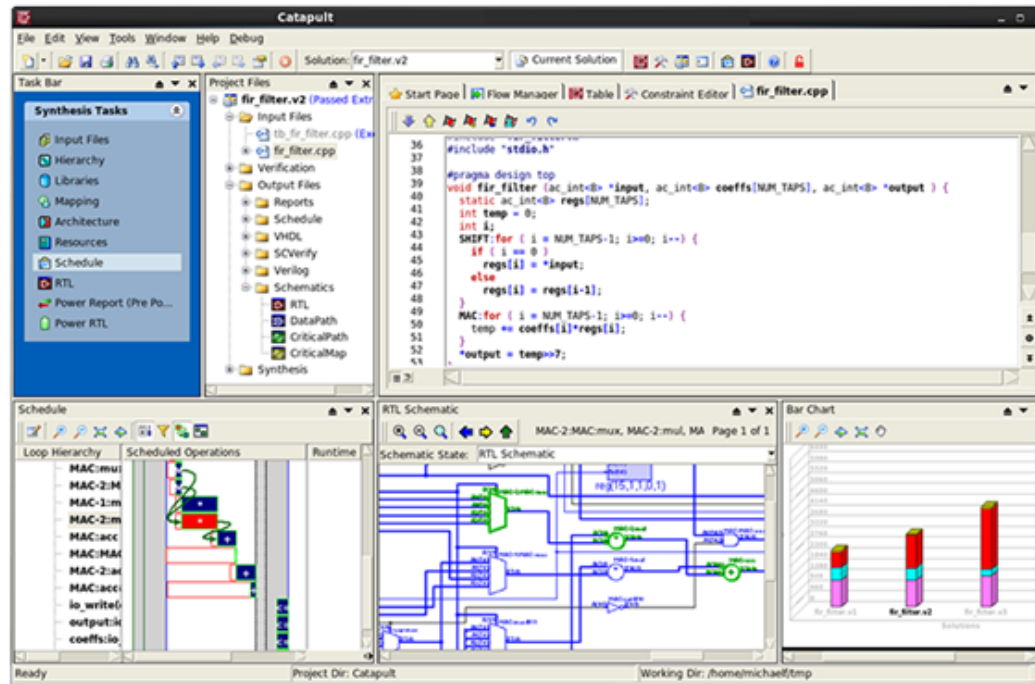
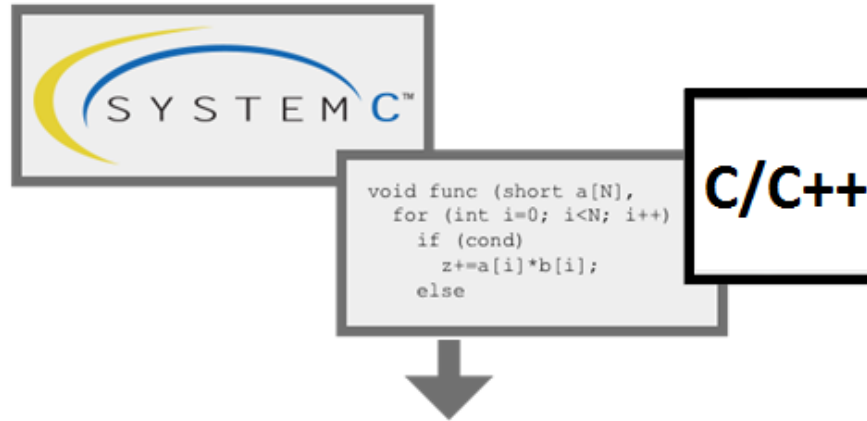
Catapult

- ❑ *Catapult* is a product of Mentor Graphics, which is used for high-level synthesis.
- ❑ Catapult takes C/C++ and System C as inputs and generates register transfer level (RTL) code.
- ❑ The target device can be FPGA and ASIC.

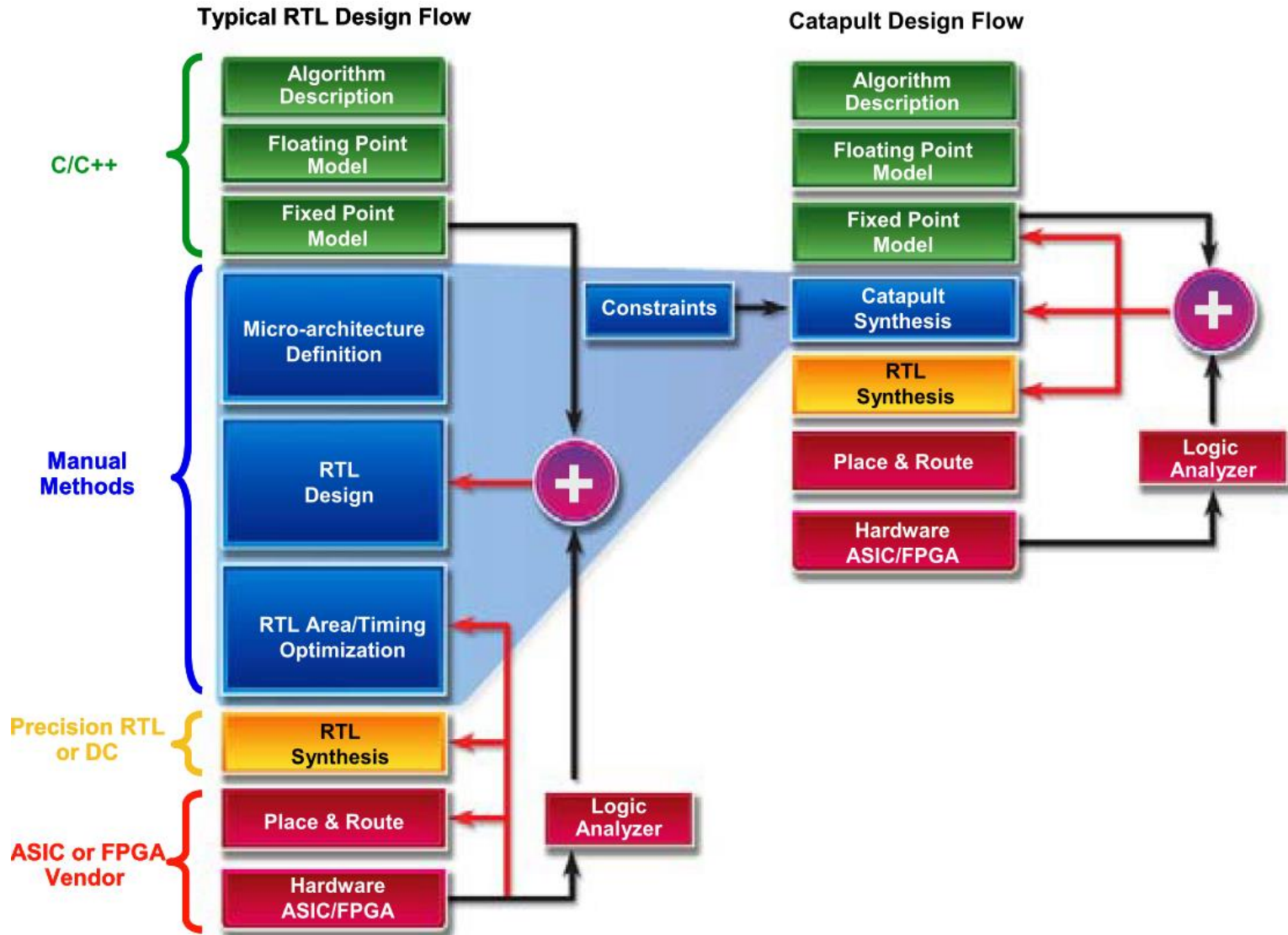
Catapult

- The concepts that you have learned in the “DSP Design” course are used in the Catapult SW
 - Folding/Unfolding
 - Re-timing/Pipelining
 - Bit-level optimization
 - and more...

Catapult

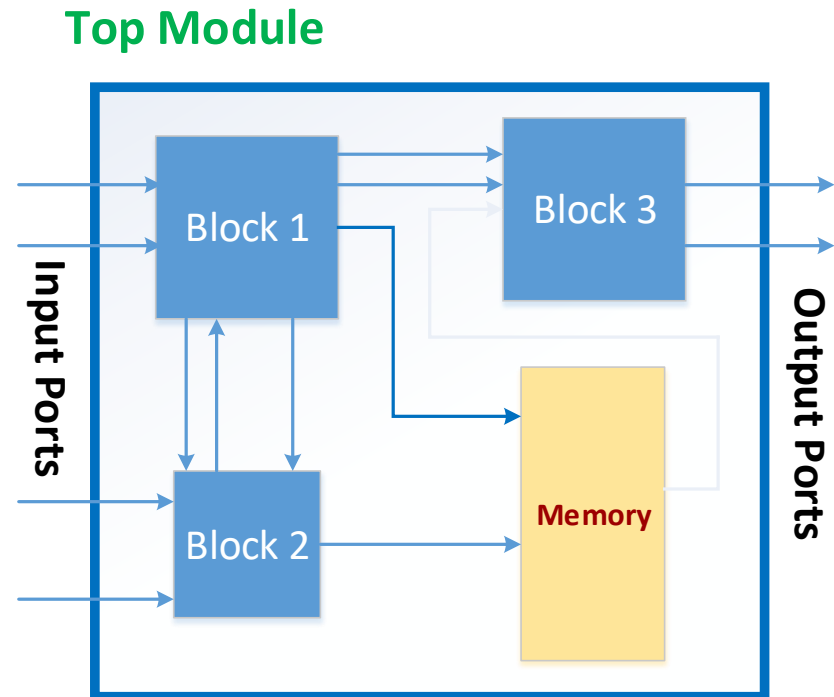


Traditional Design Flow vs. Catapult Flow



Top-level Design

- ❑ Similar to RTL designs, the "top" level of design should be specified in HLS design flow.
 - The highest level of the design
- ❑ Top module specifies:
 - The design interfaces with the outside world
 - Port definitions, direction, bit widths, and data types



Registered Outputs

- ❑ In general, high-level synthesis builds synchronous designs by default.
- ❑ All outputs of the top-level design are registered
 - To guarantee that timing is met when connecting to another design

Design Example

- Consider the following example to introduce:
 - **HLS** concept
 - corresponding steps in **Catapult** design flow

```
void accumulate(int a, int b, int c, int d, int &dout){  
    int t1,t2;  
  
    t1 = a + b;  
    t2 = t1 + c;  
    dout = t2 + d;  
}
```

Port Width and Direction

- ❑ Bit widths of the top-level ports, excluding clock and reset, are implied by the data type.
- ❑ The port direction is implied by how an interface variable is used in the C++ code.

Control Ports

- ❑ C++ code has no concept of timing.

- ❑ No clock, enable, or reset in the design source files.
 - These signals are added by the synthesis tool.

- ❑ Control over things like polarity, type of reset, etc., are taken care by setting design constraints.

Outline

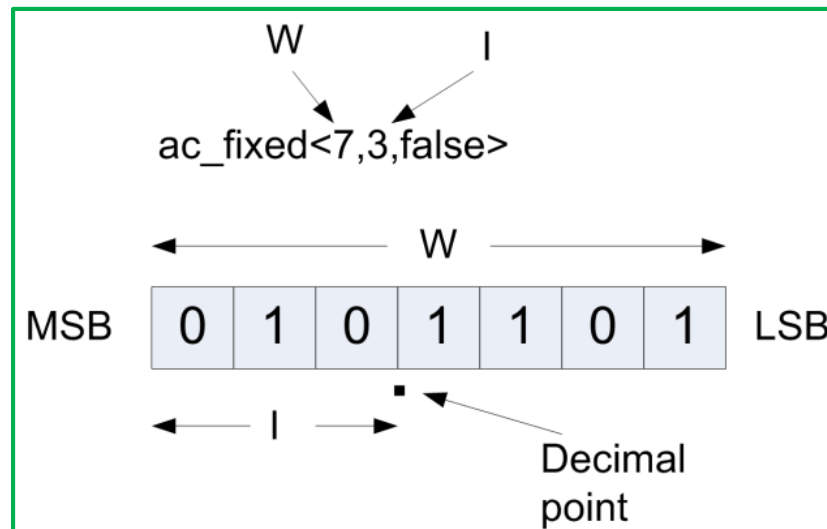
- ❑ High Level Synthesis (HLS)
- ❑ Catapult Basics
- ❑ Data Types
- ❑ Design Flow in Catapult

Data Types

- ❑ Two important data types:
 - `int`, `ac_int`
 - `ac_fixed`
- ❑ To use these data types the following header files should be included in the design source files:
 - “`ac_int.h`”
 - “`ac_fixed.h`”
- ❑ It is possible to define user data types.

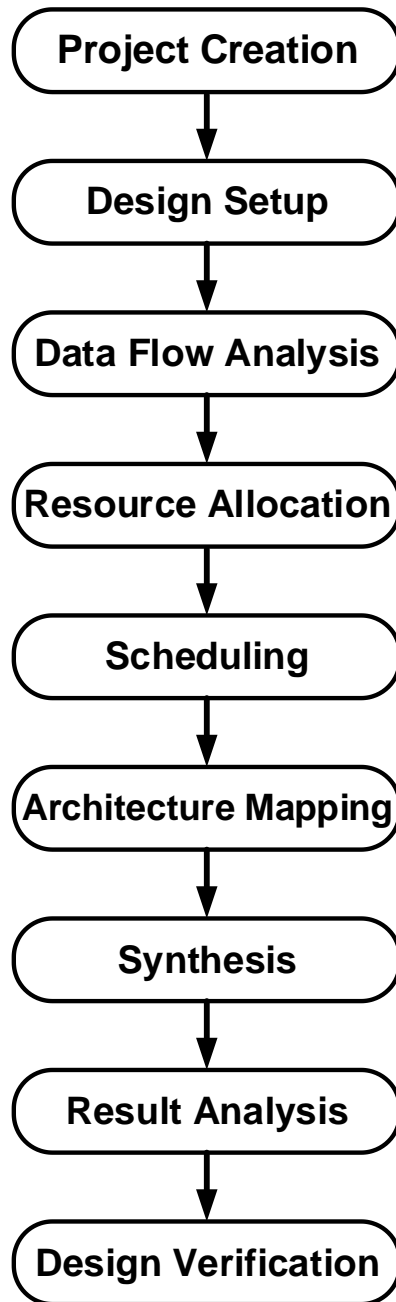
Data Types

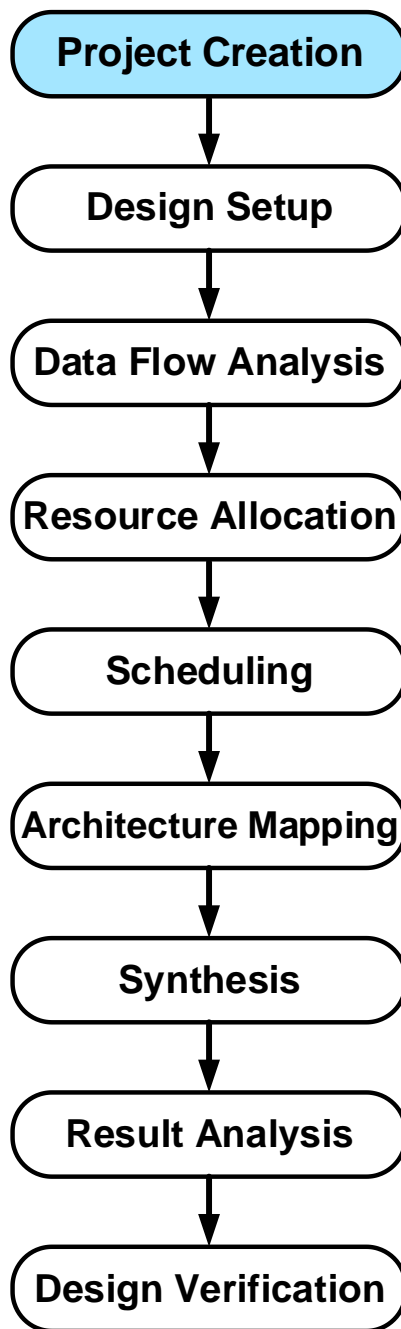
Type	Description	Numerical Range	Quantum
ac_int<W, false>	unsigned integer	0 to $2^W - 1$	1
ac_int<W, true>	signed integer	-2^{W-1} to $2^{W-1} - 1$	1
ac_fixed<W, I, false>	unsigned fixed-point	0 to $(1 - 2^{-W}) 2^I$	2^{I-W}
ac_fixed<W, I, true>	signed fixed-point	$(-0.5) 2^I$ to $(0.5 - 2^{-W}) 2^I$	2^{I-W}



Outline

- High Level Synthesis (HLS)
- Catapult Basics
- Data Types
- Design Flow in Catapult





Create a New Project

The screenshot displays the Calypto University software interface. The title bar reads "Calypto University Version 8.0/226676 (Production Release) -- Start Page". The menu bar includes "File", "View", "Tools", "Window", and "Help". The "File" menu is open, with "New Project..." highlighted. Other menu items include "New Solution...", "New File", "Set Working Directory", "Open Project", "Generate Memory", "Synthesis Tasks", and "Input Files". The "Project Files" pane shows a tree view for "solution.v1 (New)" containing "Input Files" and "Libraries". The main workspace shows the "CALYPTO | Catapult UV" logo and several panels: "Recent Projects" (New Project, Open Project), "Getting Started" (Toolkits, Application Notes, The High-Level Synthesis Blu..., Help and Documentation, Product Page), and "What's New". On the right, there are three product family logos: "Catapult® High-level Synthesis", "PowerPro® Power Optimization", and "SLEC® Formal Verification". The "Transcript" pane at the bottom shows the following text:

```
# Message
(i) // DesignPad v2.78_1.0
(i) //
(i) // This version may only be used for academic purposes. Some optimizations
(i) // are disabled, so results obtained from this version may be sub-optimal.
(i) //
# Logging session transcript to file "/usr/local-eit/cad2/mentor/cat80/Mgc_home/tmp/log15543a2176cf.0"
# Launching web browser to open http://calypto.com/en/products/catapult/overview

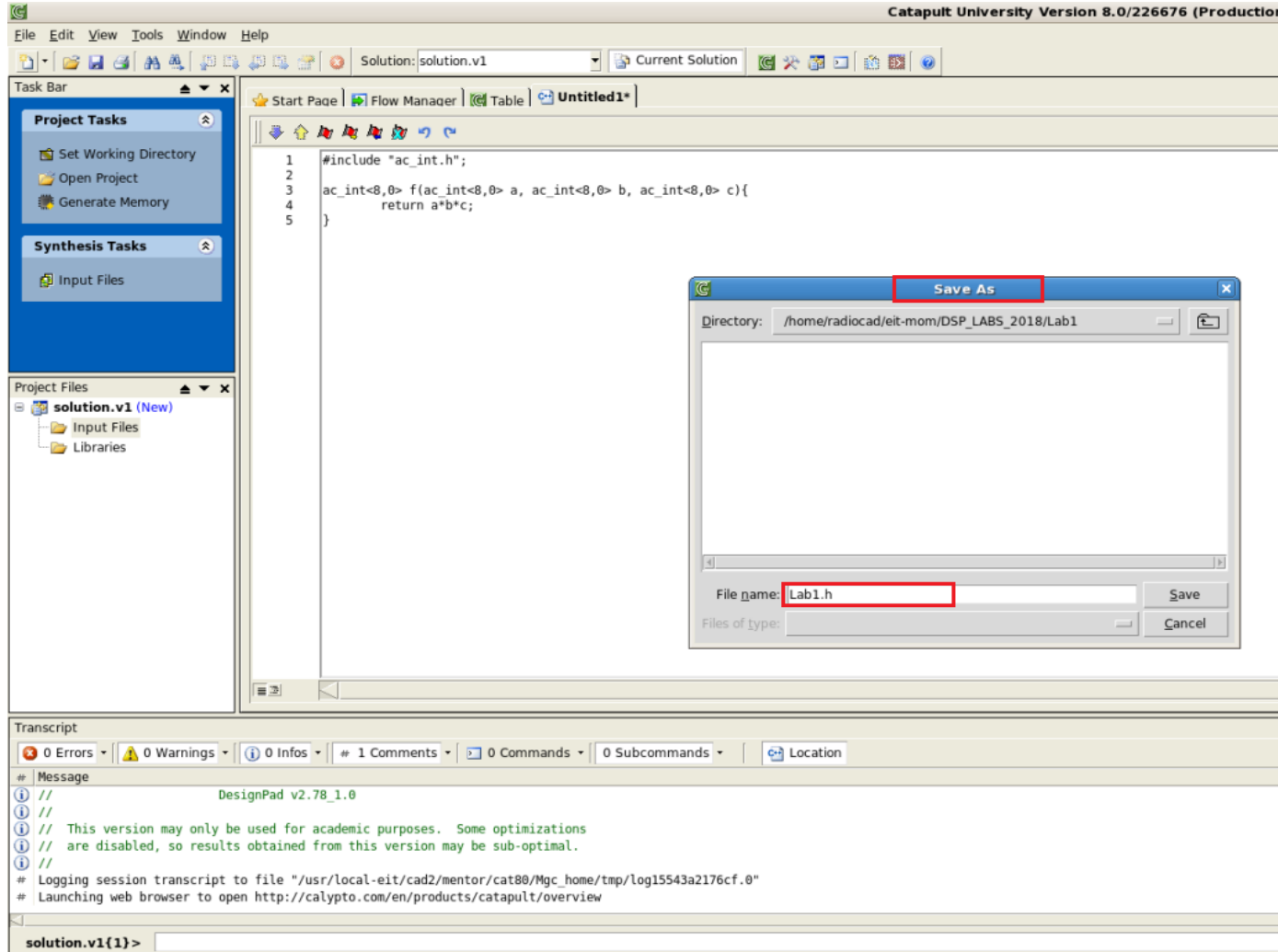
solution.v1{1}>
```

The status bar at the bottom indicates "Ready".

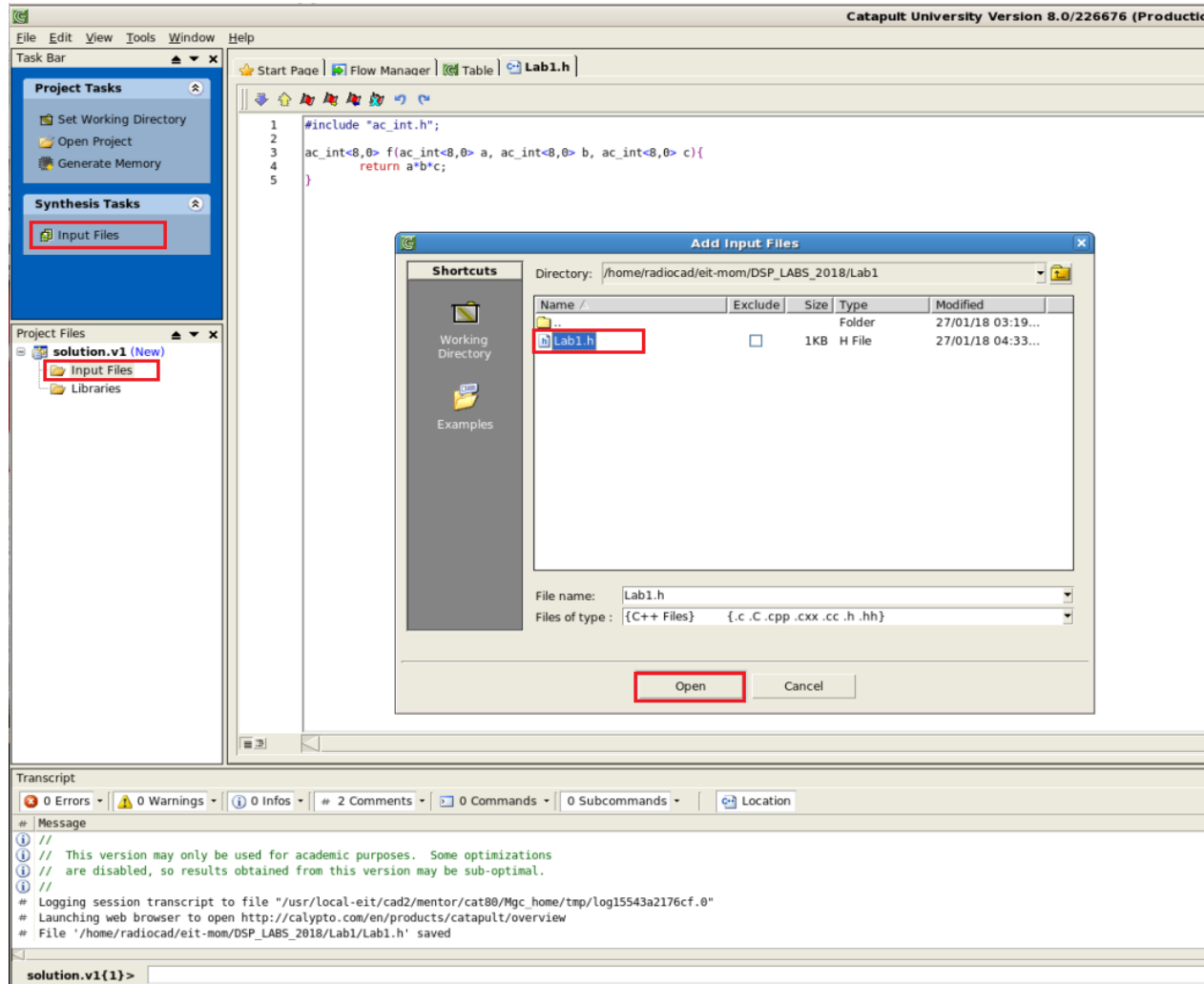
Create a New Project

The screenshot displays the Calypto UV software interface. The title bar reads "Calypto Univ (Production Release) -- Start Page". The menu bar includes "File", "View", "Tools", "Window", and "Help". The "File" menu is open, showing options like "New", "Open...", "Close", "Run Script...", "Open Project...", "Save Project", "Set Working Directory...", "Save Session Commands...", "Print", "Recent Projects", "Recent Files", and "Exit". The "New" option is highlighted with a red box, and its sub-menu is visible, showing "Project...", "Solution...", and "File" (with "Ctrl+N" shortcut). The "Recent Projects" section lists "New Project" and "Open Project". The "Getting Started" section includes "Toolkits", "Application Notes", "The High-Level Synthesis Blu...", "Help and Documentation", and "Product Page". The "What's New" section is also visible. The main workspace area displays the "CALYPTO | Catapult UV" logo and three product family buttons: "Catapult® High-level Synthesis", "PowerPro® Power Optimization", and "SLEC® Formal Verification". The bottom status bar shows "Transcript" with "0 Errors", "0 Warnings", "0 Infos", "# 2 Comments", "0 Commands", and "0 Subcommands". The transcript content includes a message about academic use, logging session transcript to a file, and launching a web browser. The command prompt shows "solution.v1{1}>" and "Ready".

Create a New Design File



Add Input Files



Set Top Module

Catapult University Version 8.0/226676 (Produ

File View Tools Window Help

Solution: solution.v1 Current Solution

Task Bar

Synthesis Tasks

- Input Files
- Hierarchy
- Libraries

Project Files

- solution.v1 (Passed Analyze)
 - Input Files
 - Lab1.h
 - ac_int.h
 - Libraries
 - Output Files

Lab1.h

```
ac_int<8, false> f(ac_int<8, false>, ac_int<
```

ac_int.h

Function: ac_int<8, false> f(ac_int<8, false>, ac_int<8, false>, ac_int<8, false>)

Hierarchy Setting

- Inline
- Block
- Top

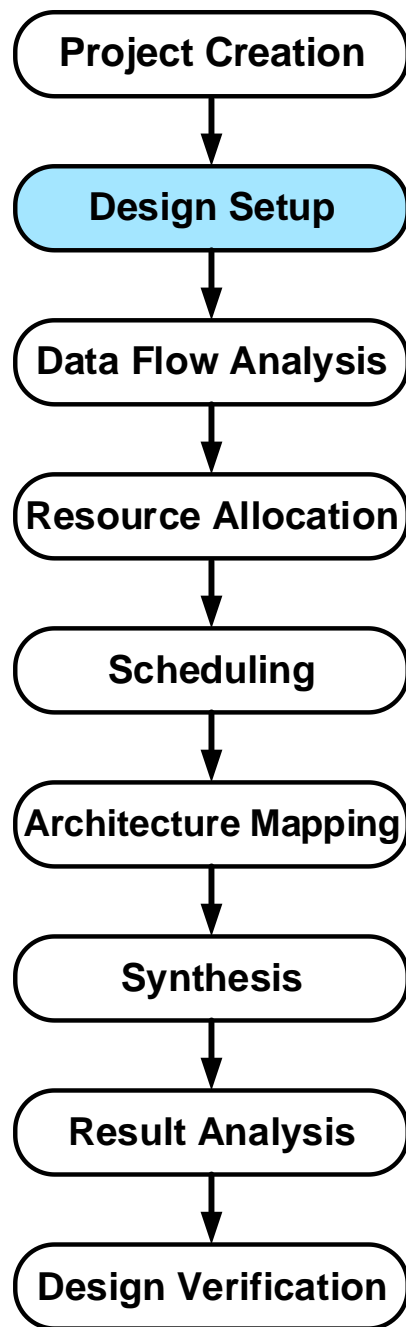
Settings

Transcript

0 Errors 2 Warnings 2 Infos 5 Comments 2 Commands 0 Subcommands Location

```
# Message
# Moving session transcript to file "/home/radiocad/eit-mom/catapult.log"
# Front End called with arguments: -- /home/radiocad/eit-mom/DSP_LABS_2018/Lab1/Lab1.h
# Edison Design Group C++/C Front End - Version 4.9
# extra text after expected end of preprocessing directive
# last line of file ends without a newline
# Source file analysis completed
# Completed transformation 'analyze' on solution 'solution.v1': elapsed time 0.86 seconds, memory usage 431968kB, peak memory usage 431968kB
```

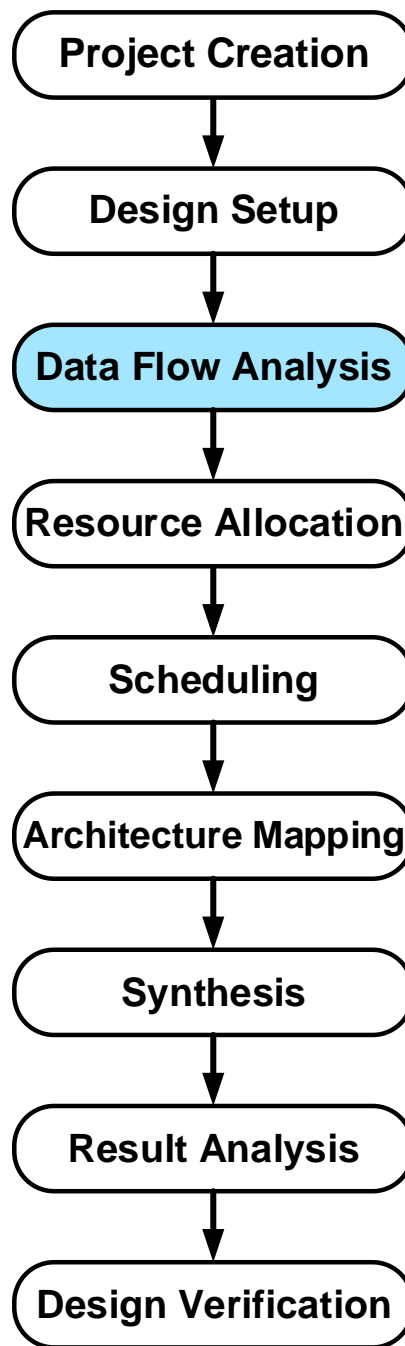
solution.v1{3}>



Design Setup

The screenshot displays the Catapult University Version 8.0/ software interface. The main window is titled "Constraint Editor" and shows the "Technology" section. The "RTL Synthesis Tool" is set to "DesignCompiler", the "Vendor" is "Sample", and the "Technology" is "090nm". The "Compatible Libraries" section is expanded, showing "Base ASIC Library" selected. The "Transcript" window at the bottom shows the following messages:

```
# Message
(i) Optimizing partition '/f/core': (Total ops = 7, Real ops = 2, Vars = 5)
(i) Optimizing partition '/f/core': (Total ops = 2, Real ops = 2, Vars = 0)
(i) Optimizing partition '/f': (Total ops = 2, Real ops = 2, Vars = 4)
(i) Optimizing partition '/f': (Total ops = 2, Real ops = 2, Vars = 4)
(i) Optimizing partition '/f/core': (Total ops = 2, Real ops = 2, Vars = 0)
# Design 'f' was read
(i) Completed transformation 'compile' on solution 'f.v1': elapsed time 0.13 seconds, memory usage 433824kB, peak memory usage 433824kB
```

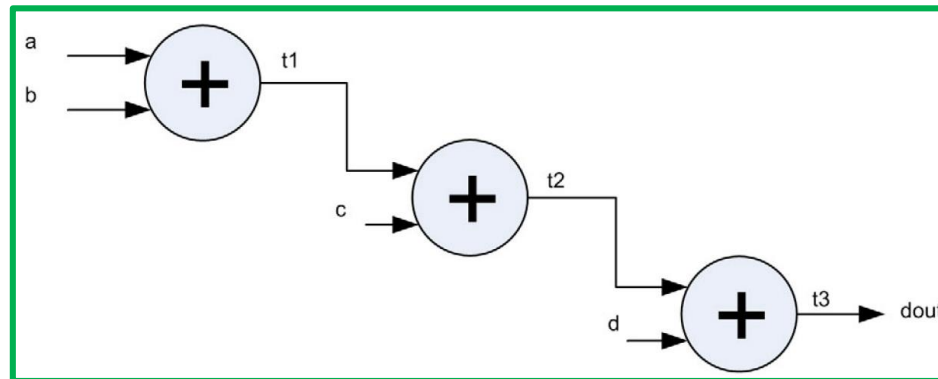



Data Flow Graph Analysis

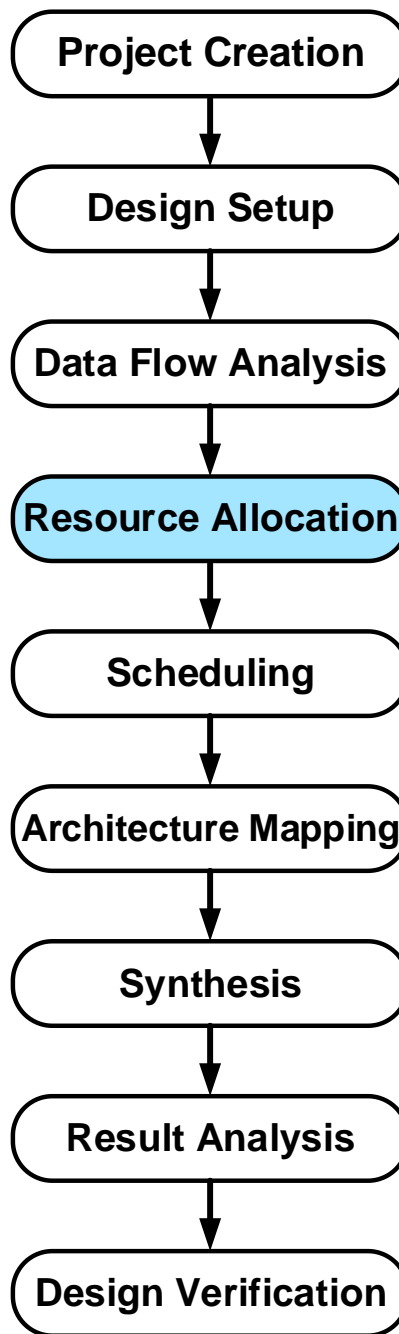
- ❑ HLS process starts by analyzing the data dependencies between the various steps in the algorithm.
- ❑ The analysis leads to a Data Flow Graph (DFG).

Design Example

```
void accumulate(int a, int b, int c, int d, int &dout){  
    int t1,t2;  
  
    t1 = a + b;  
    t2 = t1 + c;  
    dout = t2 + d;  
}
```



- ❑ Data dependencies and the order of operations are specified by the connection between nodes.



Resource Allocation

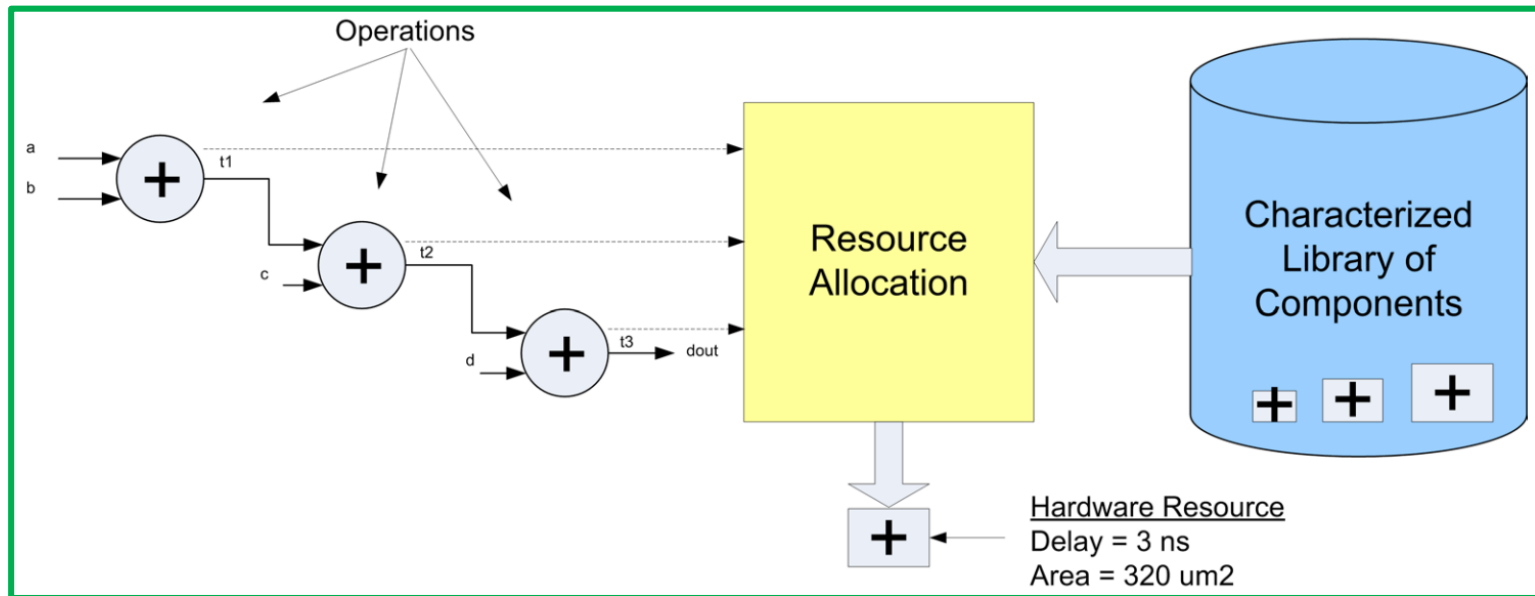
- ❑ Each operation in the extracted DFG is mapped onto a hardware resource.
 - Each resource corresponds to a physical implementation of the operator in hardware.

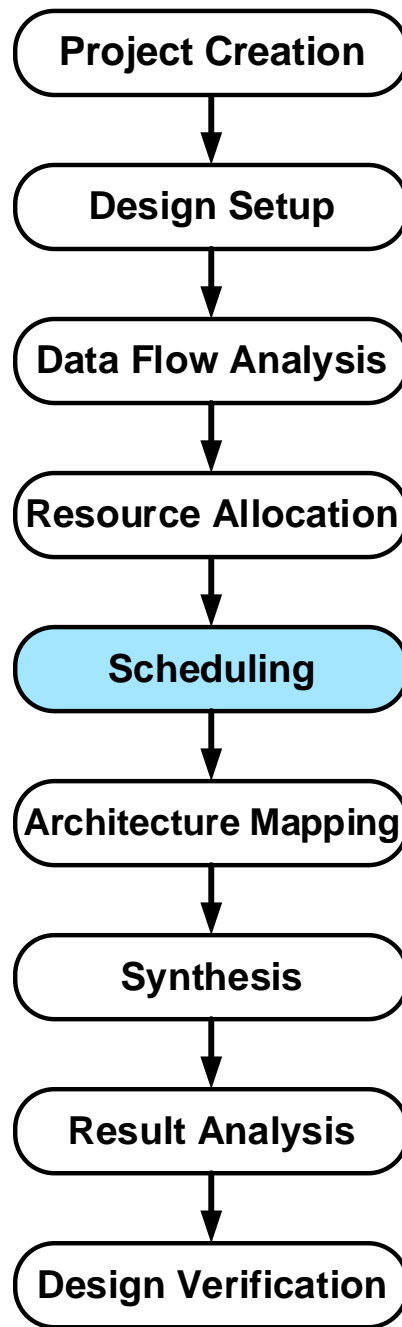
- ❑ Some operators may have multiple hardware resource implementations
 - Different area, delay, and latency specifications.

- ❑ Timing and area constraints will be applied to this implementation.

Resource Allocation

- All resources are selected from a technology specific pre-characterized library
 - Library is already determined in “Design Setup” step





Scheduling

- ❑ HLS adds "time" to the design during the "scheduling" step.

- ❑ Scheduling determines that each operation in the DFG should be performed in which clock cycle.
 - So, based on a target clock frequency, registers should be added between operations.

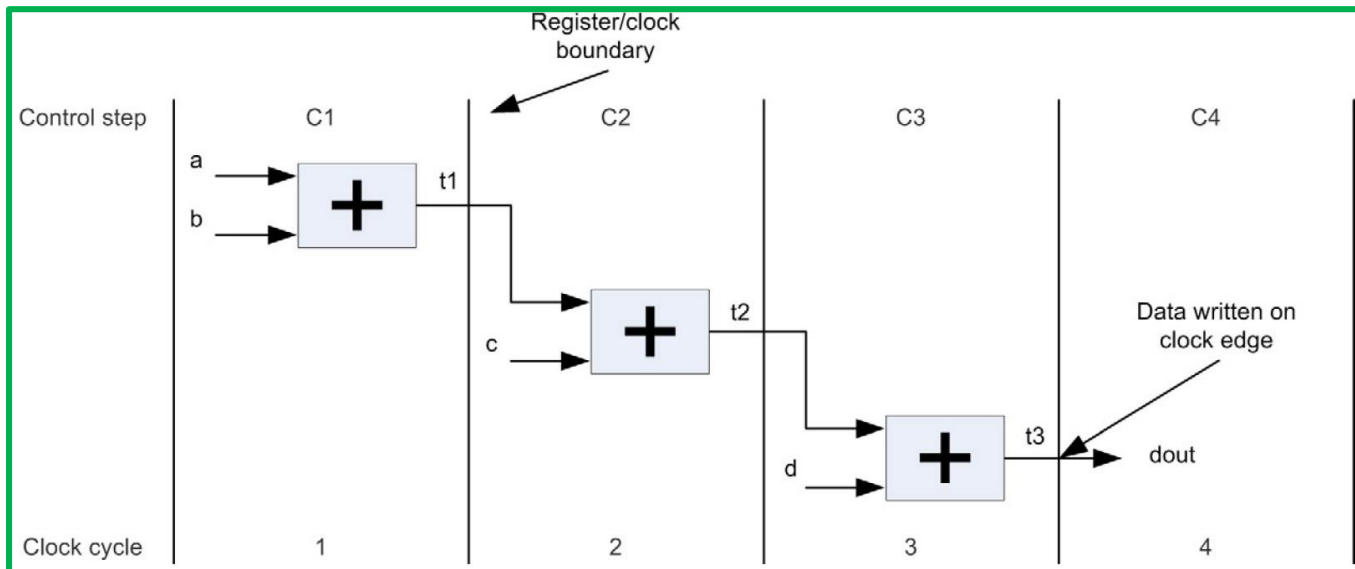
Scheduling

- ❑ This process is similar to technique what RTL designers would call **pipelining**
 - Inserting registers to reduce combinational delays

- ❑ Note that this is not the same as "**loop pipelining**" which is discussed later.

Scheduling

- ❑ Consider that in our design example:
 - "add" operation takes 3 ns
 - Clock period equals 5 ns
- ❑ Each add operation is scheduled in its own clock cycle C1, C2, and C3
 - Registers are inserted automatically between each adder.



Timing Constraints

The screenshot displays the Catapult University software interface, version 8.0/226676. The main window is titled "Constraint Editor" and shows the "Process: core" settings. The "Frequency" field is set to 50 MHz, which is highlighted with a red box. The "Period" is 20.0000 ns, "High Time" is 10.00 ns, "Offset" is 0.000000 ns, "Edge" is set to "rising", and "Uncertainty" is 0.0 ns. A timing diagram shows a clock signal "clk" with a period of 20 ns. The "Apply" button is also highlighted with a red box. The "Synthesis Tasks" panel on the left shows "Mapping" selected. The "Project Files" panel shows the project structure for "f.v1". The "Transcript" panel at the bottom shows the output of the synthesis process, including warnings and errors.

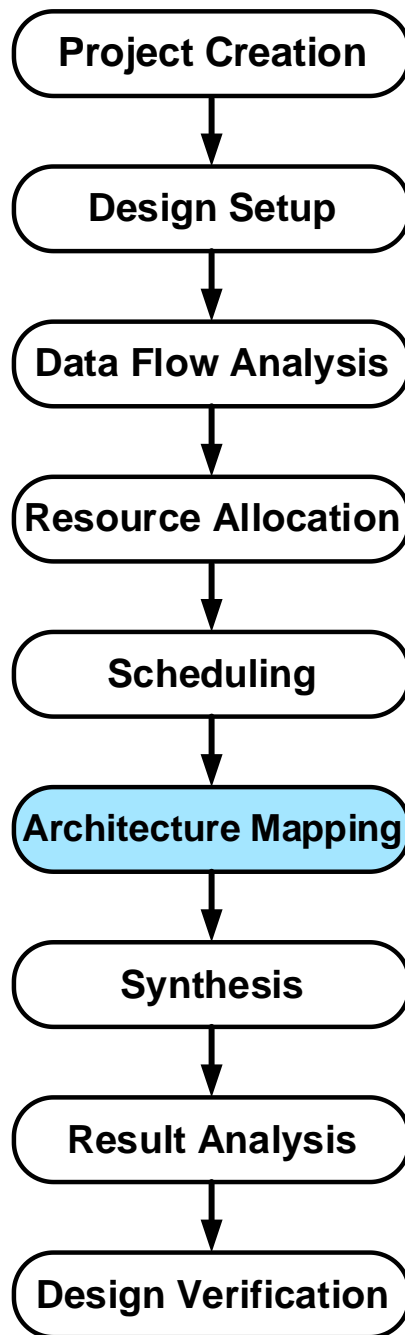
0 Errors | 4 Warnings | 20 Infos | 18 Comments | 6 Commands | 0 Subcommands | Location

#	Message	File(line)	Id
#	Reading component library 'SMGC_HOME/pkg/siflibs/mgc_ioport.lib' [mgc_ioport]...		LIB-49
#	Reading component library 'SMGC_HOME/pkg/cds_assert/assert_ops.lib' [ASSERT_OPS]...		LIB-49
#	Reading component library 'SMGC_HOME/pkg/cds_assert/assert_mods.lib' [assert_mods]...		LIB-49
#	Reading component library 'SMGC_HOME/pkg/siflibs/designcompiler/mgc_sample-090nm_beh_dc.lib' [mgc_sample-090nm_beh_dc]...		LIB-49
!	Could not read liberty file 'sample_090nm.lib' specified in a catapult library in the paths specified ()		LIB-147
!	Downstream flows that depend on a liberty file(s) will not be able to run. Please check option ComponentLibs/TechLibSearchPath.		LIB-149
i	Completed transformation 'libraries' on solution 'f.v1': elapsed time 0.29 seconds, memory usage 438388kB, peak memory usage 438388kB		SOL-9

f.v1{7}>

Ready

Terminal | Catapult University Version



Hardware Implementation

- The corresponding hardware that is generated from the schedule varies depending on how the design is constrained in terms of:
 - Resource allocation
 - The amount of loop pipelining

Loop Pipelining

- ❑ The top-level function call has an implied loop, also known as the main loop.
 - Each iteration of the implied loop corresponds to execution of the schedule

- ❑ "Loop Pipelining" allows a new iteration of a loop to be started before the current iteration has finished.
 - Execution of the loop iterations to be overlapped
 - Increasing the design performance by running loops in parallel

Initiation Interval (II)

- The amount of overlap between the loops is determined by the "Initiation Interval (II)".
 - Specifies the number of pipeline stages

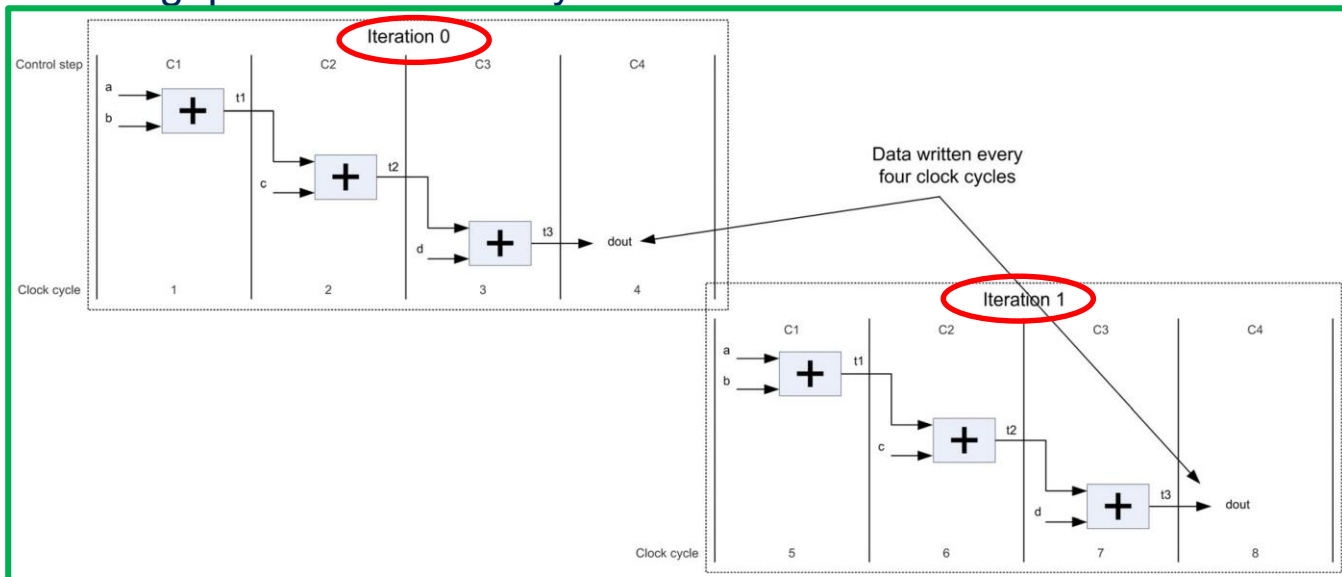
Case1: No Pipelining

❑ Design is left unconstrained

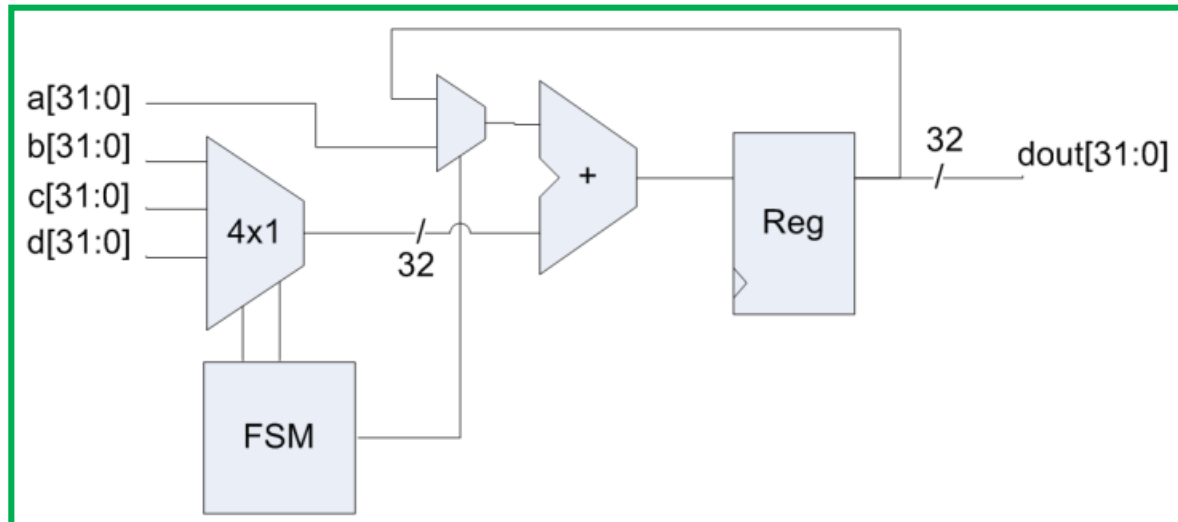
- There is only a single pipeline stage
- No overlap between execution of each iteration of main loop

❑ Data will be written every four clock cycles

- Latency of three clock cycles
- Throughput of four clock cycles



Case1: No Pipelining



- There is no overlap between any operation.
- Resulting hardware uses a single adder to accumulate a , b , c , and d .
- Reduction in the overall area.

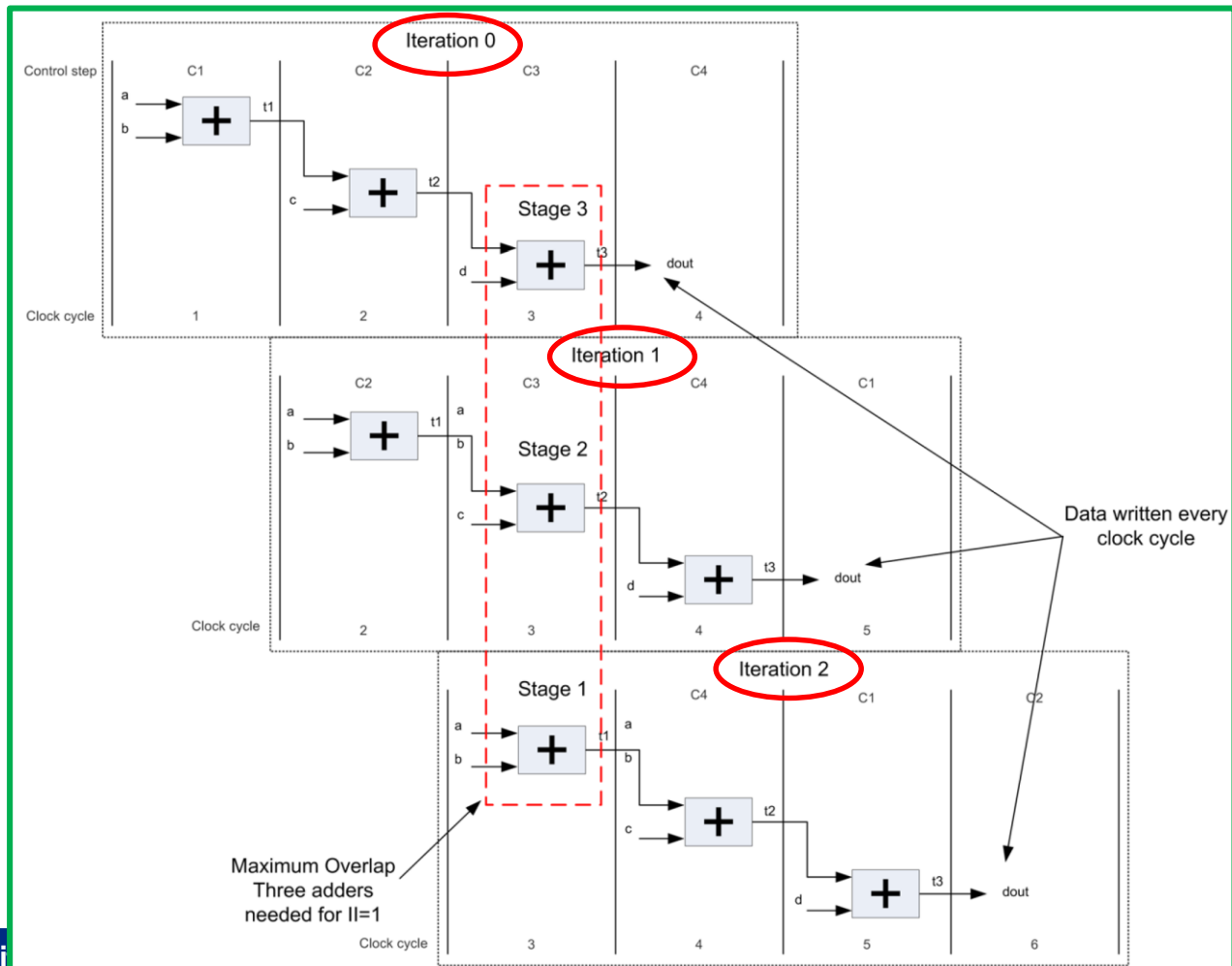
Case 2: Pipelining with $II = 1$

- ❑ Pipelining with an $II=1$ results in a new iteration started every clock cycle.
 - Latency of 3 clock cycles
 - Throughput of 1

- ❑ Three adders are required in hardware since all three pipeline stages can be active at the same time.
 - Larger area

Case 2: Pipelining with $II = 1$

- Iteration one is started in C2 and iteration 2 is started in C3.



Architecture Specification

The screenshot displays the Catapult Constraint Editor interface. The 'Synthesis Tasks' panel on the left has 'Architecture' selected. The 'Instance Hierarchy' and 'Module' panels show a hierarchy starting with 'f', which contains an 'Interface' module, which in turn contains a 'core' module. The 'core' module contains a 'main' instance with a multiplicity of 11=1. The 'Loop: main' configuration panel on the right shows the following settings:

- Iteration Count: (empty)
- Unroll
- Partial: (empty)
- Pipeline
- Initiation Interval: 1
- Generate distributed pipeline
- Loop can be Merged

The 'Settings' section at the bottom of the loop configuration shows 'Clustering' is selected. The 'Apply' button is highlighted. The 'Transcript' panel at the bottom shows the following output:

```
# Message
# directive set -CLOCKES {clk {-CLOCK_PERIOD 20.0000 -CLOCK_EDGE rising -CLOCK_HIGH_TIME 10.00 -CLOCK_OFFSET 0.000000 -CLOCK_UNCERTAINTY 0.0 -RESET_KIND sync -RESET_SYNC_NAME rst -RESET_SYNC_ACTIVE high -I
# /CLOCKES {clk {-CLOCK_PERIOD 20.0 -CLOCK_EDGE rising -CLOCK_UNCERTAINTY 0.0 -CLOCK_HIGH_TIME 10.0 -RESET_SYNC_NAME rst -RESET_ASYNC_NAME arst_n -RESET_KIND sync -RESET_SYNC_ACTIVE high -RESET_ASYNC_ACT
go assembly
(i) Starting transformation 'assembly' on solution 'f.v1'
(i) Optimizing partition '/f': (Total ops = 4, Real ops = 4, Vars = 6)
(i) Optimizing partition '/f/core': (Total ops = 3, Real ops = 3, Vars = 0)
(i) Completed transformation 'assembly' on solution 'f.v1': elapsed time 0.01 seconds, memory usage 438612kB, peak memory usage 438612kB
```

Design Schedule

The screenshot displays the Catapult University software interface for a design schedule. The main window is titled "Schedule" and shows a "Loop Hierarchy" on the left and "Scheduled Operations" in the center. The "Loop Hierarchy" lists the following operations:

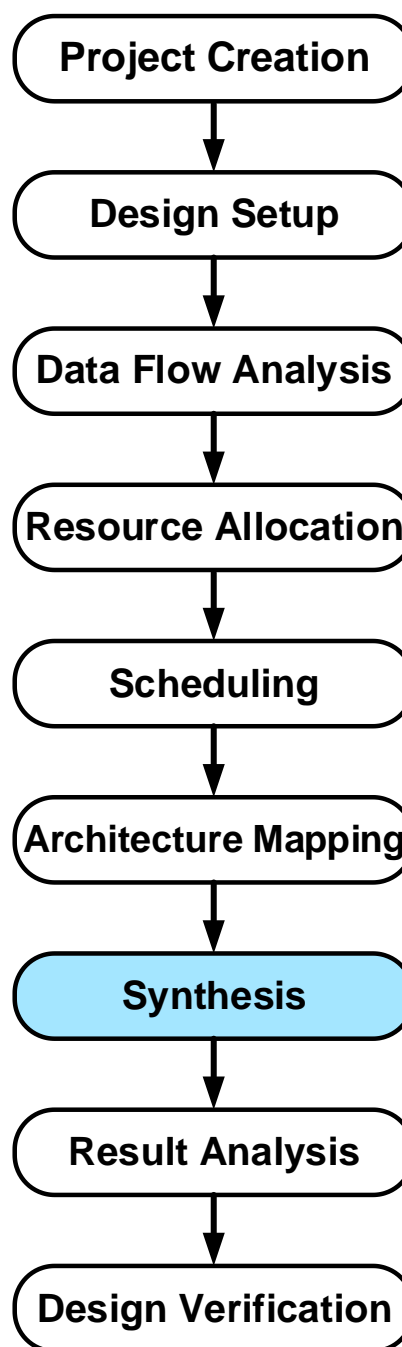
- core:rlp (core)
 - main
 - io_read(a:rsc.@)
 - io_read(b:rsc.@)
 - mul#1
 - io_read(c:rsc.@)
 - mul
 - io_write(return:rsc.@)
 - return:io_sync(return:rsc.@)
 - c:io_sync(c:rsc.triosync)
 - b:io_sync(b:rsc.triosync)
 - a:io_sync(a:rsc.triosync)

The "Scheduled Operations" window shows a Gantt chart with three columns labeled C0, C1, and C2. The chart displays the timing of various operations, with a red box highlighting a specific section. The "Runtime Profile" window on the right shows a green bar representing the execution time.

The "Transcript" window at the bottom provides the following information:

- 0 Errors, 8 Warnings, 42 Infos, 33 Comments, 10 Commands, 0 Subcommands
- Optimized LOOP '/f/core/main': Latency = 1, Area (Datapath, Register, Total) = 2367.75, 0.00, 2367.75
- Optimized LOOP '/f/core/main': Latency = 1, Area (Datapath, Register, Total) = 2278.62, 0.00, 2278.62
- Final schedule of SEQUENTIAL '/f/core': Latency = 1, Area (Datapath, Register, Total) = 2278.62, 0.00, 2278.62
- Resource allocation and scheduling done.
- Netlist written to file 'schedule.gnt'
- Extrapolation detected. Script '/home/radiocad/eit-nom/Catapult/f.v1/adjust_char_library.tcl' generated.
- Completed transformation 'allocate' on solution 'f.v1': elapsed time 0.11 seconds, memory usage 438612kB, peak memory usage 438612kB





Design Synthesis

The screenshot displays the Catapult Univer software interface. The main window shows the 'RTL' synthesis task selected in the Task Bar. The Loop Hierarchy tree on the left lists the core and main components. The Scheduled Operations diagram in the center shows the execution flow of various operations. The Transcript window at the bottom provides a log of the synthesis process, including the completion of concatenated simulation files.

Transcript

#	Message	File(line)	Id
#	Finished writing concatenated file: /home/radiocad/eit-mom/Catapult/f.v5/concat_rtl.v		
#	order file name is: rtl.v_order_sim.txt		
#	Add dependent file: /usr/local-eit/cad2/mentor/cat80/Mgc_home/pkgs/siflibs/mgc_in_wire_v1.v		
#	Add dependent file: /usr/local-eit/cad2/mentor/cat80/Mgc_home/pkgs/siflibs/mgc_out_stdreg_v1.v		
#	Add dependent file: /usr/local-eit/cad2/mentor/cat80/Mgc_home/pkgs/siflibs/mgc_io_sync_v1.v		
#	Add dependent file: ./rtl.v		
#	Finished writing concatenated simulation file: /home/radiocad/eit-mom/Catapult/f.v5/concat_sim_rtl.v		

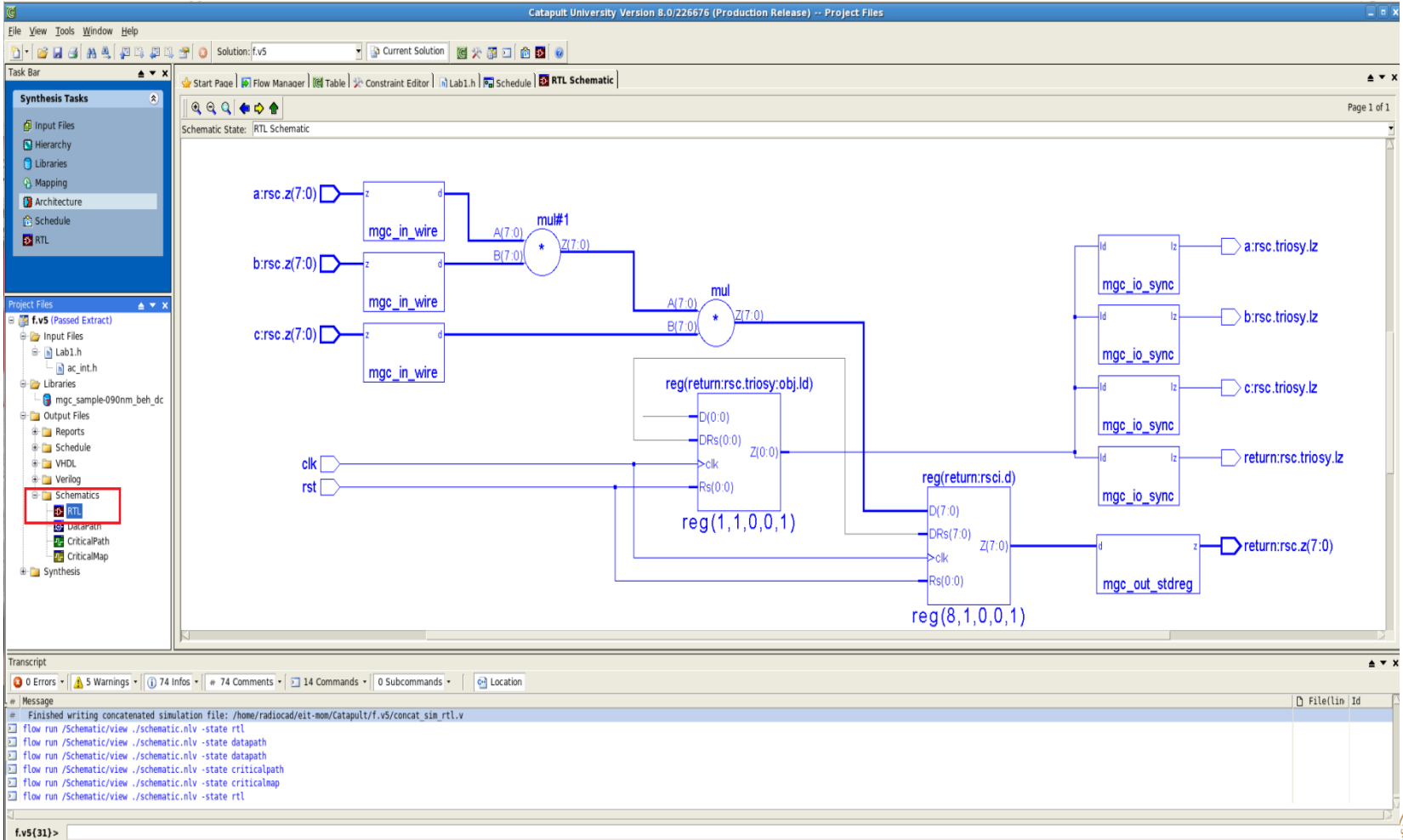
Design Schematic

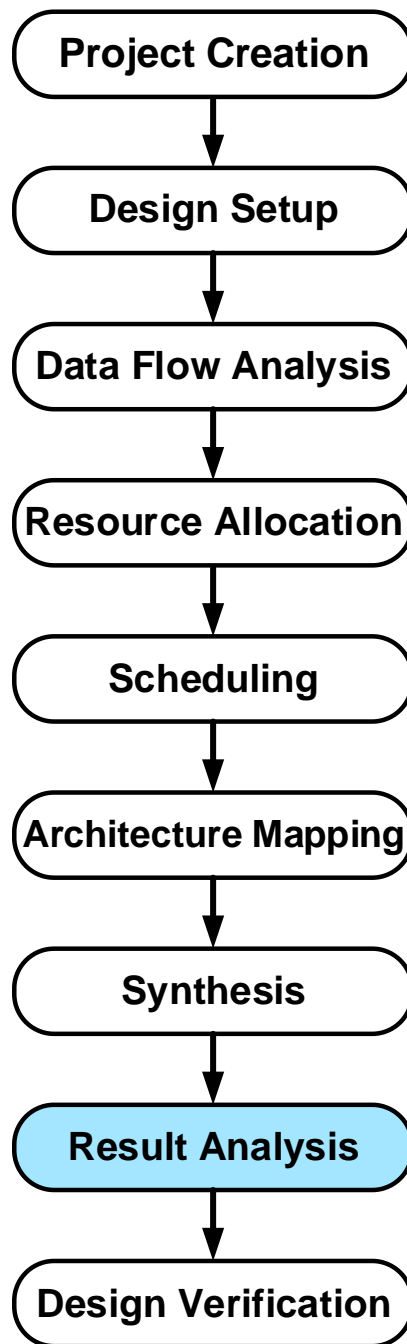
The screenshot displays the Catapult University software interface. The main window shows the RTL Schematic for a core named 'f.core'. The schematic consists of a central block labeled 'f.core:inst' with several inputs and outputs. The inputs are 'a:rsc.z(7:0)', 'b:rsc.z(7:0)', 'c:rsc.z(7:0)', 'clk', and 'rst'. The outputs are 'a:rsc.triosy.lz', 'b:rsc.triosy.lz', 'c:rsc.triosy.lz', 'return:rsc.z(7:0)', and 'return:rsc.triosy.lz'. The software interface includes a menu bar (File, View, Tools, Window, Help), a task bar with 'RTL Schematic' selected, a project files tree on the left, and a transcript window at the bottom.

Transcript:

```
0 Errors | 5 Warnings | 74 Infos | 9 Comments | 0 Subcommands | Location
Message
# order file name is: rtl.v order sim.txt
# Add dependent file: /usr/local-eit/cad2/mentor/cat80/Mgc_home/pkgs/siflibs/mgc_in_wire.v1.v
# Add dependent file: /usr/local-eit/cad2/mentor/cat80/Mgc_home/pkgs/siflibs/mgc_out_stdreg.v1.v
# Add dependent file: /usr/local-eit/cad2/mentor/cat80/Mgc_home/pkgs/siflibs/mgc_io_sync.v1.v
# Add dependent file: ./rtl.v
# Finished writing concatenated simulation file: /home/radiocad/eit-nom/Catapult/f.v5/concat_sim_rtl.v
flow run /Schematic/view ./schematic.nlv -state rtl
f.v5(26)>
```


Design Schematic





Result Analysis

The screenshot displays the Catapult University software interface for project f.v5. The main window shows synthesis results, including Area Scores, Register-to-Variable Mappings, and a Timing Report. The Project Files tree on the left shows the project structure, with 'Output Files', 'Reports', and 'RTL' highlighted. The Transcript window at the bottom shows the execution of various synthesis commands.

Area Scores

	Post-Scheduling	Post-DP & FSM	Post-Assignment
Total Area Score:	2420.3	2438.0	2438.0
Total Reg:	141.7 (6%)	159.4 (7%)	159.4 (7%)

DataPath:

	2420.3 (100%)	2438.0 (100%)	2438.0 (100%)
MUX:	0.0	0.0	0.0
FUNC:	2278.6 (94%)	2278.6 (93%)	2278.6 (93%)
LOGIC:	0.0	0.0	0.0
BUFFER:	0.0	0.0	0.0
MEM:	0.0	0.0	0.0
ROM:	0.0	0.0	0.0
REG:	141.7 (6%)	159.4 (7%)	159.4 (7%)

FSM:

	0.0	0.0	0.0
FSM-REG:	0.0	0.0	0.0
FSM-COMB:	0.0	0.0	0.0

Register-to-Variable Mappings

Register	Size(bits)	Gated Register	CG Opt	Done	Variables
return:rsci.d	8				return:rsci.d
reg(return:rsc.triosy:obj.ld).cse	1				reg(return:rsc.triosy:obj.ld).cse
Total:	9	0			0 (Total Gating Ratio: 0.00, CG Opt Gating Ratio: 0.00)

Timing Report

Critical Path

Path	Startpoint	Endpoint	Delay	Slack
1	f:core/a:rsc.z	f:core/reg(return:rsci.d)	2.494066	17.50600

Instance Component Delta Delay

Instance	Component	Delta	Delay
f:core/a:rsc.z		0.0000	0.0000
f:core/a:rsci	mgc_in_wire_v1_1_8	0.0000	0.0000
f:core/a:rsci.d		0.0000	0.0000
f:core/mul#1	mgc_mul_8_0_8_0_8_4	1.2470	1.2470

Transcript

```
f.v5(31)>
# Message
# Finished writing concatenated simulation file: /home/radiocad/eit-mom/Catapult/f.v5/concat_sim_rtl.v
# flow run /Schematic/view ./schematic.nlv -state rtl
# flow run /Schematic/view ./schematic.nlv -state datapath
# flow run /Schematic/view ./schematic.nlv -state datapath
# flow run /Schematic/view ./schematic.nlv -state criticalpath
# flow run /Schematic/view ./schematic.nlv -state criticalmap
# flow run /Schematic/view ./schematic.nlv -state rtl
```



HDL Generation

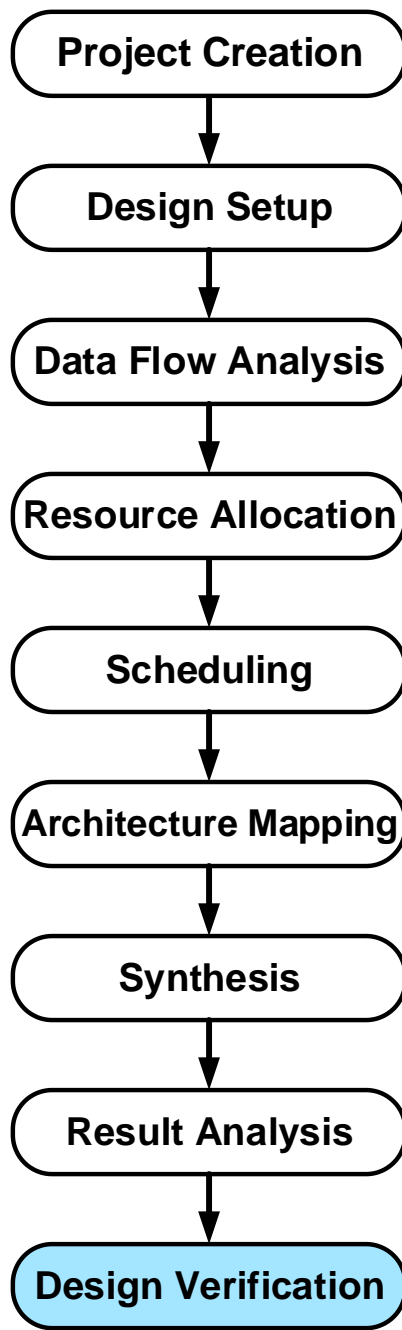
The screenshot displays the Catapult University software interface. The main window shows the VHDL code for an entity named `f_core`. The code includes library declarations for IEEE and mgc_hls, followed by the entity definition and its architecture. The architecture is named `v5` and contains several signal declarations and a process block.

```
1
2 -- HLS HDL:      VHDL Netlist
3 -- HLS Version:  8.0/226676 Production Release
4 -- HLS Date:     Tue Sep 30 16:38:21 PDT 2014
5
6 --
7 -- Generated by: eit-mom@popov
8 -- Generated date: Sat Jan 27 19:16:10 2018
9
10
11
12 -- Design Unit:  f_core
13
14
15 LIBRARY IEEE;
16 USE IEEE.STD_LOGIC_1164.ALL;
17 USE IEEE.STD_LOGIC_ARITH.ALL;
18 LIBRARY mgc_hls;
19 USE mgc_hls.funcs.ALL;
20 USE mgc_hls.mgc_ioport_comps_v3.ALL;
21
22
23 ENTITY f_core IS
24 PORT (
25   a_rsc_z : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
26   a_rsc_triosy_lz : OUT STD_LOGIC;
27   b_rsc_z : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
28   b_rsc_triosy_lz : OUT STD_LOGIC;
29   c_rsc_z : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
30   c_rsc_triosy_lz : OUT STD_LOGIC;
31   return_rsc_z : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
32   return_rsc_triosy_lz : OUT STD_LOGIC;
33   clk : IN STD_LOGIC;
34   rst : IN STD_LOGIC
35 );
36
37 END f_core;
38 ARCHITECTURE v5 OF f_core IS
39 -- Default Constants
40
41 -- Interconnect Declarations
42 SIGNAL a_rsc1_d : STD_LOGIC_VECTOR (7 DOWNTO 0);
43 SIGNAL b_rsc1_d : STD_LOGIC_VECTOR (7 DOWNTO 0);
44 SIGNAL c_rsc1_d : STD_LOGIC_VECTOR (7 DOWNTO 0);
45 SIGNAL return_rsc1_d : STD_LOGIC_VECTOR (7 DOWNTO 0);
46 SIGNAL reg_return_rsc_triosy_obj_ld_cse : STD_LOGIC;
```

The Transcript window at the bottom shows the following messages:

```
0 Errors - 5 Warnings - 74 Infos - 74 Comments - 14 Commands - 0 Subcommands - Location
# Message Id
# Finished writing concatenated simulation file: /home/radiocad/eit-mom/Catapult/f.v5/concat_sim_rtl.v
# flow run /Schematic/view ./schematic.nlv -state rtl
# flow run /Schematic/view ./schematic.nlv -state datapath
# flow run /Schematic/view ./schematic.nlv -state datapath
# flow run /Schematic/view ./schematic.nlv -state criticalpath
# flow run /Schematic/view ./schematic.nlv -state criticalmap
# flow run /Schematic/view ./schematic.nlv -state rtl
```





Test Bench

```
#include "ac_int.h";
#include <iostream>
#include <fstream>
// Include the C/C++ function header
#include "lab1.h"
// Include the SCVerify header
#include "mc_testbench.h"
int main(int argc, char *argv[]) {
    ac_int<8> a = 1;
    ac_int<8> b = 2;
    ac_int<8> result = 0;
    // Test simuli. Five iterations.
    for (int s_idx = 0; s_idx < 5; s_idx++) {
        result = f(a,b,s_idx);
        // Generate some output
        std::cout << "a*b*c = result: " << a.to_int() << "*"
<< b.to_int() << "*" << s_idx << " = " << result.to_int()
<< std::endl;
    }
    return 0;
}
```

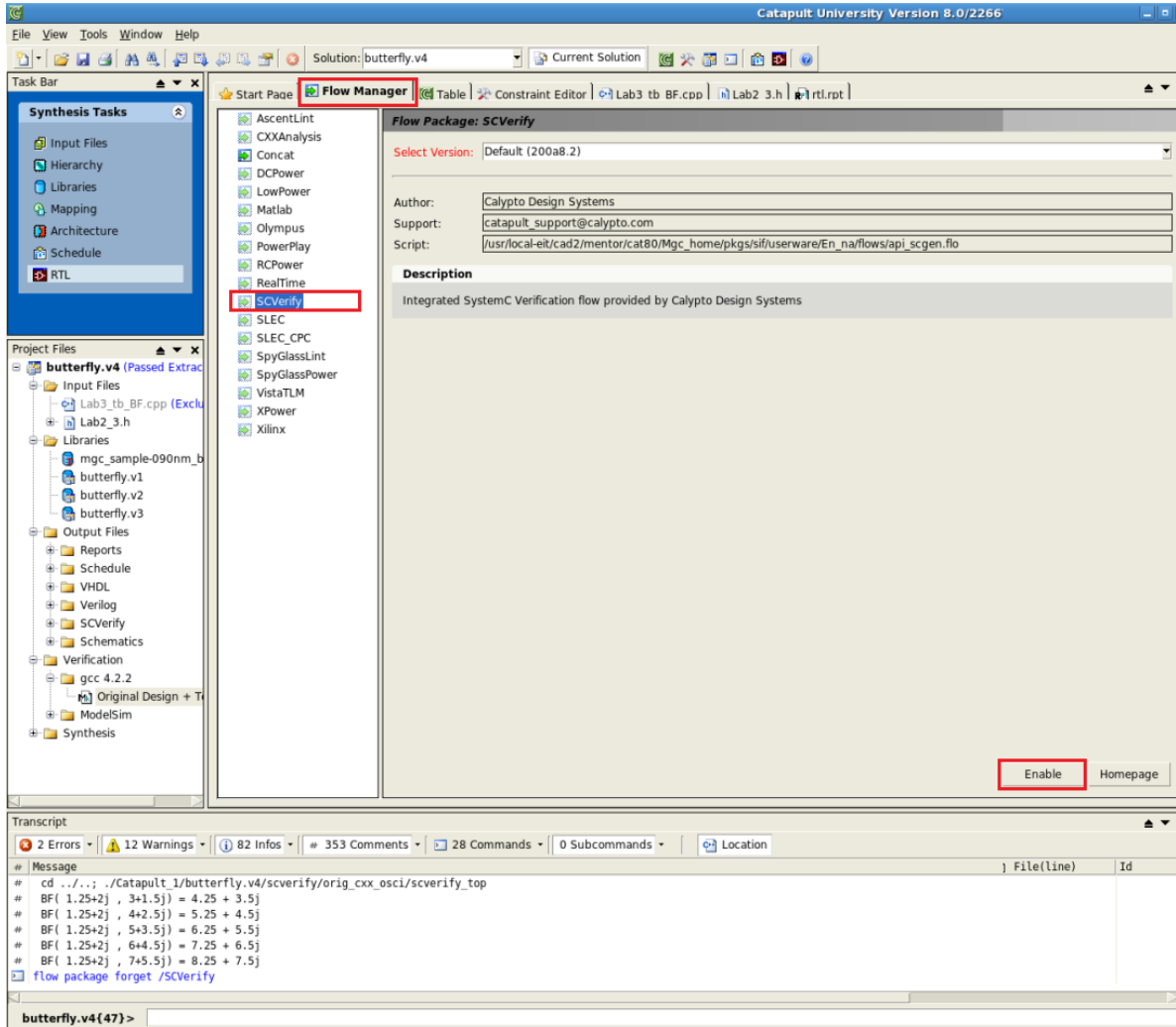
- Include your design source file
- Call your design
- Generate inputs to the design
- Write outputs in the console

Design Verification

- ❑ **SCVerify** flow in Catapult automatically generates the verification infrastructure.
 - The functionality of the generated RTL against the users original source code will be verified.

- ❑ SCVerify supports Mentor QuestaSim/ModelSim, Synopsys VCS and Cadence IUS/NCSim simulation environments.

Design Verification



Design Verification

The screenshot displays the Catapult University software interface. The main window is titled "Flow Manager" and shows the "Flow Package: SCVerify" configuration. The "Reset Duration (cycles)" is set to 2, and "Synchronize all resets" is checked. The "Default stack size" is 64000000. The "Testbench invocation args" field is empty. The "Abort simulation when error count reaches:" is set to 0. The "Enable automatic deadlock detection" checkbox is checked. The "Additional include directory paths:" and "Additional link library paths:" fields are empty. The "Additional link libraries:" section has several checkboxes: "Use ModelSim / QuestaSim for simulation" (checked), "Use OSCI for simulation" (checked), "Use NCSim for simulation" (unchecked), "Use VCS for simulation" (unchecked), "Turn off inputs when empty" (unchecked), "Use design IDLE to synchronize transactions" (unchecked), "On output mismatch (golden vs DUT), print only the array elements that mismatch" (checked), "Add probes to waveform" (unchecked), and "Compile SCVerify wrappers optimized" (unchecked). The "Generate makefiles at stages:" field contains "schedule extract power".

The "Project Files" pane on the left shows a tree view for the "butterfly.v4 (Passed Extract)" project. The "Verification" folder is expanded, showing "gcc 4.2.2" and "Original Design + Testbench" subfolders. The "Transcript" pane at the bottom shows the following output:

```
2 Errors - 12 Warnings - 82 Infos - 355 Comments - 30 Commands - 0 Subcommands - Location
# Message
# BF( 1.25+2j , 6+4.5j) = 7.25 + 6.5j
# BF( 1.25+2j , 7+5.5j) = 8.25 + 7.5j
# flow package target /SCVerify
# flow package require /SCVerify
# 200a8.2
# flow run /SCVerify
# Makefile for Original Design + Testbench written to file './scverify/Verify_orig_cxx_osci.mk'
```



References

- ❑ High-Level Synthesis, Blue Book, Mentor Graphics Corporation
- ❑ Algorithmic C™ Datatypes, Calypto Design Systems, Inc



LUND
UNIVERSITY