

20 YEARS of

ALTERA®

INNOVATION



SOPC

WORLD

2003



SOPC
WORLD
2003

Timing-Driven Design for Optimal Area & Performance

October 2003

Agenda

- **Synplicity Overview**
- **Synplicity's Solutions**
- **Identify Demo**

Synplicity, Inc.

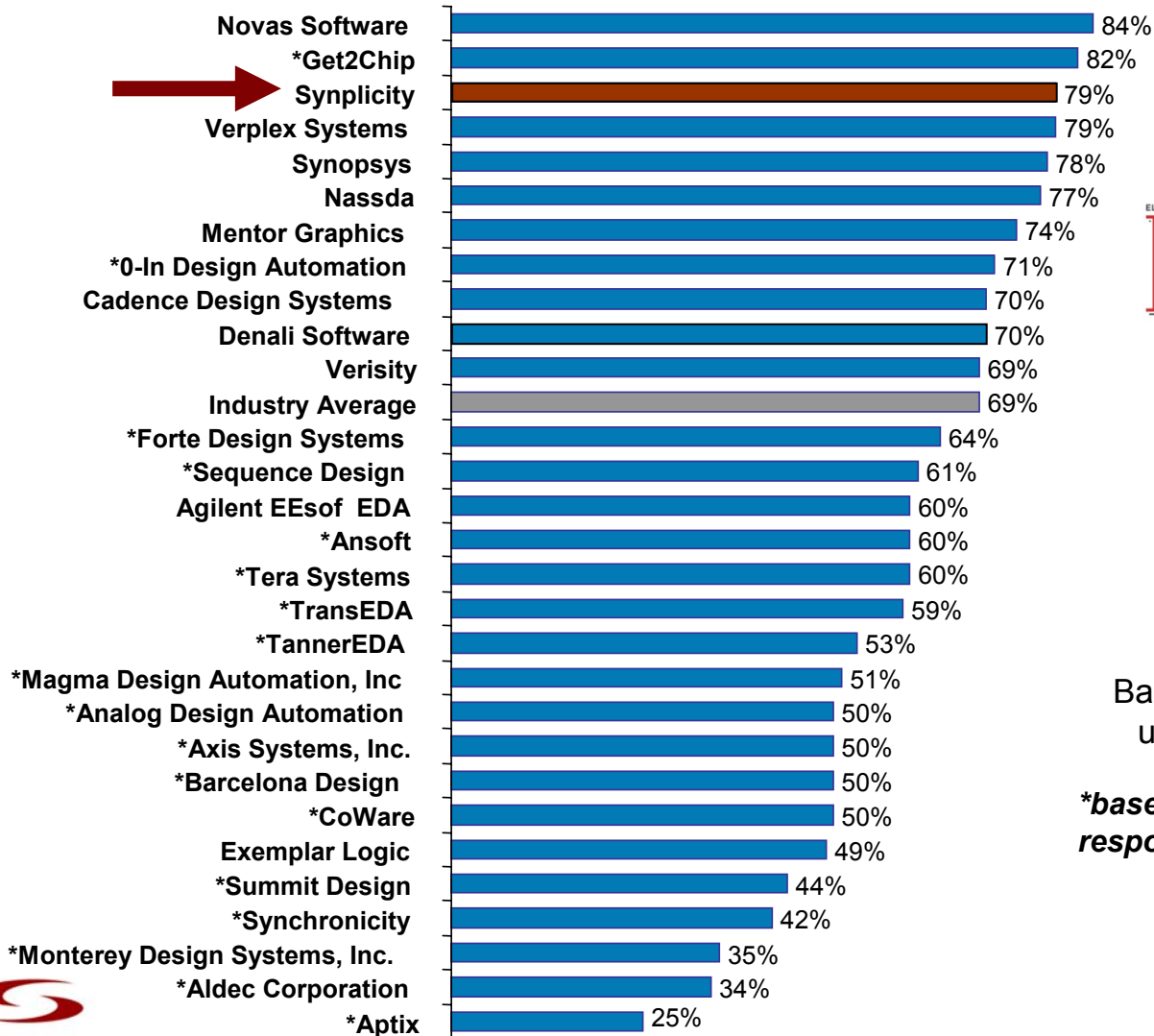
- EDA software company founded in 1994
- Unique company philosophy
 - Best results don't have to come from hard to use tools
 - Flexible and easy to work with company
 - Dedicated to providing the best technical support
- FPGA expertise
 - Synthesis
 - Physical Synthesis
 - Debug



Simply Better Results[®]



Top Ranking in Customer Satisfaction



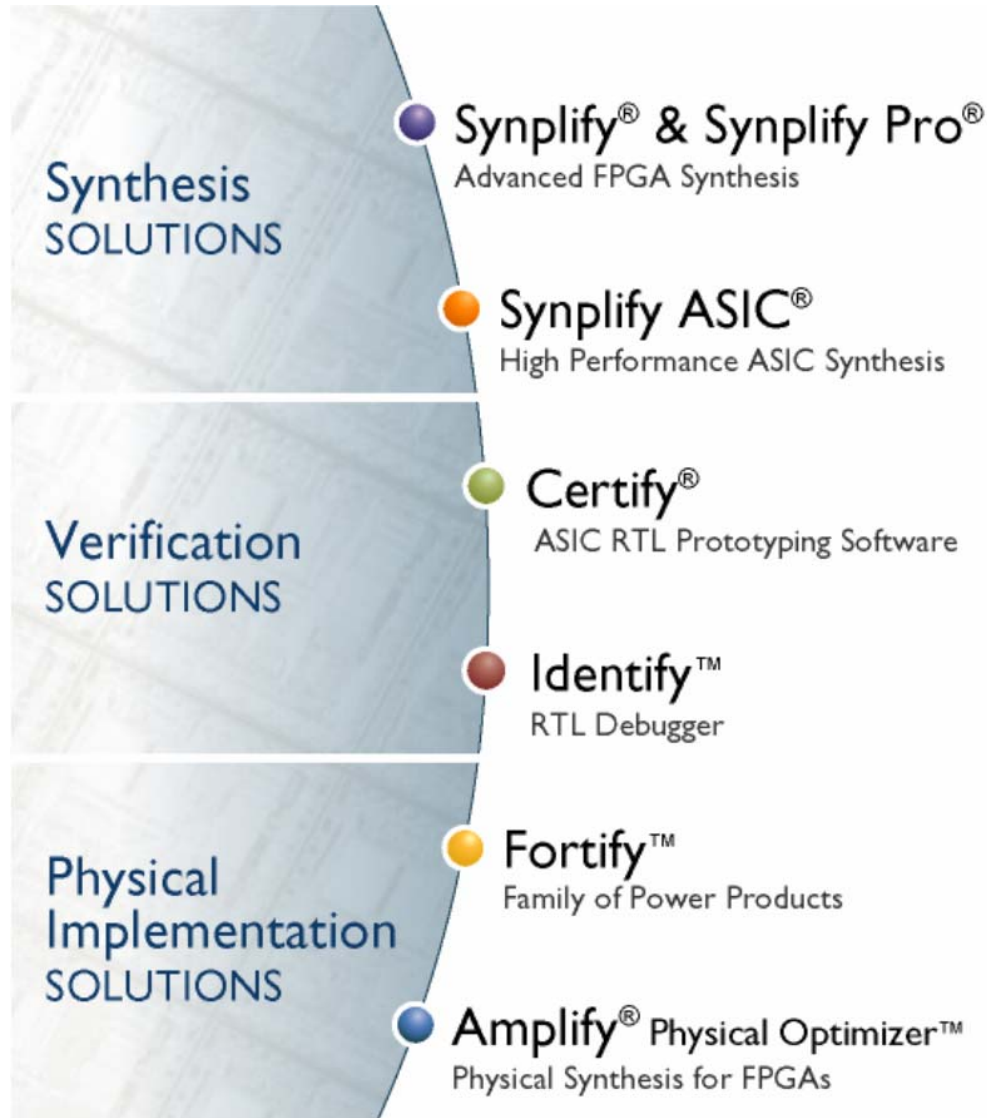
ELECTRONIC ENGINEERING
EETIMES

**Customer
Satisfaction
June 2003**

Base = # who have purchased or used a product from the vendor

**base too small (20 or fewer responses); interpret data with caution*

Synplicity's Design Solutions



Top FPGA Design Challenges

As described by today's FPGA designers

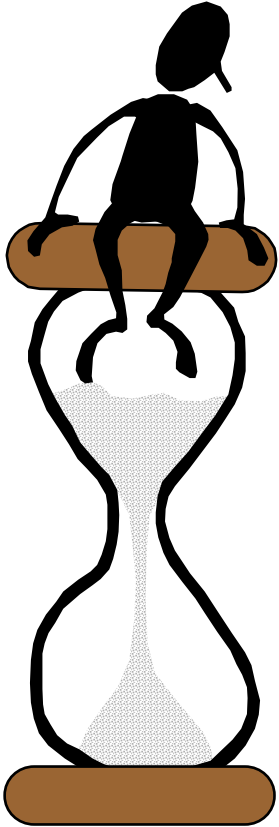
- Achieving Performance Goals
- Timing Closure
- Productivity
- Debug
- Prototyping

Achieving Performance Goals (Synplify & Synplify Pro)

Performance Goals

- Synplify (Mappers) are architecture aware
- Optimizations are performed based on timing constraints
 - Hierarchical boundary optimizationTM
- BEST algorithms extract high level components
 - RAM's, FSM's, wide muxes, adders/multipliers, etc.
 - Technology-specific optimizations made *after* extraction

Set Proper Timing Constraints



- Synplify and Synplify Pro are timing driven
 - Optimization decisions are made based on the timing constraints
 - Not simply optimizing for performance or area
 - Saves on device cost by using the smallest part while meeting your timing
- Forward annotation to P&R
 - Timing information is forward annotated to Quartus place & route
 - More detailed and accurate timing constraints yield the best results

Basic Clock Options

■ Frequency / Period

- Enter one (displayed in bold)
- Others are automatically derived (regular font)



■ Clock Group

- Default clock group: *default_clkgroup*
- Only paths between clock domain from the same clock group are analyzed

■ Use real constraints for your design – don't over constrain

■ To optimize strictly for area set frequency to 1MHz

	Enabled	Clock	Frequency (MHz)	Period (ns)	Clock Group	Clock Rise (ns)	Clock Fall (ns)	Duty Cycle (%)	Route (ns)	Virtual Clock
3	<input checked="" type="checkbox"/>	clk3	100	10.000	D1			50	2	<input type="checkbox"/>
4	<input checked="" type="checkbox"/>	clk4	83.333	12.000	D1			50		<input checked="" type="checkbox"/>

Navigation: < > Clocks / Inputs/Outputs / Registers / Multi-Cycle Paths / False Paths / Attributes / Other <

Advanced Clock Options

■ Rise, Fall, Duty Cycle

- Use *Clock Rise/Fall* to specify rising and falling edges
 - duty cycle automatically derived
- OR Specify the clock cycle as a % of the clock period
 - rise reset to 0, fall automatically derived
- Default is Rise=0, Fall=period/2, Duty Cycle=50%

■ Route

- Use to shrink the effective clock period without affecting the clock constraint forward-annotated to Quartus P&R

■ Virtual Clock

- Use for external clock signals clocking top-level ports

IP Support

- Altera Clearbox support allows Synplify to perform realistic timing analysis, optimizations, and reporting
 - Synplify Pro only

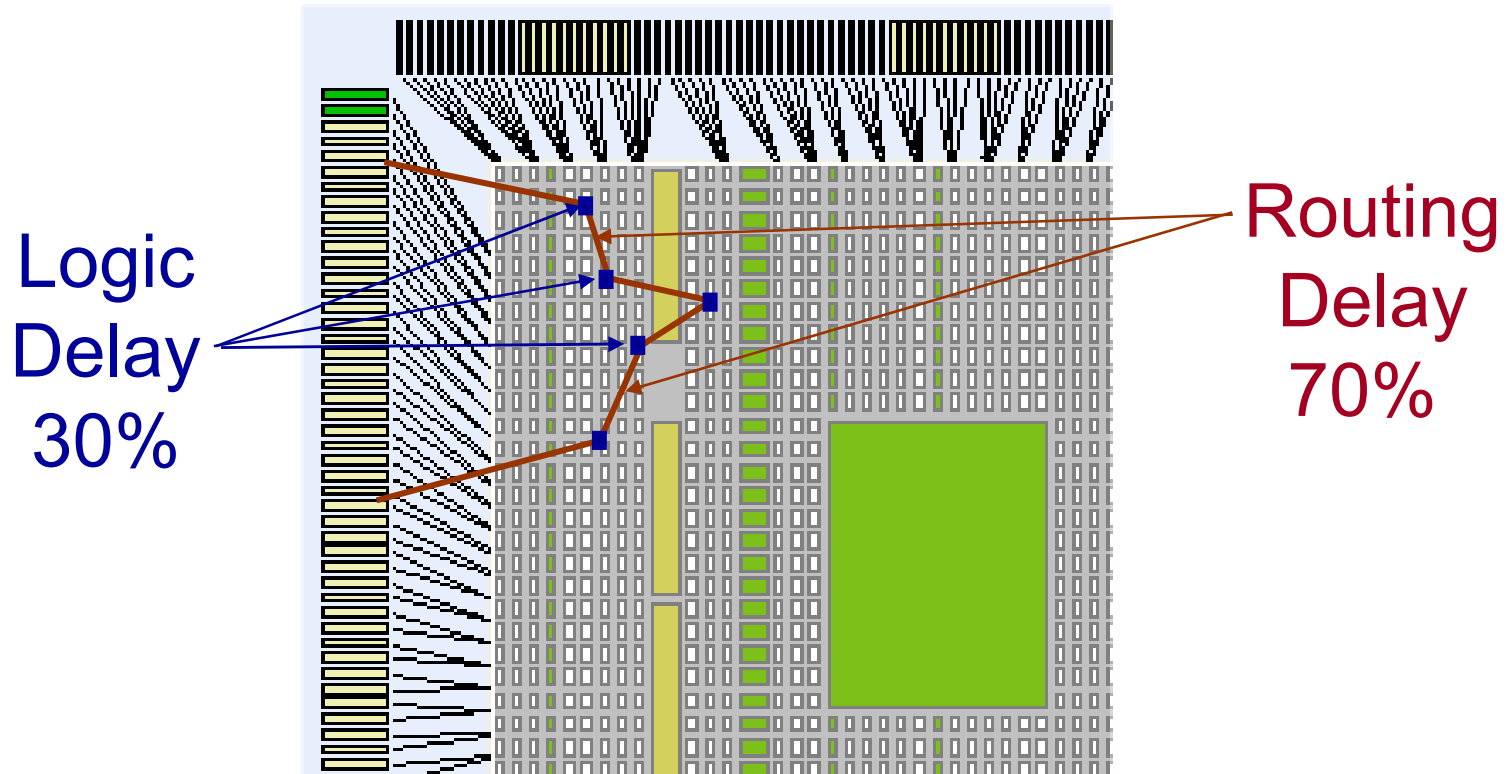
- Mixed language support
 - Synplify Pro only

Forward Annotation to Quartus

- Timing constraints forward annotated to Quartus Place & Route
 - Frequency
 - Duty Cycle
 - I/O Delay
 - Multi-Cycle Paths
 - False Paths & max_delay path
 - Clock Relationships
 - Pin Assignments
- Accomplished through .vqm & .tcl

Timing Closure (Amplify Physical Optimizer)

Routing Governs Performance



- Must incorporate physical information into synthesis
- The larger the design, the larger the problem

Amplify Physical Synthesis

- Simultaneous placement and optimization
- Integrated Design Planning guides physical optimization
- Boosts performance an average of 20% over synthesis

Synplify Pro with Amplify - amplify2 *

File Edit View Project Run HDL Analyst Tools Options Window Help

amplify2 *

Name	Area	Area Use	Area Use(%)
rgn1	720	70	9%
rgn2	1...	149	11%

Sheet 1 of 1 - top level (of module EIGHT_BIT_UC) (RTL View) Impl-npc

Instances (8)

- REGS (REG_FILE)
- UC_ALU (ALU) Axi
- SPECIAL_REGS (SPECIAL_REGS)
- DMux (Data_Mux)
- PignCtrl (Pign_Ctr)
- DECODE (INS_De)
- ROM (INS_ROM)
- IQ_buff (IQ)
- Primitives
- Pots (5)
- Nets (35)
- Clock Tree

eight_bit_uc... | eight_bit_uc... | Log Parameter | npc

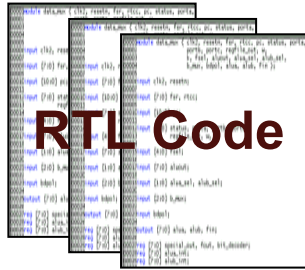
TCL Script | 0 errors | 0 warnings | 0 notes |

Drag over rgn2 (Stratix Region)

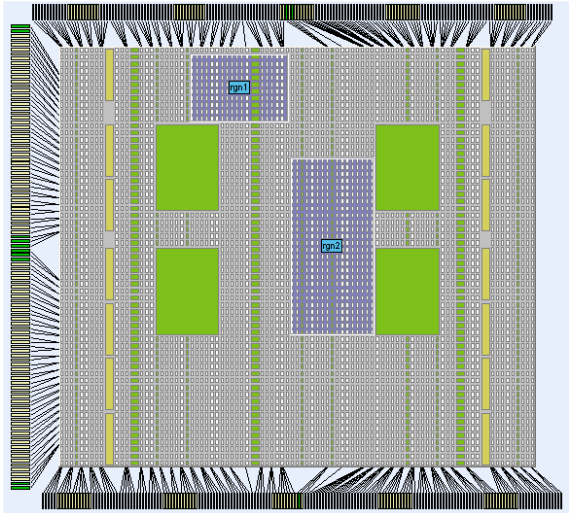
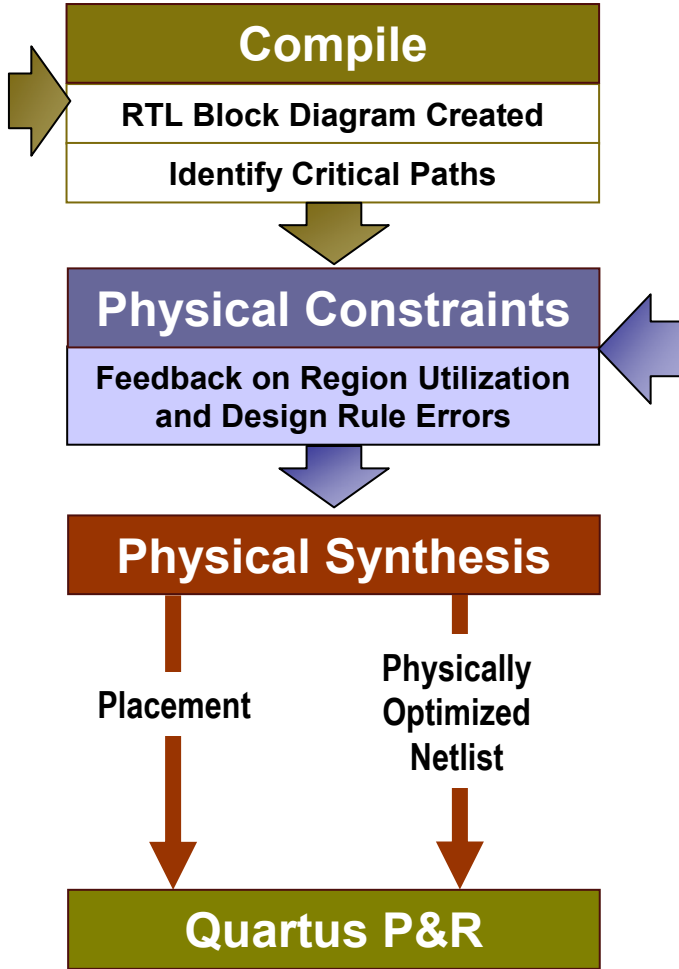
When To Use Amplify

- For fast timing closure
- When you need the highest possible performance
- Need to reduce a speed-grade for cost
- When the majority of delay is in routing

Amplify – Interactive Flow



```
module m0 (clk, rst, enable, for, r0, r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12, r13, r14, r15, r16, r17, r18, r19, r20, r21, r22, r23, r24, r25, r26, r27, r28, r29, r30, r31, r32, r33, r34, r35, r36, r37, r38, r39, r40, r41, r42, r43, r44, r45, r46, r47, r48, r49, r50, r51, r52, r53, r54, r55, r56, r57, r58, r59, r60, r61, r62, r63, r64, r65, r66, r67, r68, r69, r70, r71, r72, r73, r74, r75, r76, r77, r78, r79, r80, r81, r82, r83, r84, r85, r86, r87, r88, r89, r90, r91, r92, r93, r94, r95, r96, r97, r98, r99, r100, r101, r102, r103, r104, r105, r106, r107, r108, r109, r110, r111, r112, r113, r114, r115, r116, r117, r118, r119, r120, r121, r122, r123, r124, r125, r126, r127, r128, r129, r130, r131, r132, r133, r134, r135, r136, r137, r138, r139, r140, r141, r142, r143, r144, r145, r146, r147, r148, r149, r150, r151, r152, r153, r154, r155, r156, r157, r158, r159, r160, r161, r162, r163, r164, r165, r166, r167, r168, r169, r170, r171, r172, r173, r174, r175, r176, r177, r178, r179, r180, r181, r182, r183, r184, r185, r186, r187, r188, r189, r190, r191, r192, r193, r194, r195, r196, r197, r198, r199, r200, r201, r202, r203, r204, r205, r206, r207, r208, r209, r210, r211, r212, r213, r214, r215, r216, r217, r218, r219, r220, r221, r222, r223, r224, r225, r226, r227, r228, r229, r230, r231, r232, r233, r234, r235, r236, r237, r238, r239, r240, r241, r242, r243, r244, r245, r246, r247, r248, r249, r250, r251, r252, r253, r254, r255, r256, r257, r258, r259, r260, r261, r262, r263, r264, r265, r266, r267, r268, r269, r270, r271, r272, r273, r274, r275, r276, r277, r278, r279, r280, r281, r282, r283, r284, r285, r286, r287, r288, r289, r290, r291, r292, r293, r294, r295, r296, r297, r298, r299, r300, r301, r302, r303, r304, r305, r306, r307, r308, r309, r310, r311, r312, r313, r314, r315, r316, r317, r318, r319, r320, r321, r322, r323, r324, r325, r326, r327, r328, r329, r330, r331, r332, r333, r334, r335, r336, r337, r338, r339, r340, r341, r342, r343, r344, r345, r346, r347, r348, r349, r350, r351, r352, r353, r354, r355, r356, r357, r358, r359, r360, r361, r362, r363, r364, r365, r366, r367, r368, r369, r370, r371, r372, r373, r374, r375, r376, r377, r378, r379, r380, r381, r382, r383, r384, r385, r386, r387, r388, r389, r390, r391, r392, r393, r394, r395, r396, r397, r398, r399, r400, r401, r402, r403, r404, r405, r406, r407, r408, r409, r410, r411, r412, r413, r414, r415, r416, r417, r418, r419, r420, r421, r422, r423, r424, r425, r426, r427, r428, r429, r430, r431, r432, r433, r434, r435, r436, r437, r438, r439, r440, r441, r442, r443, r444, r445, r446, r447, r448, r449, r450, r451, r452, r453, r454, r455, r456, r457, r458, r459, r460, r461, r462, r463, r464, r465, r466, r467, r468, r469, r470, r471, r472, r473, r474, r475, r476, r477, r478, r479, r480, r481, r482, r483, r484, r485, r486, r487, r488, r489, r490, r491, r492, r493, r494, r495, r496, r497, r498, r499, r500, r501, r502, r503, r504, r505, r506, r507, r508, r509, r510, r511, r512, r513, r514, r515, r516, r517, r518, r519, r520, r521, r522, r523, r524, r525, r526, r527, r528, r529, r530, r531, r532, r533, r534, r535, r536, r537, r538, r539, r540, r541, r542, r543, r544, r545, r546, r547, r548, r549, r550, r551, r552, r553, r554, r555, r556, r557, r558, r559, r560, r561, r562, r563, r564, r565, r566, r567, r568, r569, r570, r571, r572, r573, r574, r575, r576, r577, r578, r579, r580, r581, r582, r583, r584, r585, r586, r587, r588, r589, r590, r591, r592, r593, r594, r595, r596, r597, r598, r599, r600, r601, r602, r603, r604, r605, r606, r607, r608, r609, r610, r611, r612, r613, r614, r615, r616, r617, r618, r619, r620, r621, r622, r623, r624, r625, r626, r627, r628, r629, r630, r631, r632, r633, r634, r635, r636, r637, r638, r639, r640, r641, r642, r643, r644, r645, r646, r647, r648, r649, r650, r651, r652, r653, r654, r655, r656, r657, r658, r659, r660, r661, r662, r663, r664, r665, r666, r667, r668, r669, r670, r671, r672, r673, r674, r675, r676, r677, r678, r679, r680, r681, r682, r683, r684, r685, r686, r687, r688, r689, r690, r691, r692, r693, r694, r695, r696, r697, r698, r699, r700, r701, r702, r703, r704, r705, r706, r707, r708, r709, r710, r711, r712, r713, r714, r715, r716, r717, r718, r719, r720, r721, r722, r723, r724, r725, r726, r727, r728, r729, r730, r731, r732, r733, r734, r735, r736, r737, r738, r739, r740, r741, r742, r743, r744, r745, r746, r747, r748, r749, r750, r751, r752, r753, r754, r755, r756, r757, r758, r759, r760, r761, r762, r763, r764, r765, r766, r767, r768, r769, r770, r771, r772, r773, r774, r775, r776, r777, r778, r779, r780, r781, r782, r783, r784, r785, r786, r787, r788, r789, r790, r791, r792, r793, r794, r795, r796, r797, r798, r799, r800, r801, r802, r803, r804, r805, r806, r807, r808, r809, r810, r811, r812, r813, r814, r815, r816, r817, r818, r819, r820, r821, r822, r823, r824, r825, r826, r827, r828, r829, r830, r831, r832, r833, r834, r835, r836, r837, r838, r839, r840, r841, r842, r843, r844, r845, r846, r847, r848, r849, r850, r851, r852, r853, r854, r855, r856, r857, r858, r859, r860, r861, r862, r863, r864, r865, r866, r867, r868, r869, r870, r871, r872, r873, r874, r875, r876, r877, r878, r879, r880, r881, r882, r883, r884, r885, r886, r887, r888, r889, r890, r891, r892, r893, r894, r895, r896, r897, r898, r899, r900, r901, r902, r903, r904, r905, r906, r907, r908, r909, r910, r911, r912, r913, r914, r915, r916, r917, r918, r919, r920, r921, r922, r923, r924, r925, r926, r927, r928, r929, r930, r931, r932, r933, r934, r935, r936, r937, r938, r939, r940, r941, r942, r943, r944, r945, r946, r947, r948, r949, r950, r951, r952, r953, r954, r955, r956, r957, r958, r959, r960, r961, r962, r963, r964, r965, r966, r967, r968, r969, r970, r971, r972, r973, r974, r975, r976, r977, r978, r979, r980, r981, r982, r983, r984, r985, r986, r987, r988, r989, r990, r991, r992, r993, r994, r995, r996, r997, r998, r999, r1000);  
endmodule
```



Types of Physical Constraints

- Module Level Physical Constraints
 - Logical module
 - Use when critical paths are within logical modules
- Detailed Level Physical Constraints
 - Point to point critical path
 - Use when critical paths cross module boundaries
OR
 - When meeting timing within a module becomes critical

Benefits of Physical Synthesis

- Synthesis with physical constraints provides
 - Simultaneous placement and logic optimization
 - Placement based optimization
 - Register replication for high fanout nets across region boundaries
 - Register tunneling across boundaries
 - Replication and re-assignment of registers that drive primary I/Os
 - Placement of logic constrained to regions

Value of Using the Best Synthesis

■ Saves you money



\$100 / chip



\$120 / chip



\$144 / chip

**20% premium
for 12%-15%
performance**

Savings are huge for volume applications

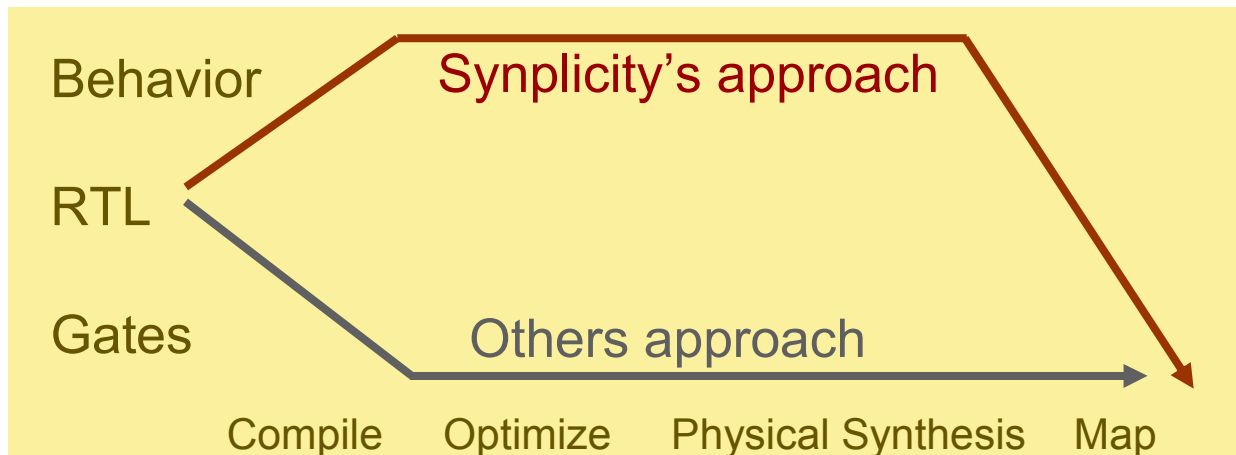
■ Makes your products more competitive

- Better performing chips make a better product
- Reaching timing goals quickly gets you to market sooner

Design Productivity (Synplify & Synplify Pro)

Core Synthesis Technology

- **BEST™** – **B**ehavior **E**xtracting **S**ynthesis **T**echnology®
 - Infers and optimizes behavior from RTL
 - Optimizes across hierarchical module boundaries
 - Integrated physical synthesis algorithms
 - Multi-million gate capacity
 - Extremely FAST - Unparalleled runtimes



Managing Complex Designs

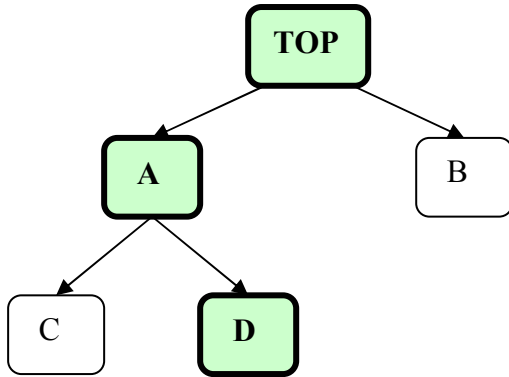
MultiPoint™

■ A Powerful Synthesis Flow for

- Incremental design using Synplify Pro or Amplify
- Unlimited gate capacity
- Minimal scripting effort
- No compromise Quality of Results
- Altera Logic Lock flow

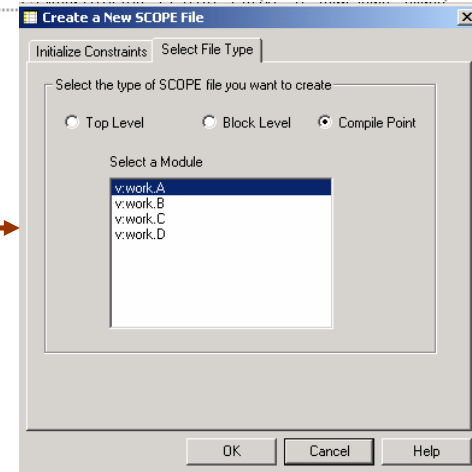


MultiPoint Flow



Enabled	Module	Type
<input checked="" type="checkbox"/>	v:work.A	locked
<input checked="" type="checkbox"/>	v:work.D	locked

Define compile points and compile point constraints



First Run

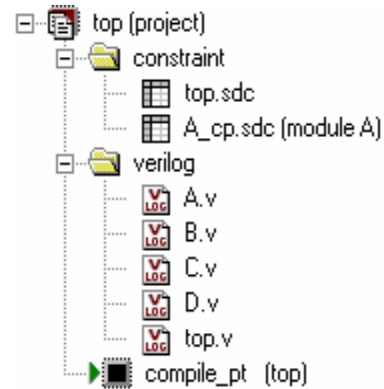
Summary of Compile Points

Name	Status	Reason
D	Mapped	No database
A	Mapped	No database
top	Mapped	No database

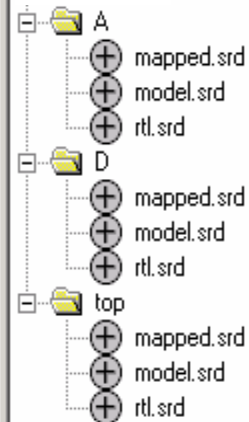
Second Run

Summary of Compile Points

Name	Status	Reason
D	Remapped	Design changed
A	Unchanged	-
top	Unchanged	-



Project View



MultiPoint Synthesis

Difference Based Incremental Synthesis

Re-synthesize a locked Compile Point module for:

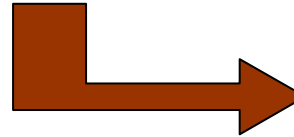
- A logic change in your RTL code
- Changes to constraints
 - Timing constraints change in the .sdc or Project View
 - Project settings change
 - FSM Compiler or Explorer
 - Retiming
 - Pipelining
- Re-synthesis is not based on time stamp

Debug (Identify)

Evolution of Hardware Debug

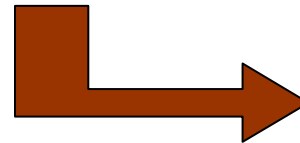
Identify
(Simulator-Like)

Embedded
HDL Analyzer

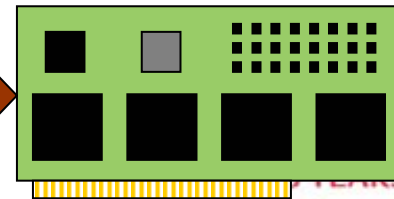


SignalTap
(Logic Analyzer-Like)

Embedded
Logic Analyzer

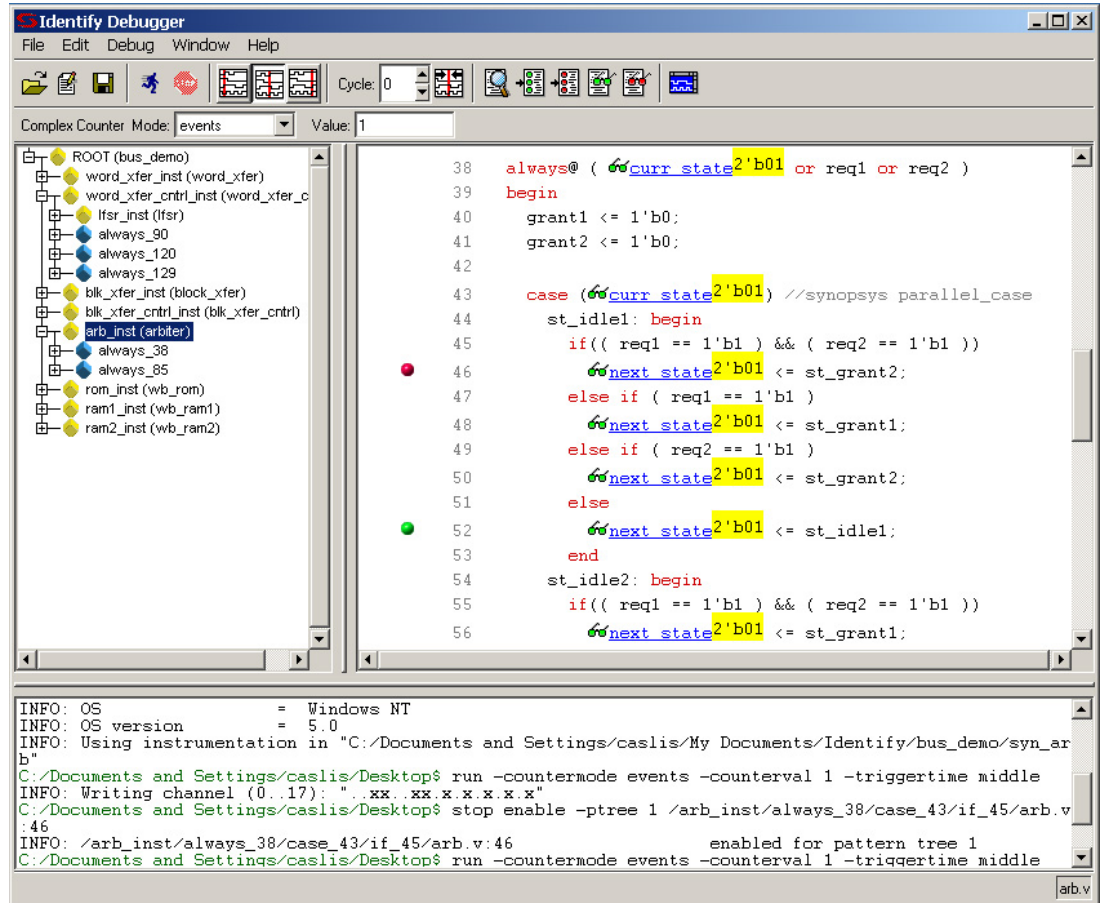


Logic Analyzer



Identify RTL Debugger

- Debug and instrument FPGA directly in RTL code
- Provides internal visibility in the target system at full speed
- Trigger on Data Path and Control Path
- Standard VCD Output for Waveform Display



The screenshot displays the Identify Debugger interface. On the left, a hierarchical tree shows the RTL design structure, with the 'arb_inst (arbiter)' component selected. The main window shows the RTL code for the arbiter, with a red dot indicating a trigger point on line 46. The code snippet is as follows:

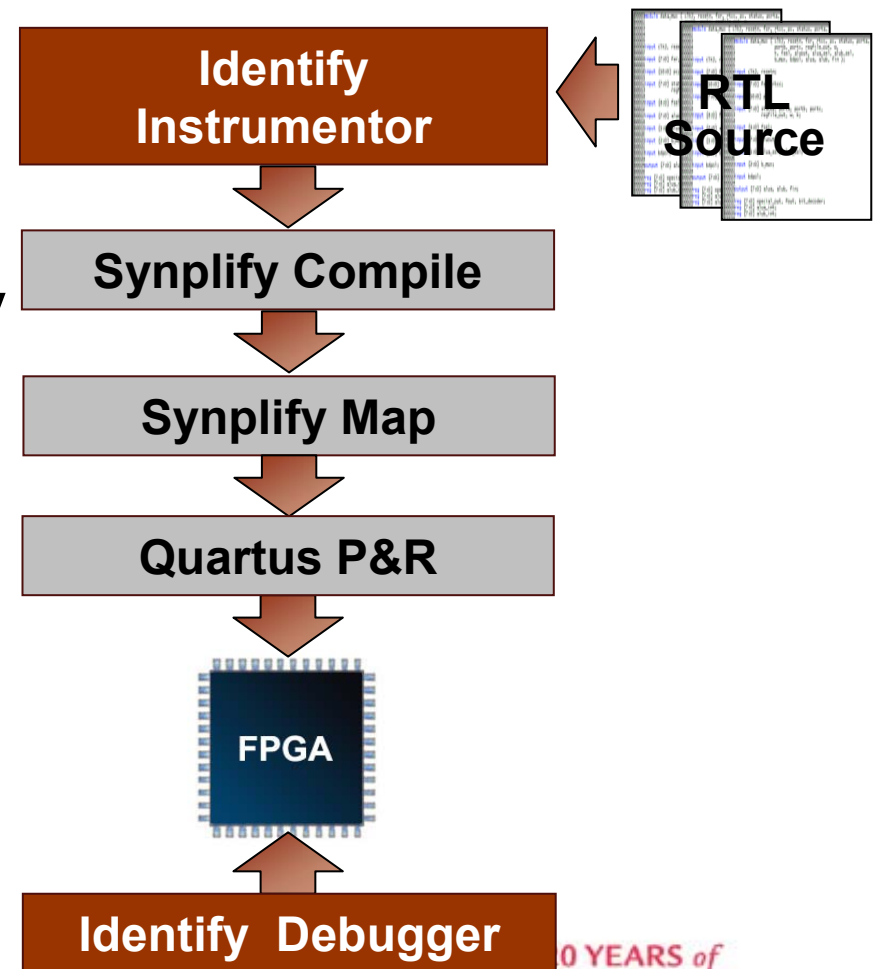
```
38 always@ ( curr_state2'b01 or req1 or req2 )
39 begin
40   grant1 <= 1'b0;
41   grant2 <= 1'b0;
42
43   case ( curr_state2'b01 ) //synopsys parallel_case
44     st_idle1: begin
45       if(( req1 == 1'b1 ) && ( req2 == 1'b1 ))
46         next_state2'b01 <= st_grant2;
47       else if ( req1 == 1'b1 )
48         next_state2'b01 <= st_grant1;
49       else if ( req2 == 1'b1 )
50         next_state2'b01 <= st_grant2;
51       else
52         next_state2'b01 <= st_idle1;
53     end
54     st_idle2: begin
55       if(( req1 == 1'b1 ) && ( req2 == 1'b1 ))
56         next_state2'b01 <= st_grant1;
```

The bottom panel shows the command-line output for running the debugger:

```
INFO: OS = Windows NT
INFO: OS version = 5.0
INFO: Using instrumentation in "C:/Documents and Settings/caslis/My Documents/Identify/bus_demo/syn_arb.v"
C:/Documents and Settings/caslis/Desktop$ run -countermode events -counterval 1 -triggertime middle
INFO: Writing channel (0..17): ".xx.xx.x.x.x.x"
C:/Documents and Settings/caslis/Desktop$ stop enable -ptree 1 /arb_inst/always_38/case_43/if_45/arb.v:46
INFO: /arb_inst/always_38/case_43/if_45/arb.v:46 enabled for pattern tree 1
C:/Documents and Settings/caslis/Desktop$ run -countermode events -counterval 1 -triggertime middle
```

Design Flow with Identify

- Read RTL into Identify Instrumentor
- Compile and map Identify output in Synplify
- FPGA Place & Route
- Use Identify Debugger to view data in FPGA



Instrumenting for Debug

Add Signal
Visibility

Design
Hierarchy

TCL
Scripting
Interface

The screenshot shows the Identify Instrumentor application window. The left pane displays a design hierarchy tree with components like 'ROOT (bus_demo)', 'word_xfer_inst', 'blk_xfer_inst', and 'arb_inst (arbiter)'. The center pane shows a Verilog code snippet with line numbers 74-92. A context menu is open over line 88, showing options: 'Sample and trigger', 'Sample only', 'Trigger only', and 'Not instrumented'. The bottom pane shows a TCL console with the following text:

```
INFO: Set instrumentation directory to "C:/Documents and Settings/caslis/My Documents/Identify/bus_demo/syn_arb"  
C:/Documents and Settings/caslis/Desktop$ breakpoints add /arb_inst/always_38/case_43/if_75/arb.v:76  
INFO: Current area estimates: Logic: 341 logic elements, Buffer: 1024 bits  
INFO: /arb_inst/always_38/case_43/if_75/arb.v:76 added  
C:/Documents and Settings/caslis/Desktop$ breakpoints add /arb_inst/always_38/case_43/if_75/if_77/arb.v:80  
INFO: Current area estimates: Logic: 348 logic elements, Buffer: 1152 bits  
INFO: /arb_inst/always_38/case_43/if_75/if_77/arb.v:80 added  
C:/Documents and Settings/caslis/Desktop$
```


Debugging Data From FPGA

Actual data from
FPGA

Click to enable
triggers

Automate Debugger
with Script

A screenshot of the Identify Debugger software interface. The window title is "Identify Debugger". It features a menu bar (File, Edit, Debug, Window, Help), a toolbar with various icons, and a main workspace. On the left, a tree view shows a project structure for "bus_demo" with components like "word_xfer_inst", "blk_xfer_inst", and "arb_inst". The main workspace displays Verilog code with a syntax highlighter. A red dot on the code indicates a trigger point. The bottom pane shows a command prompt with the following text:

```
INFO: OS = Windows NT
INFO: OS version = 5.0
INFO: Using instrumentation in "C:/Documents and Settings/caslis/My Documents/Identify/bus_demo/syn_arb.v"
C:/Documents and Settings/caslis/Desktop$ run -countermode events -counterval 1 -triggertime middle
INFO: Writing channel (0..17): ".xx.xx.x.x.x.x"
C:/Documents and Settings/caslis/Desktop$ stop enable -ptree 1 /arb_inst/always_38/case_43/if_45/arb.v:46
INFO: /arb_inst/always_38/case_43/if_45/arb.v:46 enabled for pattern tree 1
C:/Documents and Settings/caslis/Desktop$ run -countermode events -counterval 1 -triggertime middle
```

RTL Display of Triggers and Sampled Data

- Full support of symbolic values
- Control Path triggers as breakpoints
- Data Path triggers as watch points
- Configurable counters and state machine triggering

```
46  ● grant2'1' <= '0';
47
48  case (● curr_state st_idle2) is
49    when st_idle1 =>
50      if ( ● req1'0' = '1' ) and ( ● req2'1' = '1' ) then
51        ● next_state st_idle2 <= st_grant2;
52      elsif ( ● req1'0' = '1' ) then
53        ● next_state st_idle2 <= st_grant1;
54      elsif ( ● req2'1' = '1' ) then
55        ● next_state st_idle2 <= st_grant2;
56      else
57        ● next_state st_idle2 <= st_idle1;
58      end if;
59  when st_idle2 =>
60    if ( ● req1'0' = '1' ) and ( ● req2'1' = '1' ) then
61      ● next_state st_idle2 <= st_grant1;
62    elsif ( ● req1'0' = '1' ) then
63      ● next_state st_idle2 <= st_grant1;
```

Sampled Data From Chip

Control Path

Data Path

● Trigger



Synplicity®

Simply Better Results

20 YEARS of

ALTERA®

INNOVATION

Setting Triggers on Data Path

```
ex : PROCESS (a, b) BEGIN
CASE curr_state IS
WHEN read_state =>
    IF b = "100110" THEN
        o <= "110";
    END IF;
WHEN OTHERS =>
    o <= "000";
END CASE;
END PROCESS;
```

Watchpoint Setup

Setup the first value only to watch a value in a pattern tree on signal "next_state" or both values to watch a transition from the first to the second in a pattern tree:

First value: state1

Second value (optional): state2

OK Cancel

Setting Triggers on Control Path

a == read_state

**a == read_state
&&**

b == "100110"

a != read_state

```
ex : PROCESS (a, b) BEGIN
CASE a IS
  WHEN read_state =>
    IF b = "100110" THEN
      o <= "110";
    END IF;
  WHEN OTHERS =>
    o <= "000";
  END CASE;
END PROCESS;
```

1

2

3



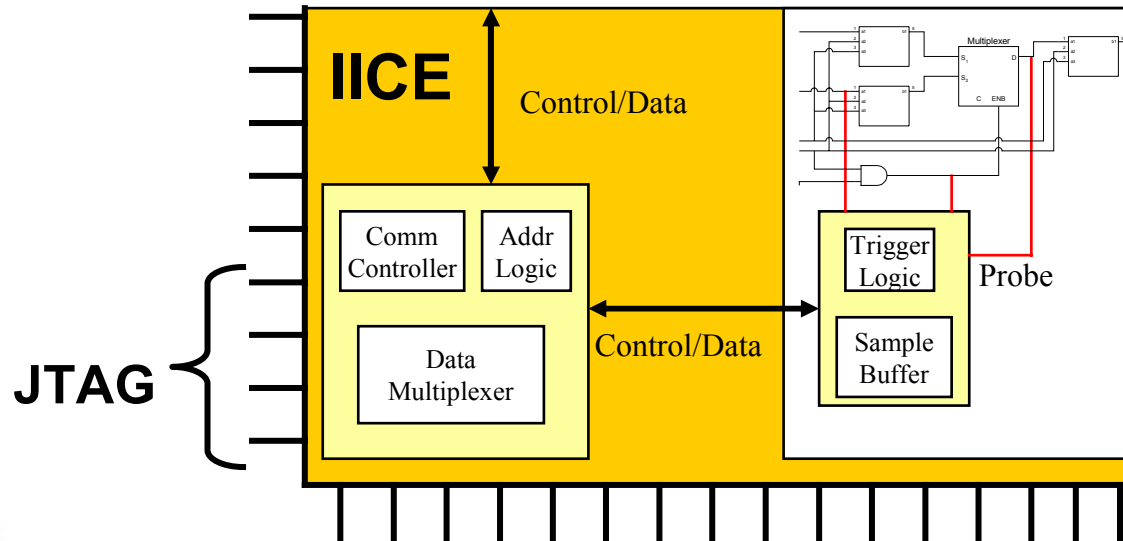
Synplicity®

Simply Better Results

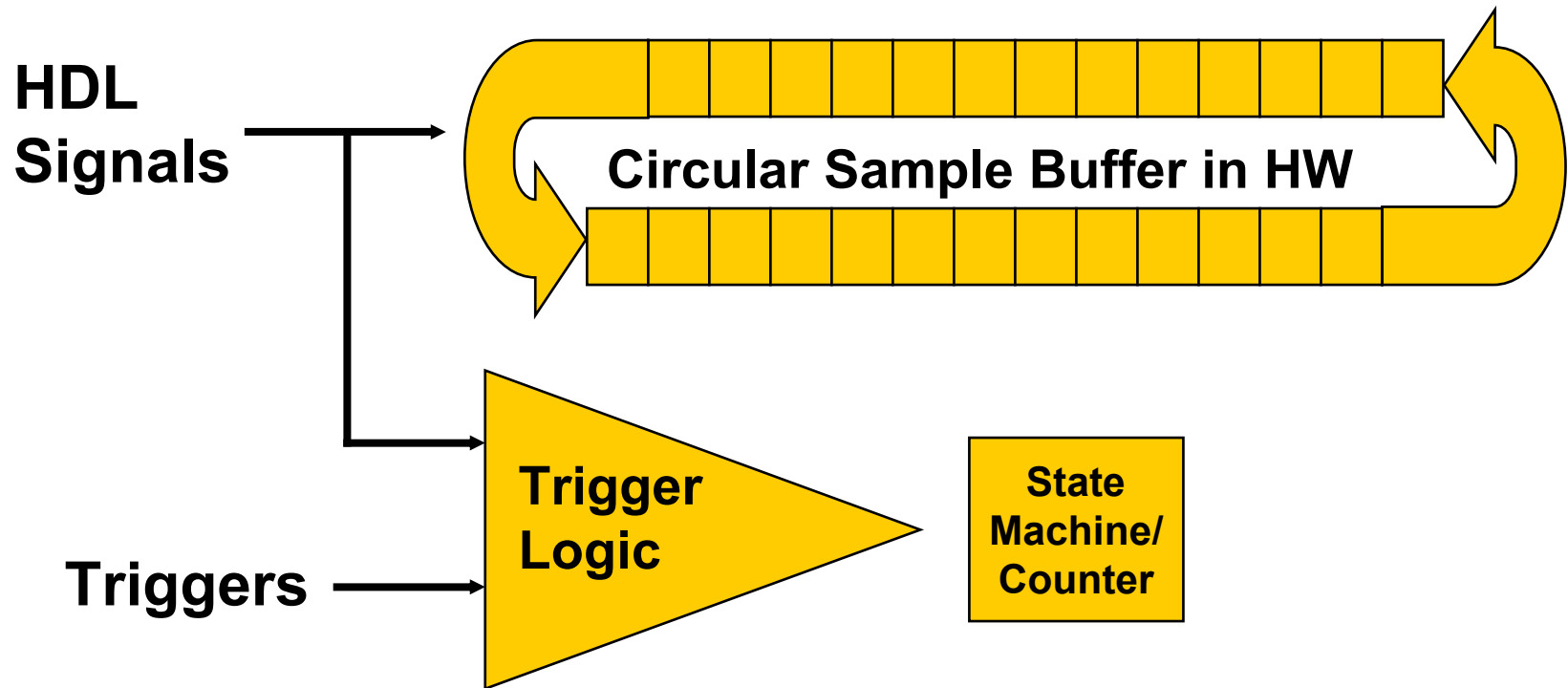
20 YEARS of
ALTERA®
INNOVATION

Intelligent In-Circuit Emulator (IICE™)

- Inserted logic used by Instrumentor & Debugger
- Uses dedicated JTAG pins or user-selected pins
- Includes controller, triggering logic, & data storage buffer



Triggering Logic and Buffer



- Trigger values changed dynamically from debugger
- Trigger halts sampling, Not hardware
- Triggers pipelined, only 2 gate delay

Customer Success

Foundry Networks

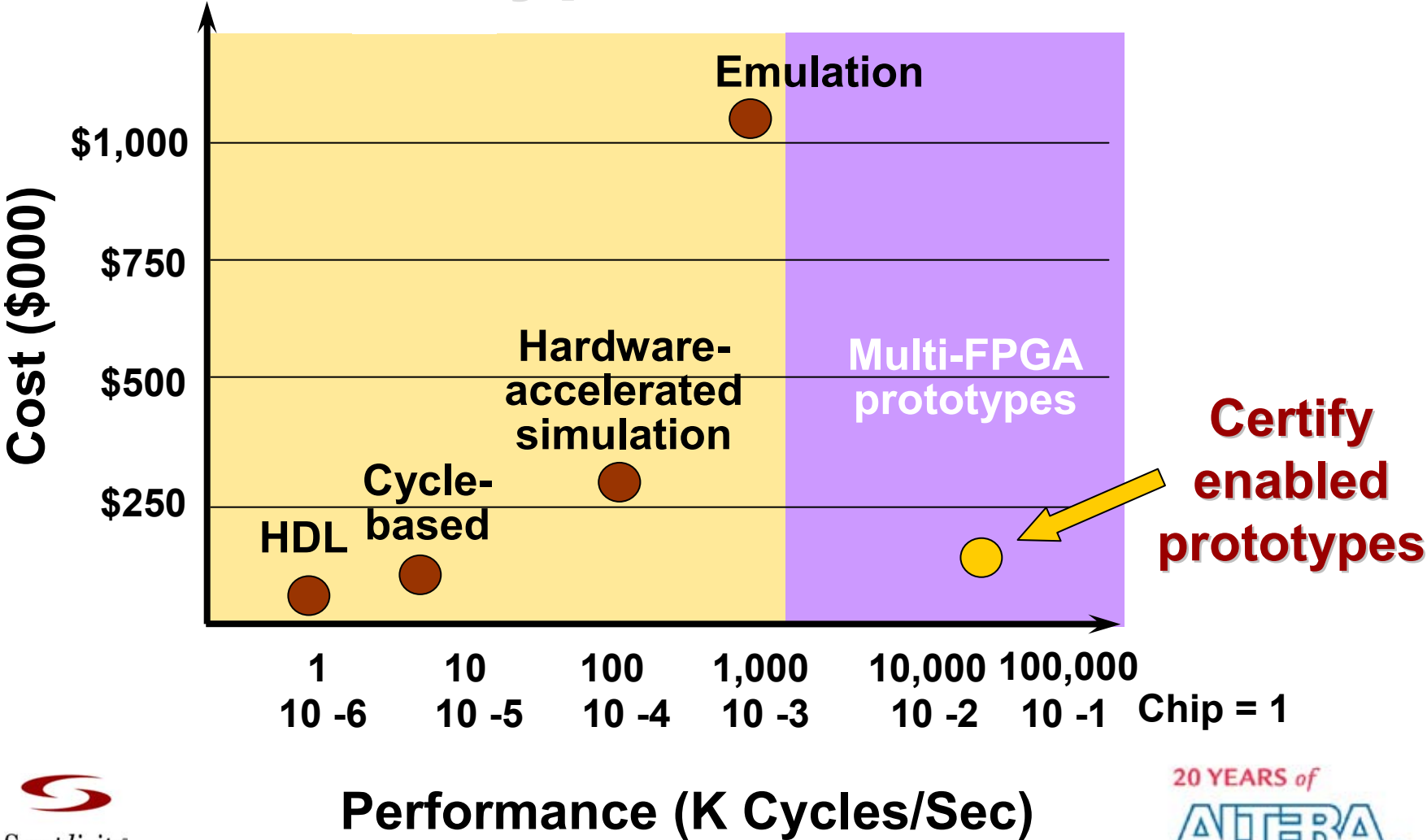
“In just our first six weeks with the product, we used it to find and fix bugs in three of our designs. In each case the process was completed within a day. It would have taken 10 to 20 times longer using traditional test-bench methods.”

-Richard Grenier

Director of ASIC Development

Multi-FPGA Prototyping (Certify)

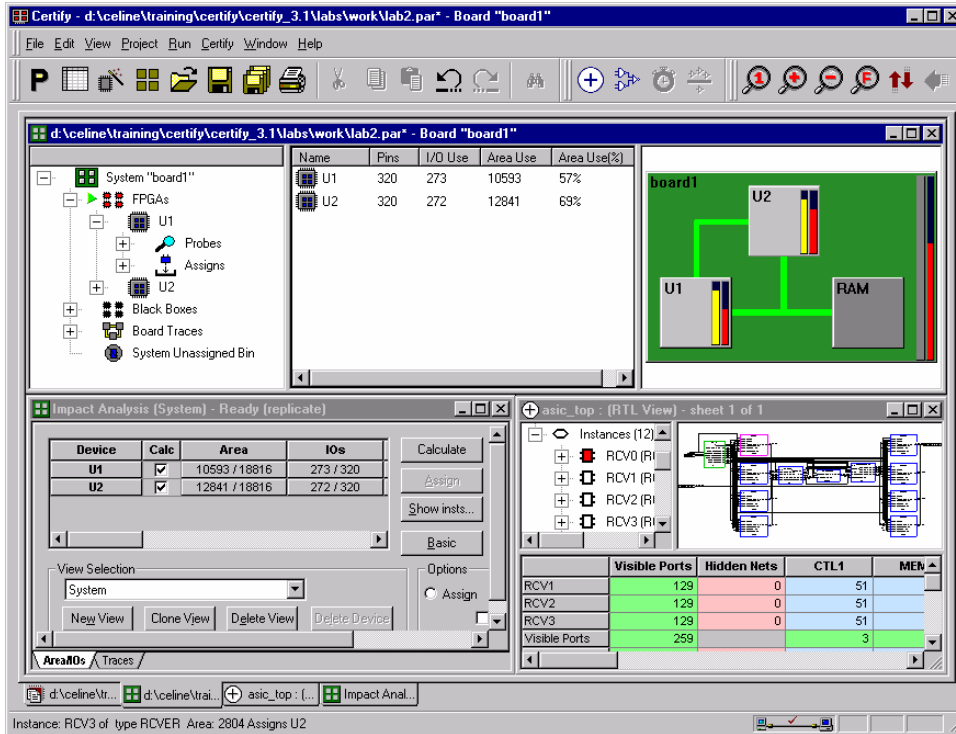
Higher Speed & Lower Cost using FPGA Prototypes



Prototyping Challenges

- ASIC RTL Code
 - Gated Clocks
 - DesignWare™
- Performance
 - Video and signal processing applications
- Partitioning
 - Pin utilization

FPGA-Based ASIC Prototypes

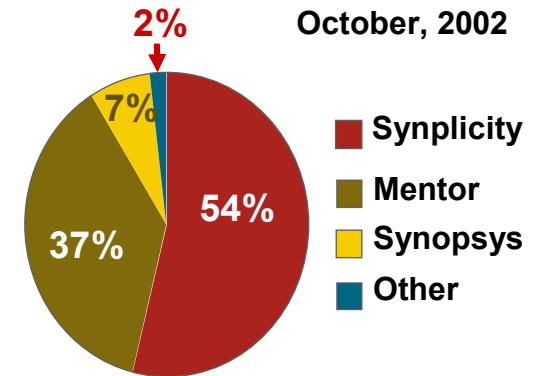


- Highest performing ASIC prototypes for
 - Functional verification
 - HW/SW co-verification
- Automatic RTL partitioning, I/O sharing, and more
- Supports all prototyping hardware including off-the-shelf boards
- Adopted by Philips, TI, LSI Logic and others

Summary

- The Market Leader in FPGA synthesis & physical synthesis
 - Best Quality of Results
 - Unmatched Productivity
- A Leader in EDA innovation
 - First in FPGA physical synthesis
 - Innovative, at-speed, RT-Level debug technology
 - Unique multi-FPGA prototyping system
- Top-ranked customer service and technical support

FPGA Synthesis Market Share
Source: DataQuest, October, 2002





Synplicity[®]

Simply Better Results

END