20 YEARS *of*

ALTERA®

INNOVATION

SOPC WORLD 2003

SOPC WORLD 2003

Timing-Driven Design for Optimal Area & Performance

October 2003

Synplicity

Simply Better Results

20 YEARS of ALTERA INNOVATION

Agenda

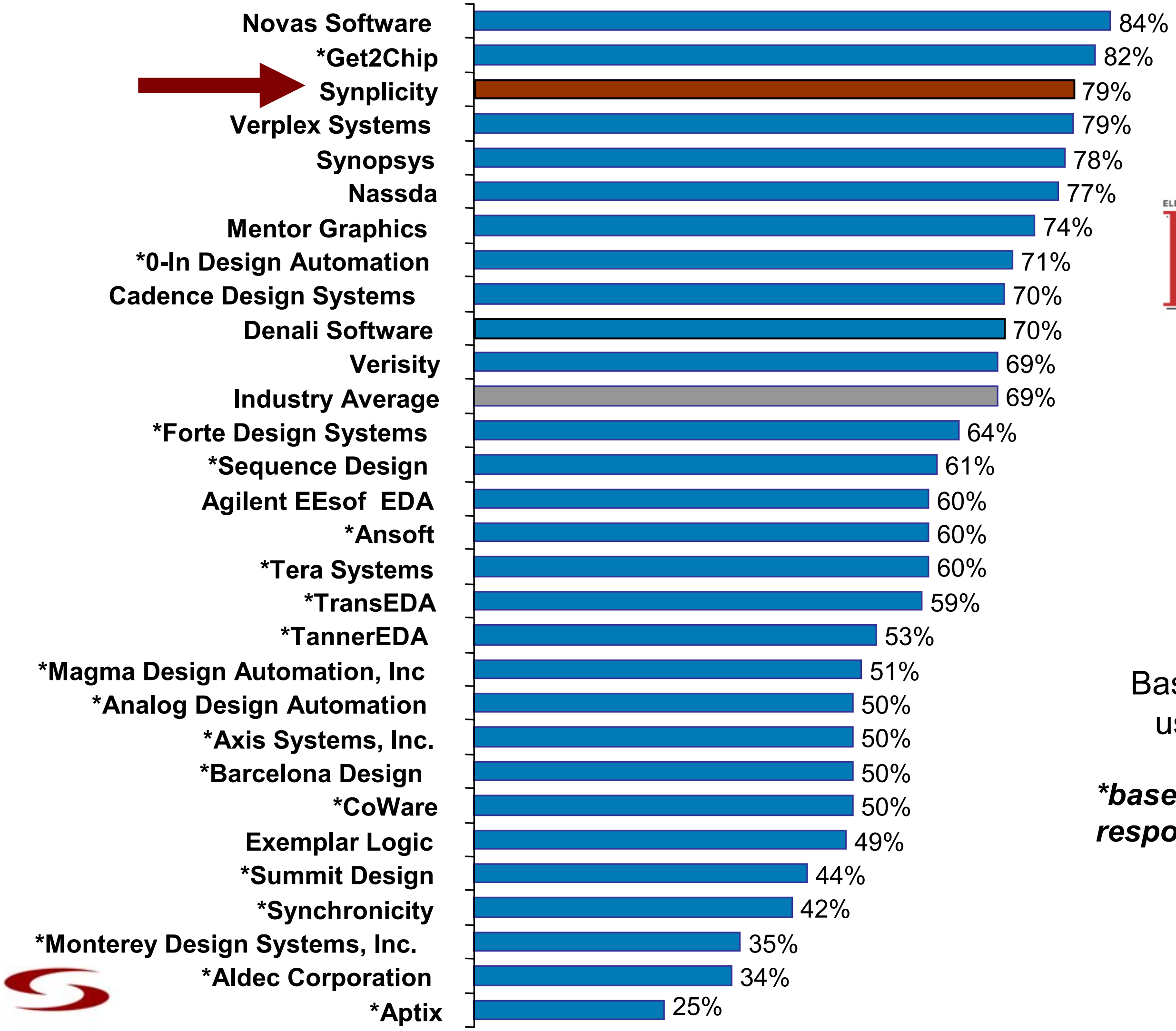
- Synplicity Overview
- Synplicity's Solutions
- Identify Demo

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

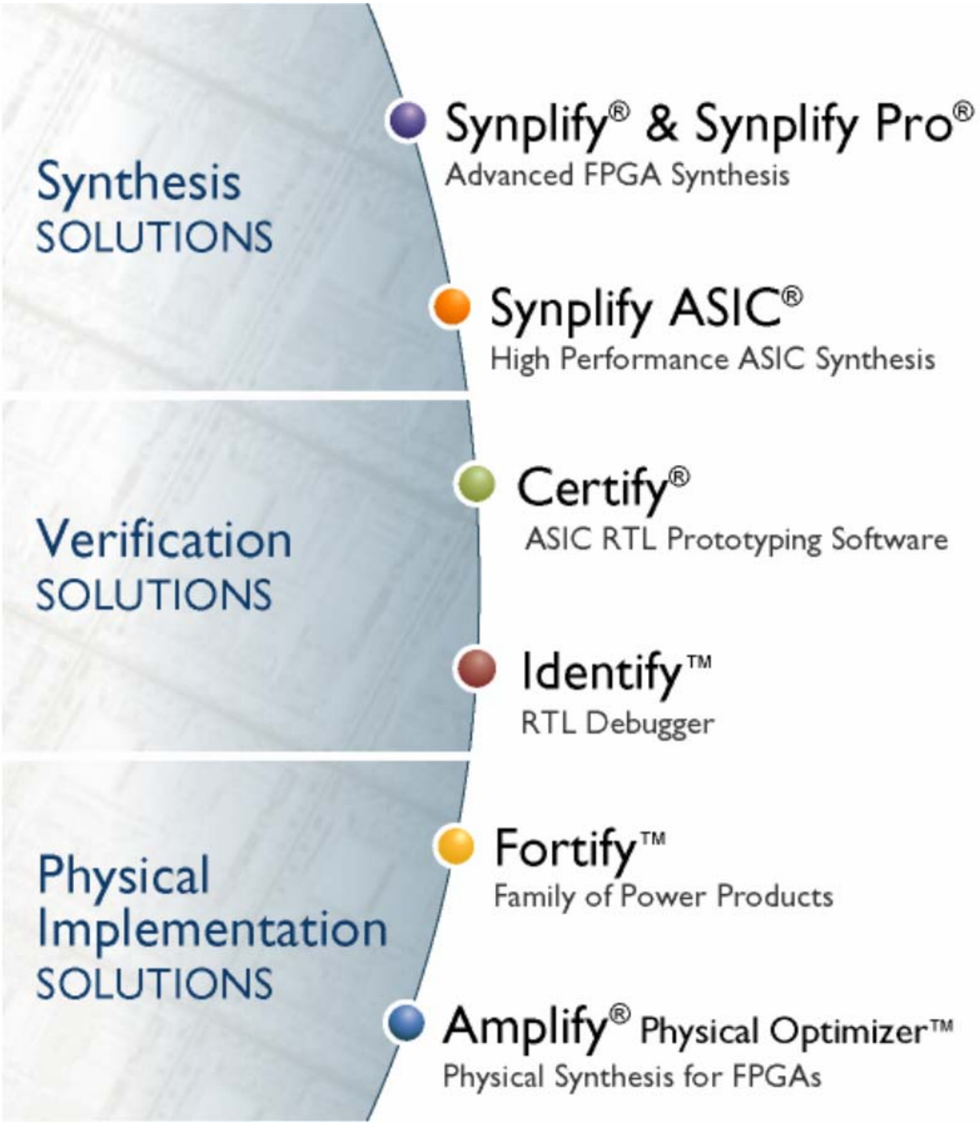
Base = # who have purchased or used a product from the vendor

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

Synplicity
Simply Better Results

20 YEARS of
ALTERA
INNOVATION

Synplicity’s Design Solutions



Synthesis SOLUTIONS

- Synplify® & Synplify Pro®
  Advanced FPGA Synthesis
- Synplify ASIC®
  High Performance ASIC Synthesis

Verification SOLUTIONS

- Certify®
  ASIC RTL Prototyping Software
- Identify™
  RTL Debugger

Physical Implementation SOLUTIONS

- Fortify™
  Family of Power Products
- Amplify® Physical Optimizer™
  Physical Synthesis for FPGAs

Synplicity
Simply Better Results

20 YEARS of ALTERA® INNOVATION

Top FPGA Design Challenges

As described by today's FPGA designers

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

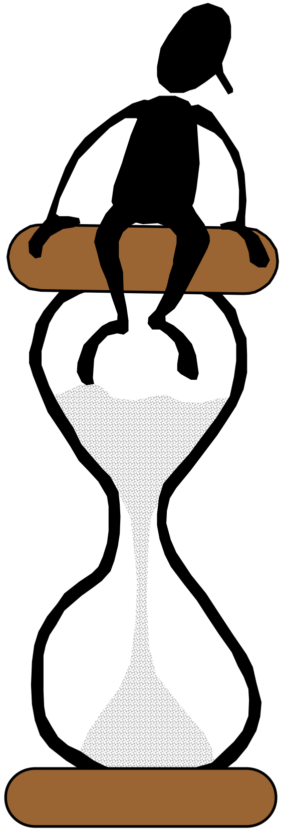
Achieving Performance Goals

(Synplify & Synplify Pro)

Performance Goals

- Synplify (Mappers) are architecture aware
- Optimizations are performed based on timing constraints
  - Hierarchical boundary optimization
- 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

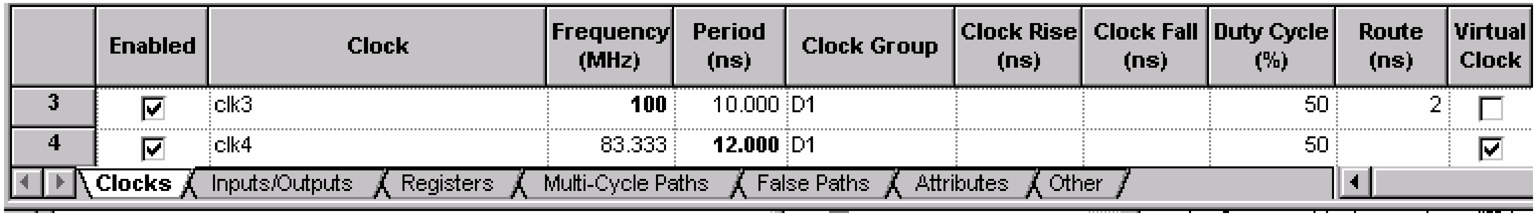
Synplicity
Simply Better Results

20 YEARS of ALTERA INNOVATION

Basic Clock Options

SCOPE®

- **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	☑	clk3	**100**	10.000	D1			50	2	□
4	☑	clk4	83.333	**12.000**	D1			50		☑

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

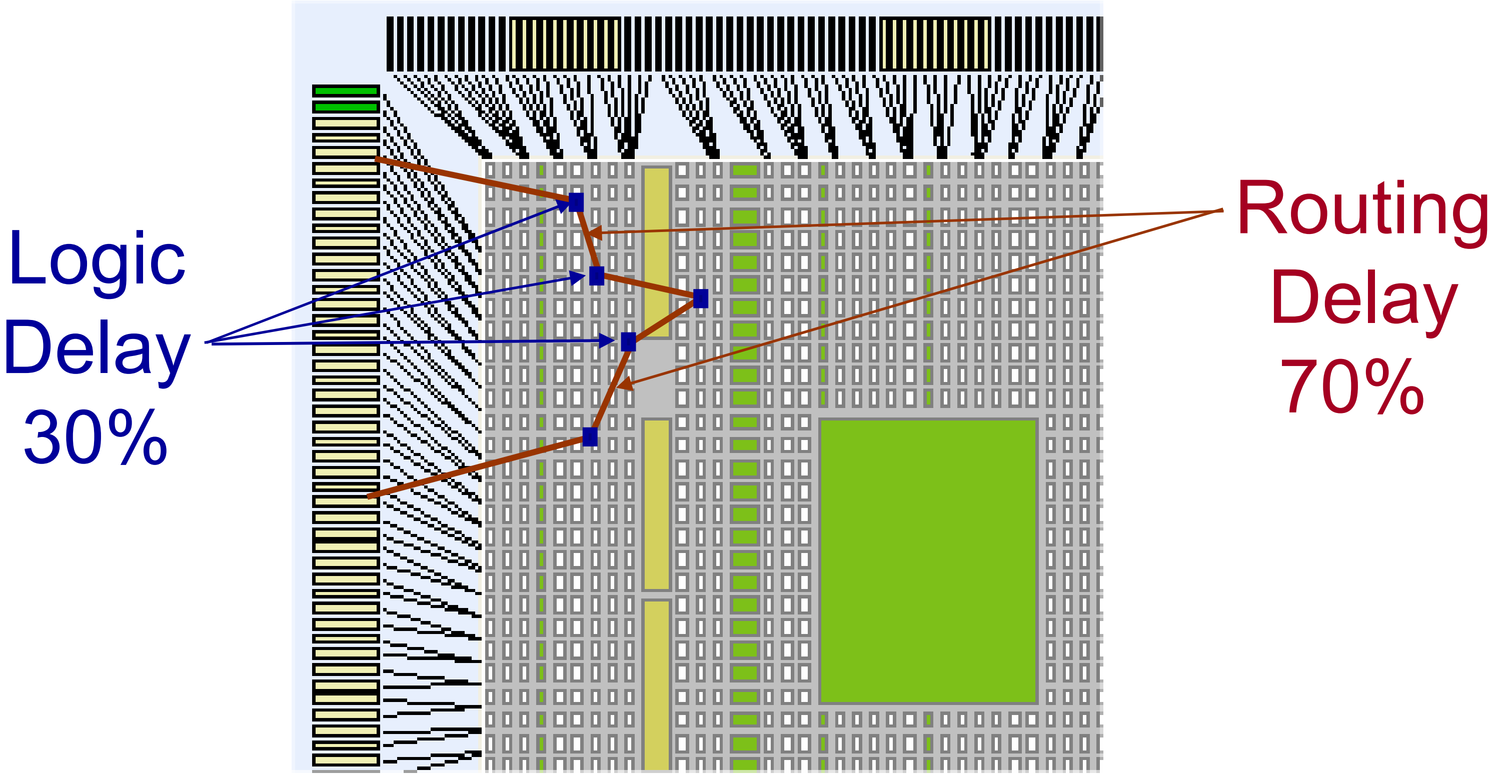
- Altera Clearbox support allows Synplify to perform realistic timing analysis, optimizations, and reporting
  - Synplify Pro only
- Mixed language support
  - Synplify Pro only

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



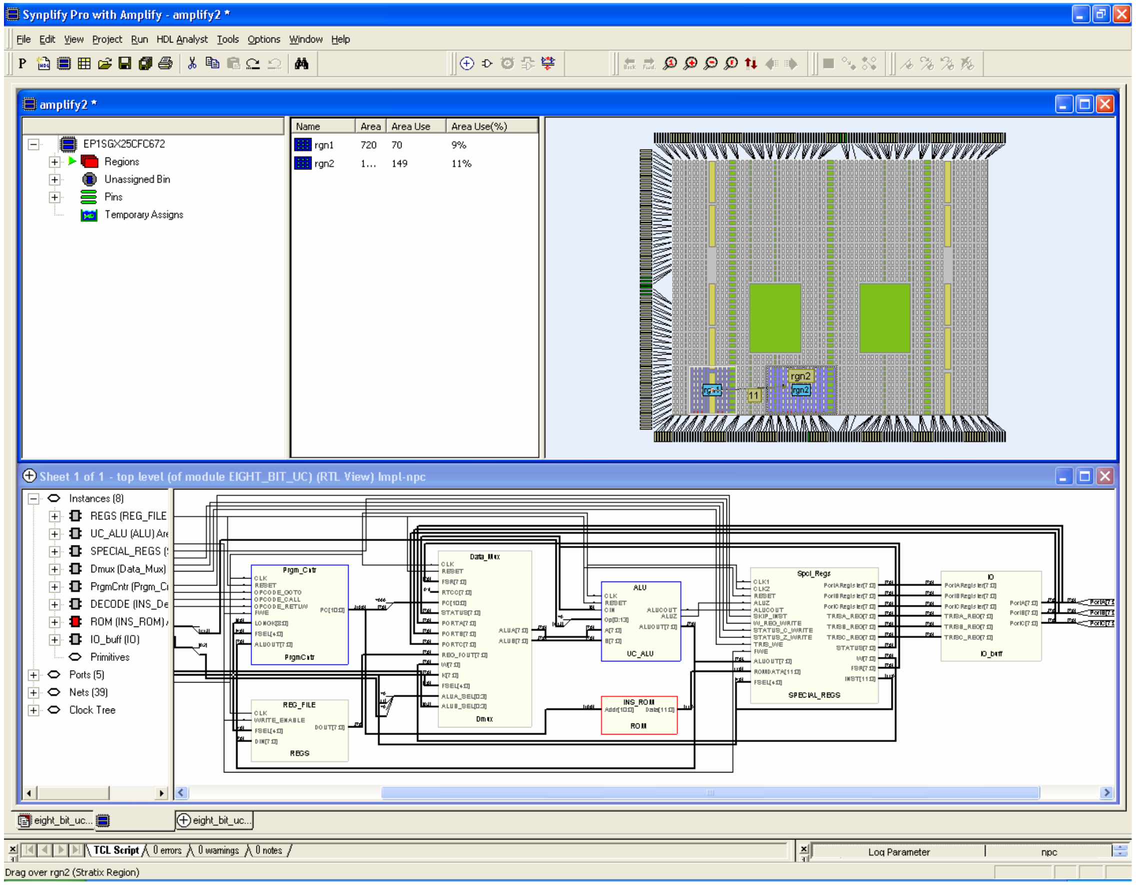
- Must incorporate physical information into synthesis
- The larger the design, the larger the problem

Synplicity
Simply Better Results

20 YEARS of
ALTERA
INNOVATION

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 *

- EP1SGX25CFC672
  - Regions
  - Unassigned Bin
  - Pins
  - Temporary Assigns

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) Are
  - SPECIAL_REGS (
  - Dmux (Data_Mux)
  - PrgmCntr (Prgm_C
  - DECODE (INS_De
  - ROM (INS_ROM) /
  - IO_buff (IO)
  - Primitives
- Ports (5)
- Nets (39)
- Clock Tree

TCL Script | 0 errors | 0 warnings | 0 notes

Log Parameter | npc

Drag over rgn2 (Stratix Region)

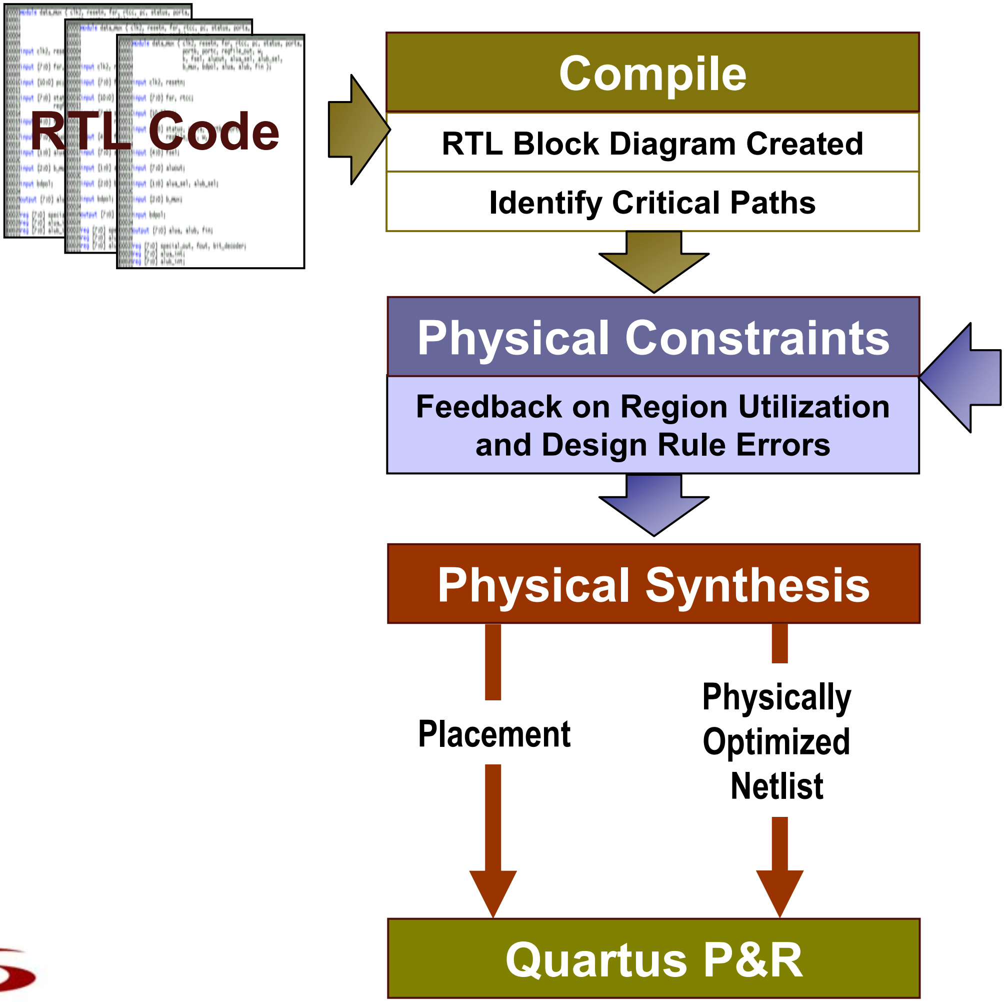
Synplicity
Simply Better Results

20 YEARS of ALTERA INNOVATION

When To Use Amplify

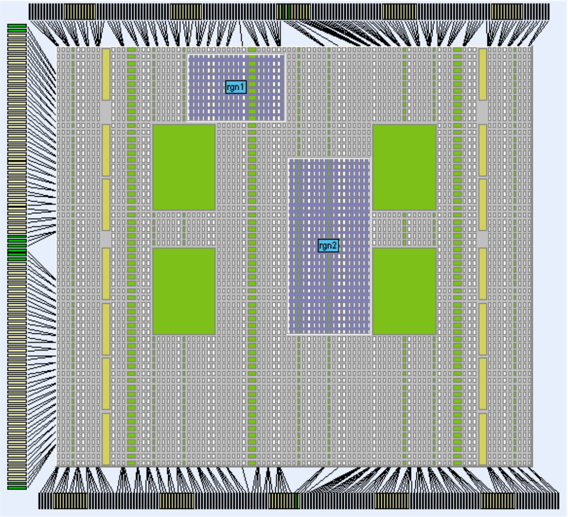
- For fast timing closure
- When you need the highest possible performance
- Need to reduce a speed-grade for cost
- When the majority of delay is in routing

Amplify – Interactive Flow



```
module data_mux ( clk2, resetn, fsr, rtcc, pc, status, porta,
                  portb, portc, regfile_out, w,
                  k, fsel, aluout, alus_sel, alub_sel,
                  b_mux, bdpol, alua, alub, fin );

input clk2, resetn;

input [7:0] fsr, rtcc;

input [10:0] pc;

input [7:0] status, porta, portb, portc,
            regfile_out, w, k;

input [4:0] fsel;

input [7:0] aluout;

input [1:0] alus_sel, alub_sel;

input [2:0] b_mux;

input bdpol;

output [7:0] alua, alub, fin;

reg [7:0] special_out, fout, bit_decoder;
reg [7:0] alua_int;
reg [7:0] alub_int;
```

RTL Code
Compile
RTL Block Diagram Created
Identify Critical Paths
Physical Constraints
Feedback on Region Utilization and Design Rule Errors
Physical Synthesis
Placement
Physically Optimized Netlist
Quartus P&R



Synplicity

Simply Better Results

20 YEARS of ALTERA INNOVATION

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)

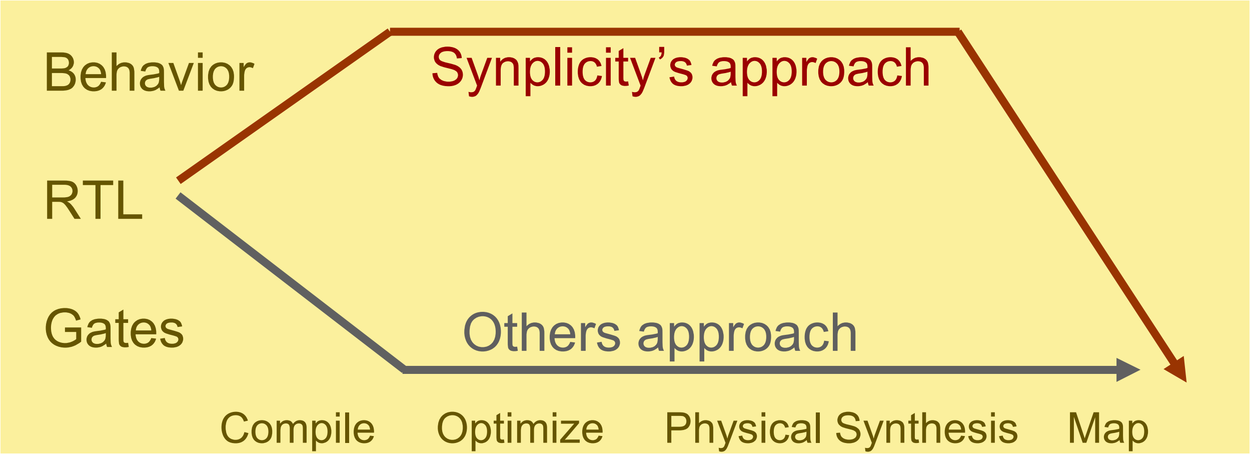
Synplicity

Simply Better Results

20 YEARS of ALTERA INNOVATION

Core Synthesis Technology

- **BEST™ – Behavior Extracting Synthesis Technology®**
  - Infers and optimizes behavior from RTL
  - Optimizes across hierarchical module boundaries
  - Integrated physical synthesis algorithms
  - Multi-million gate capacity
  - Extremely FAST - Unparalleled runtimes



Synplicity
Simply Better Results

20 YEARS of ALTERA INNOVATION

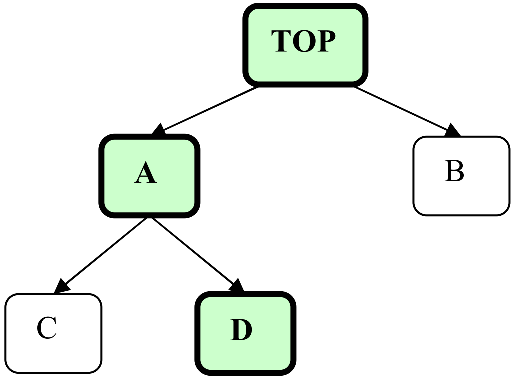
Managing Complex Designs

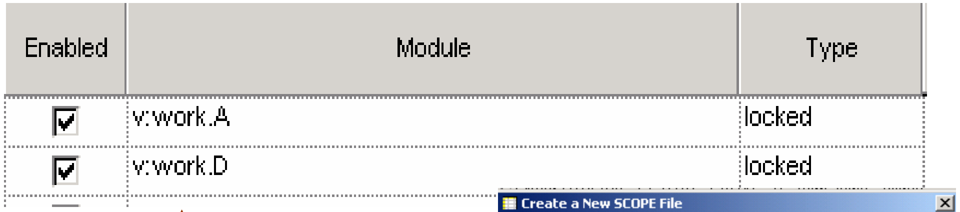
MultiPoint™

- **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 flov

LogicLock™

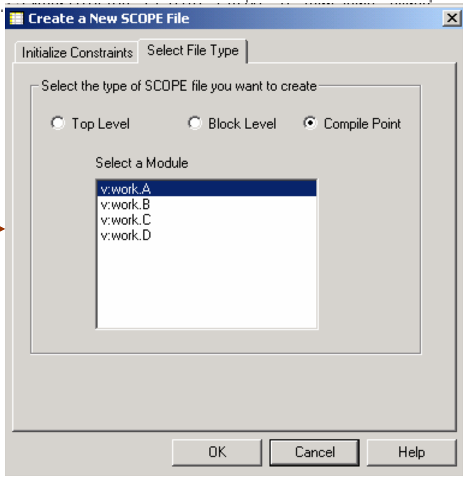
MultiPoint Flow





Enabled	Module	Type
☑	v:work.A	locked
☑	v:work.D	locked

Define compile points and compile point constraints



Create a New SCOPE File

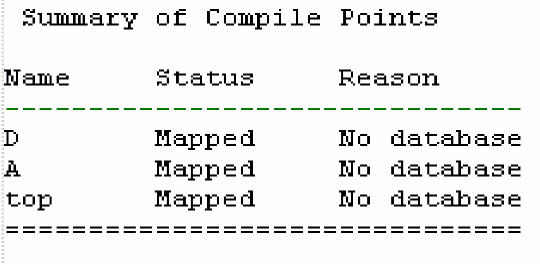
Select the type of SCOPE file you want to create

○ Top Level ○ Block Level ● Compile Point

Select a Module

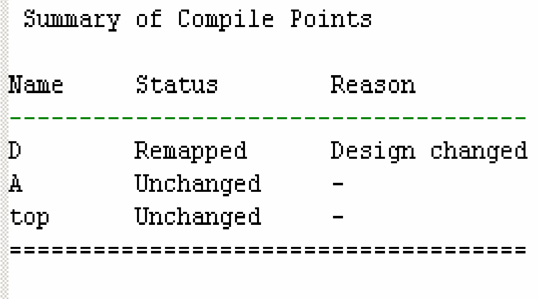
v:work.A
v:work.B
v:work.C
v:work.D

First Run

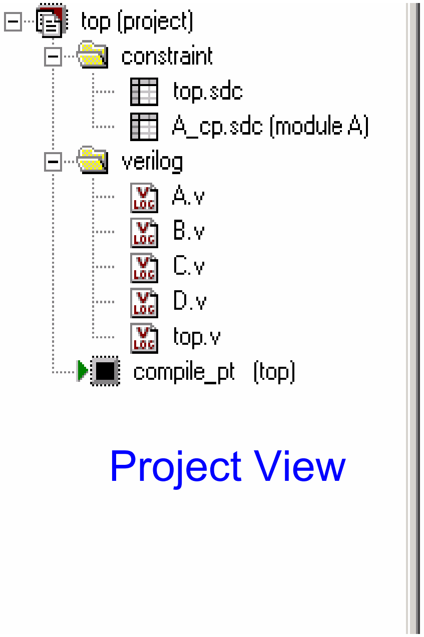


```
Summary of Compile Points

Name        Status      Reason
--------------------------------
D           Mapped      No database
A           Mapped      No database
top         Mapped      No database
================================
```

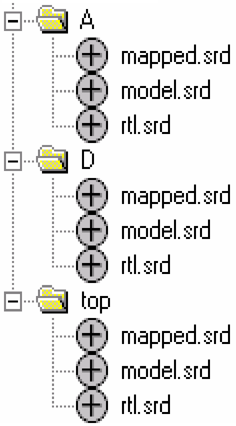
Second Run



```
Summary of Compile Points

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



```
top (project)
  constraint
    top.sdc
    A_cp.sdc (module A)
  verilog
    A.v
    B.v
    C.v
    D.v
    top.v
  compile_pt (top)
```



```
A
  mapped.srd
  model.srd
  rtl.srd
D
  mapped.srd
  model.srd
  rtl.srd
top
  mapped.srd
  model.srd
  rtl.srd
```

Project View

Synplicity
Simply Better Results

20 YEARS of ALTERA INNOVATION

MultiPoint Synthesis

Difference Based Incremental Synthesis

Re-synthesize a locked Compile Point module for:

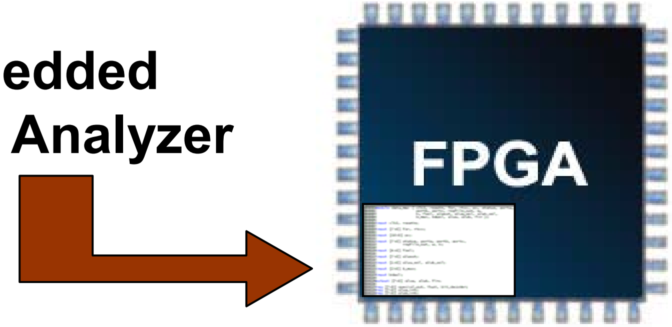
- A logic change in your RTL code
- Changes to constraints
  - Timing constraints change in the .sdc or Project View
  - Project settings change
    - FSM Compiler or Explorer
    - Retiming
    - Pipelining
- Re-synthesis is not based on time stamp

Debug (Identify)

Evolution of Hardware Debug

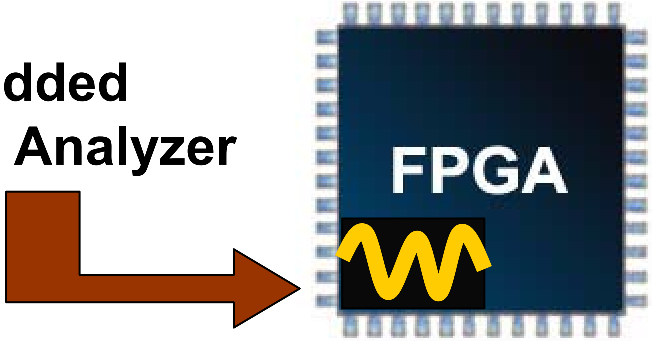
Identify
(Simulator-Like)

Embedded
HDL Analyzer

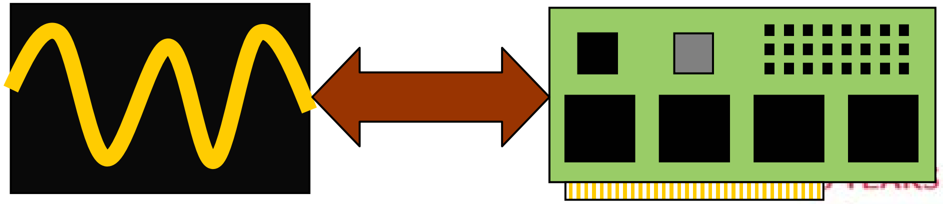


SignalTap
(Logic Analyzer-Like)

Embedded
Logic Analyzer



Logic Analyzer

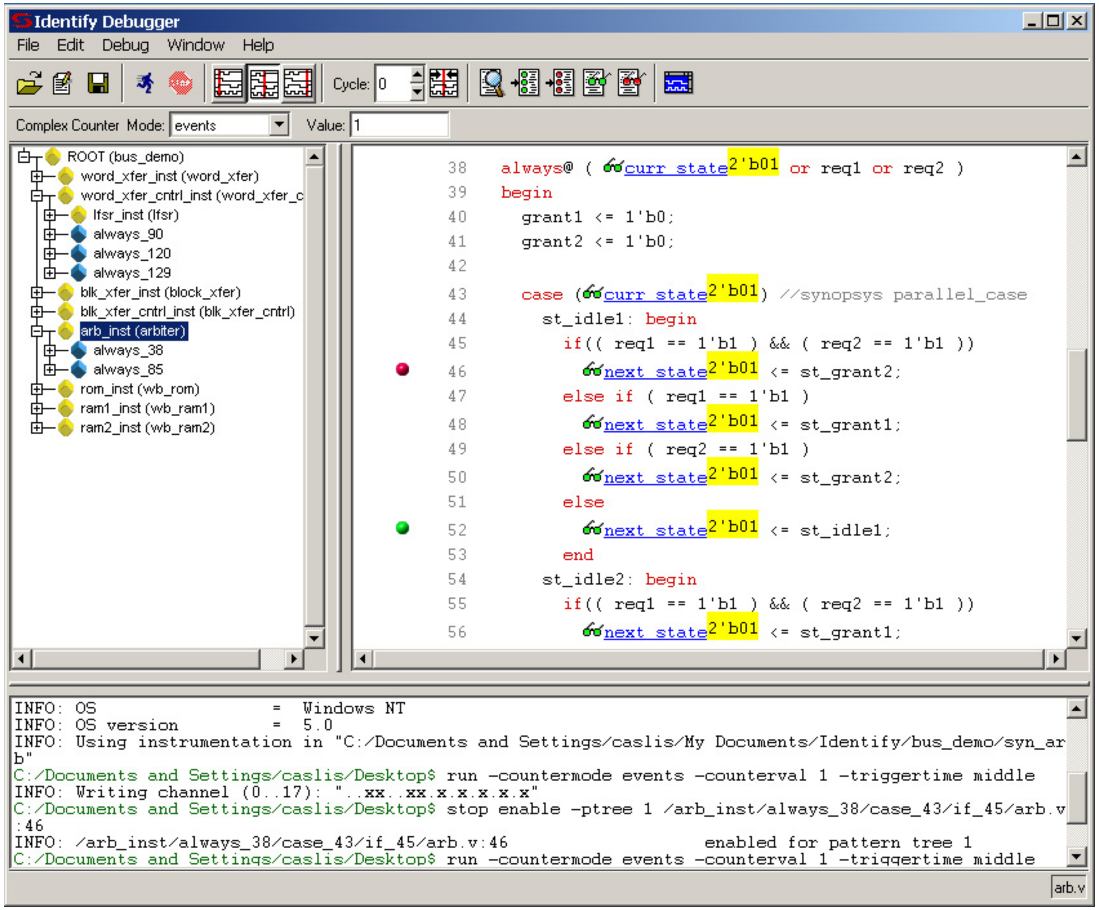


Synplicity
Simply Better Results

YEARS of
ALTERA
INNOVATION

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



Cycle: 0

Complex Counter Mode: events

Value: 1

```
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;
```

```
INFO: OS                  =  Windows NT
INFO: OS version          =  5.0
INFO: Using instrumentation in "C:/Documents and Settings/caslis/My Documents/Identify/bus_demo/syn_ar
b"
C:/Documents and Settings/caslis/Desktop$ run -countermode events -counterval 1 -triggertime middle
INFO: Writing channel (0..17): "..xx..xx.x.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
```

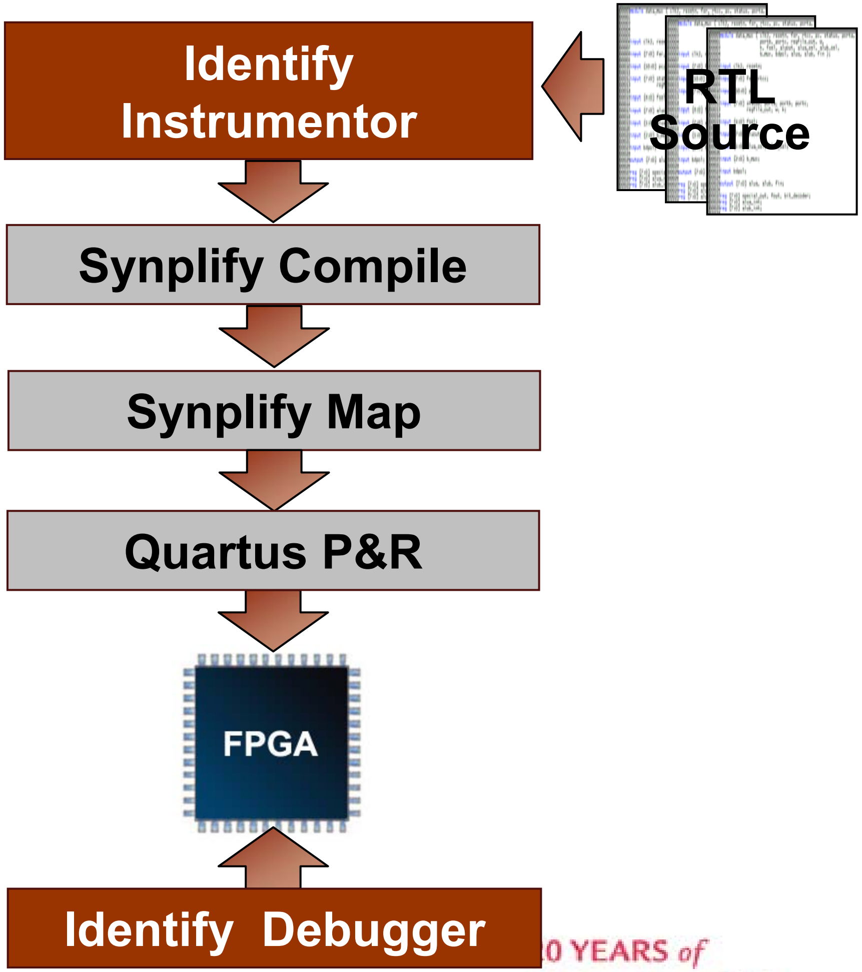
Synplicity

Simply Better Results

20 YEARS of ALTERA INNOVATION

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

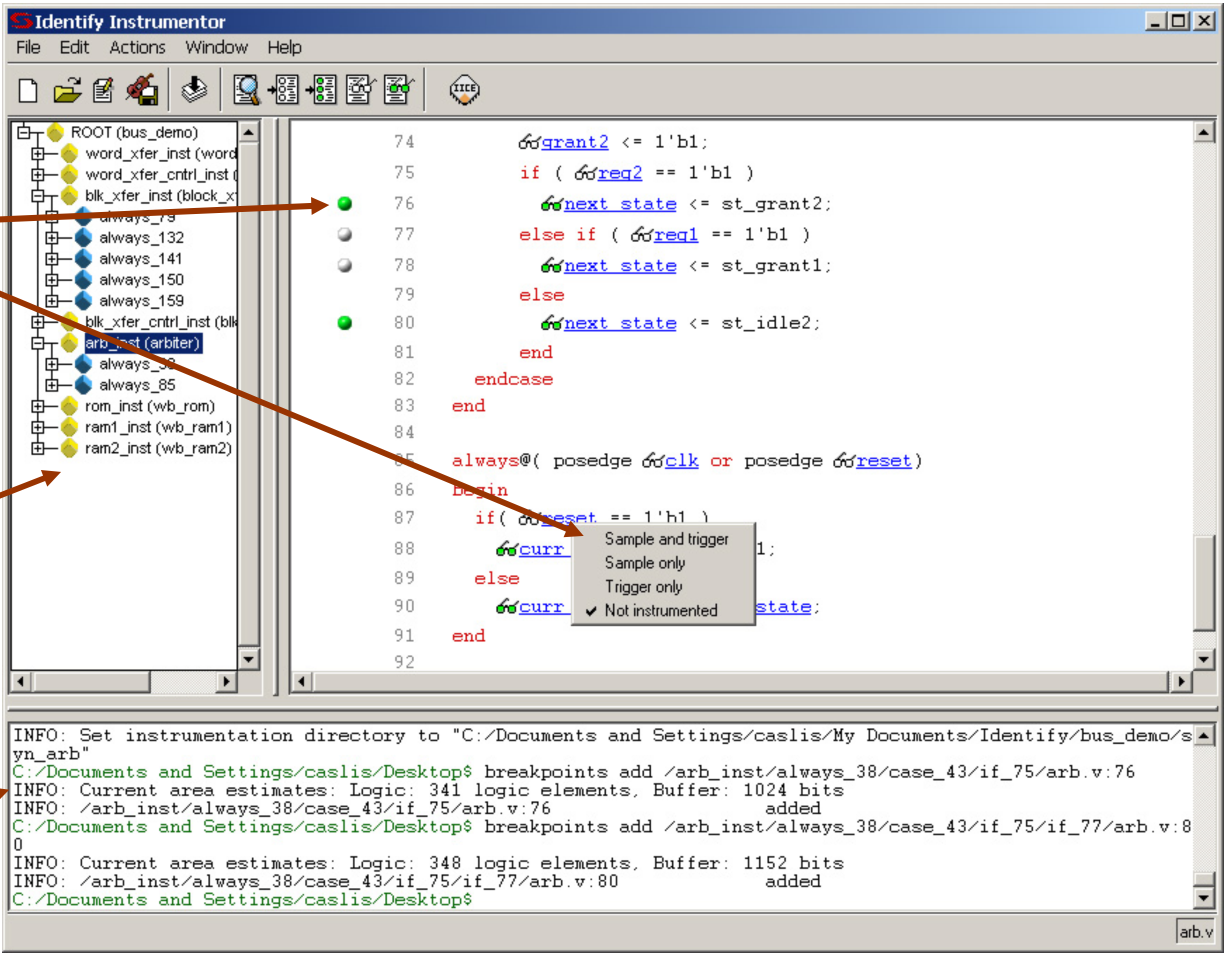


Instrumenting for Debug

Add Signal Visibility

Design Hierarchy

TCL Scripting Interface



```
74          grant2 <= 1'b1;
75          if ( req2 == 1'b1 )
76            next_state <= st_grant2;
77          else if ( req1 == 1'b1 )
78            next_state <= st_grant1;
79          else
80            next_state <= st_idle2;
81          end
82      endcase
83    end
84
85    always@( posedge clk or posedge reset)
86    begin
87      if( reset == 1'b1 )
88        curr                       1;
89      else
90        curr                       state;
91    end
92
```

```
INFO: Set instrumentation directory to "C:/Documents and Settings/caslis/My Documents/Identify/bus_demo/s
yn_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:8
0
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$
```

Identify Instrumentor

Synplicity

Simply Better Results

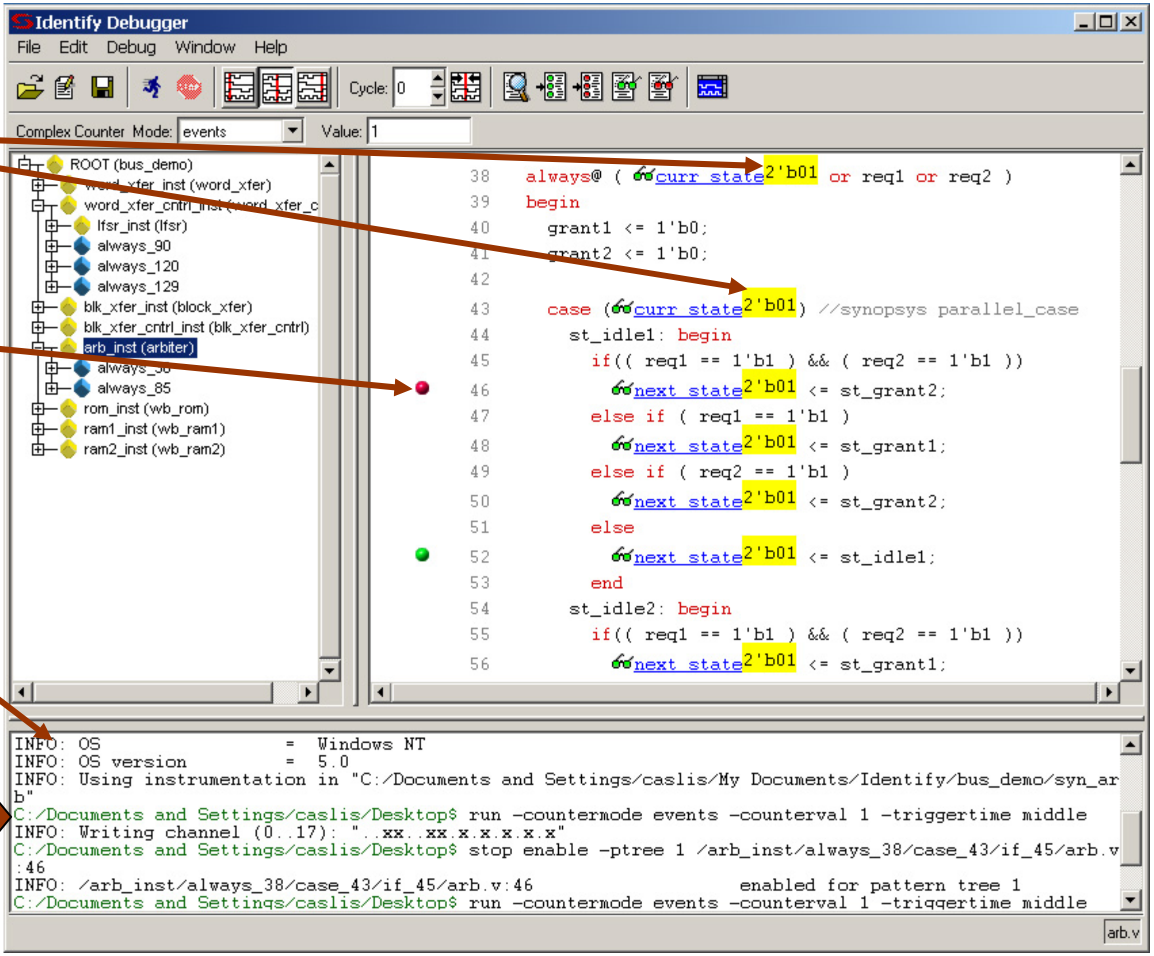
20 YEARS of ALTERA INNOVATION

Debugging Data From FPGA

Actual data from FPGA

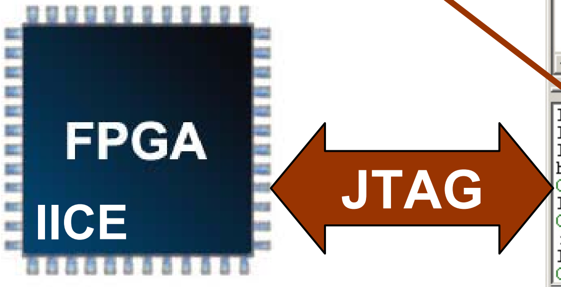
Click to enable triggers

Automate Debugger with Script



```
always@ ( curr_state2'b01 or req1 or req2 )
begin
  grant1 <= 1'b0;
  grant2 <= 1'b0;

  case (curr_state2'b01) //synopsys parallel_case
    st_idle1: begin
      if(( req1 == 1'b1 ) && ( req2 == 1'b1 ))
        next_state2'b01 <= st_grant2;
      else if ( req1 == 1'b1 )
        next_state2'b01 <= st_grant1;
      else if ( req2 == 1'b1 )
        next_state2'b01 <= st_grant2;
      else
        next_state2'b01 <= st_idle1;
      end
    st_idle2: begin
      if(( req1 == 1'b1 ) && ( req2 == 1'b1 ))
        next_state2'b01 <= st_grant1;
```

```
INFO: OS                    =  Windows NT
INFO: OS version            =  5.0
INFO: Using instrumentation in "C:/Documents and Settings/caslis/My Documents/Identify/bus_demo/syn_ar
b"
C:/Documents and Settings/caslis/Desktop$ run -countermode events -counterval 1 -triggertime middle
INFO: Writing channel (0..17): "..xx..xx.x.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
```



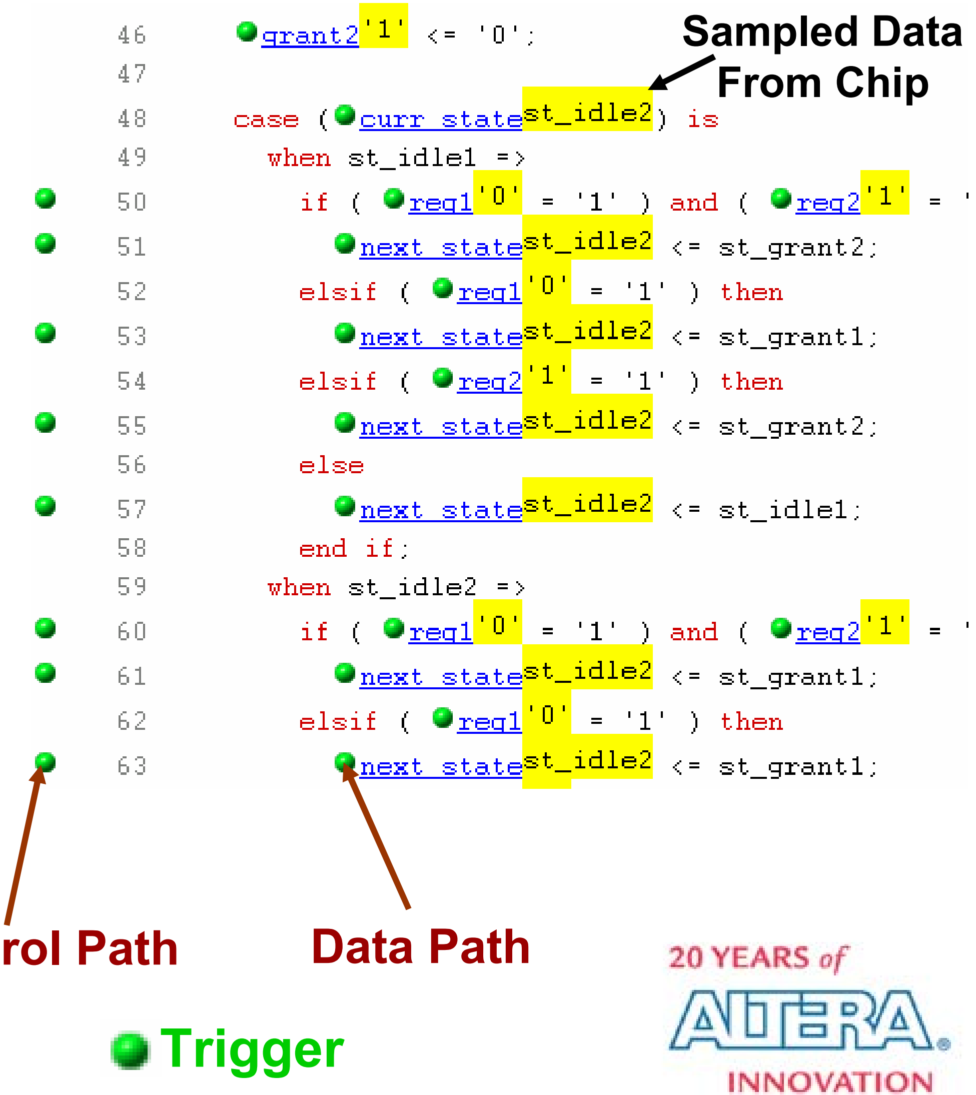
Synplicity
Simply Better Results

Identify Debugger

20 YEARS of ALTERA INNOVATION

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



```
       ●grant2'1' <= '0';

       case (●curr_statest_idle2) is
         when st_idle1 =>
●          if ( ●req1'0' = '1' ) and ( ●req2'1' = '
●            ●next_statest_idle2 <= st_grant2;
           elsif ( ●req1'0' = '1' ) then
●            ●next_statest_idle2 <= st_grant1;
           elsif ( ●req2'1' = '1' ) then
●            ●next_statest_idle2 <= st_grant2;
           else
●            ●next_statest_idle2 <= st_idle1;
           end if;
         when st_idle2 =>
●          if ( ●req1'0' = '1' ) and ( ●req2'1' = '
●            ●next_statest_idle2 <= st_grant1;
           elsif ( ●req1'0' = '1' ) then
●            ●next_statest_idle2 <= st_grant1;
```

Sampled Data From Chip

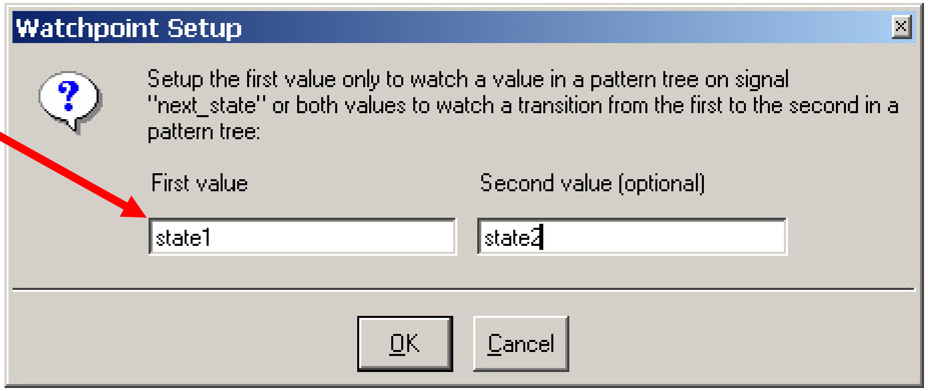
Control Path

Data Path

● Trigger

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;
```



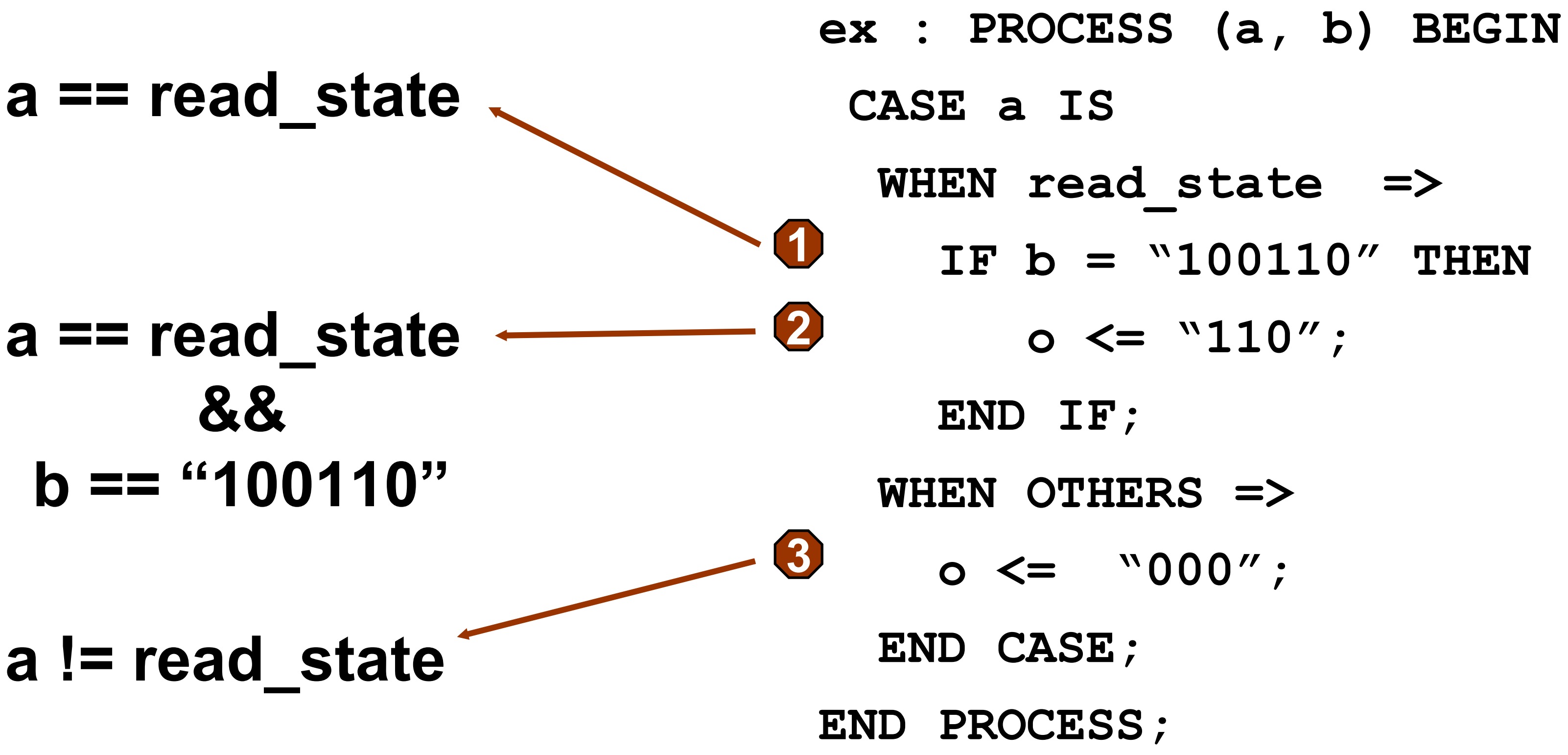
Watchpoint Setup

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

Setting Triggers on Control Path



```
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;
```

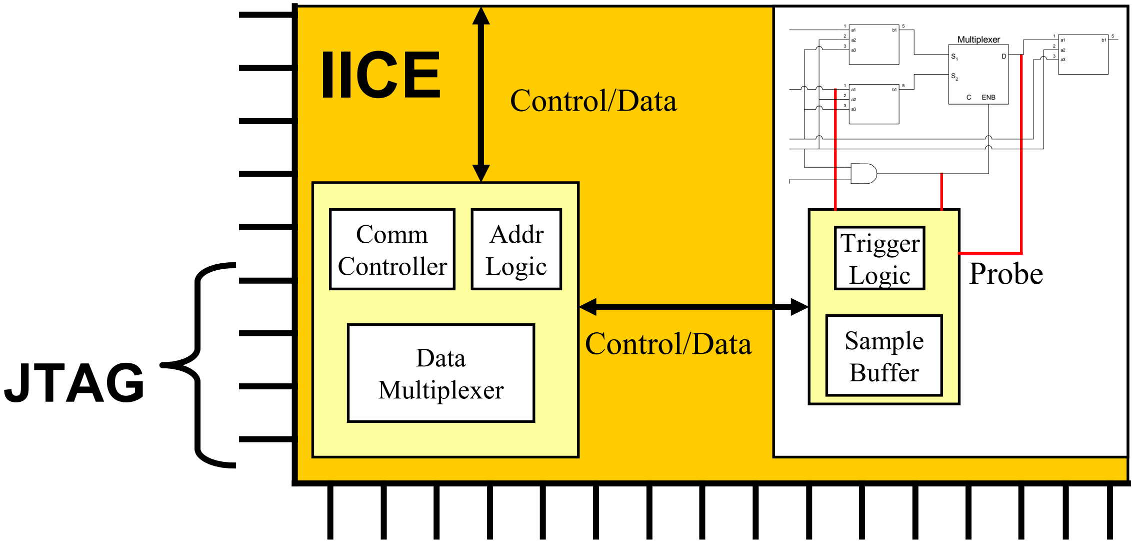
1 a == read_state

2 a == read_state && b == “100110”

3 a != read_state

Intelligent In-Circuit Emulator (IICE™)

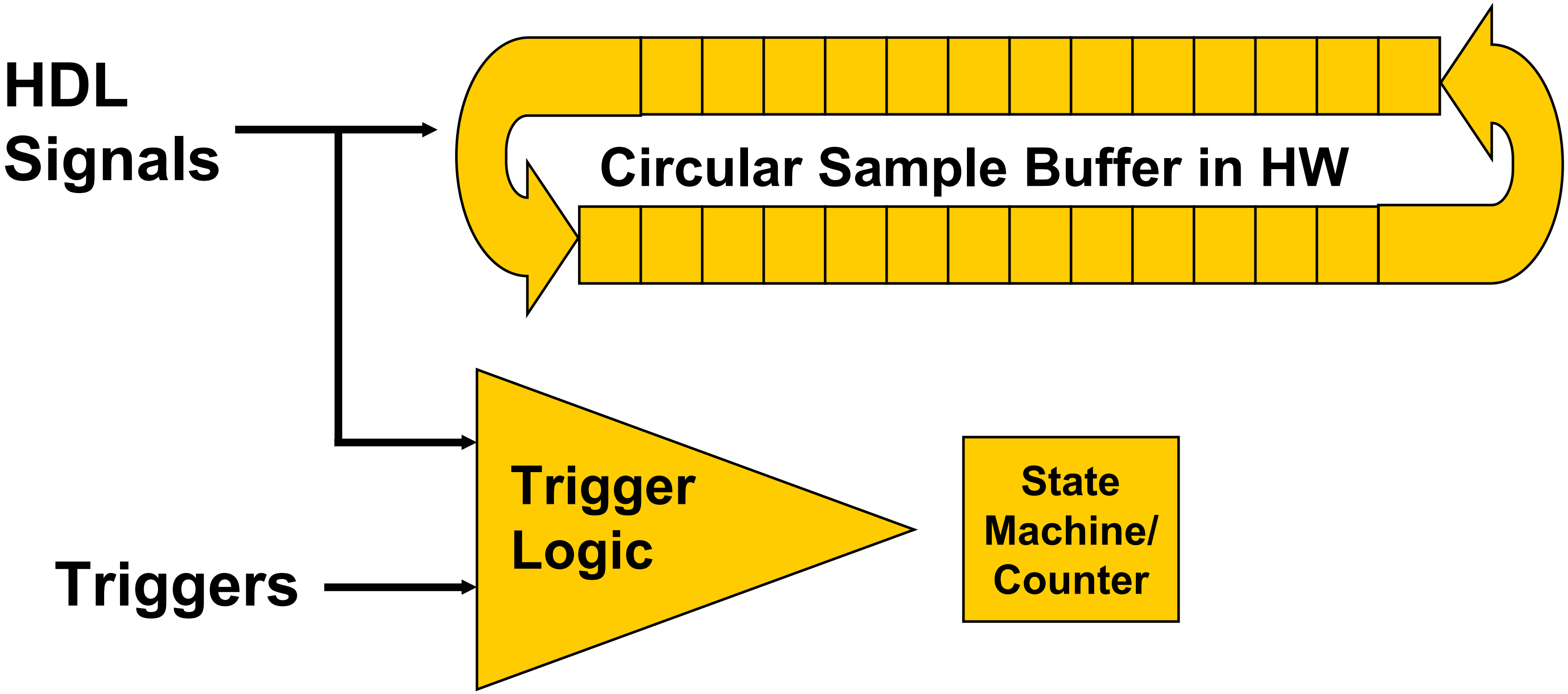
- Inserted logic used by Instrumentor & Debugger
- Uses dedicated JTAG pins or user-selected pins
- Includes controller, triggering logic, & data storage buffer



Synplicity
Simply Better Results

20 YEARS of ALTERA INNOVATION

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

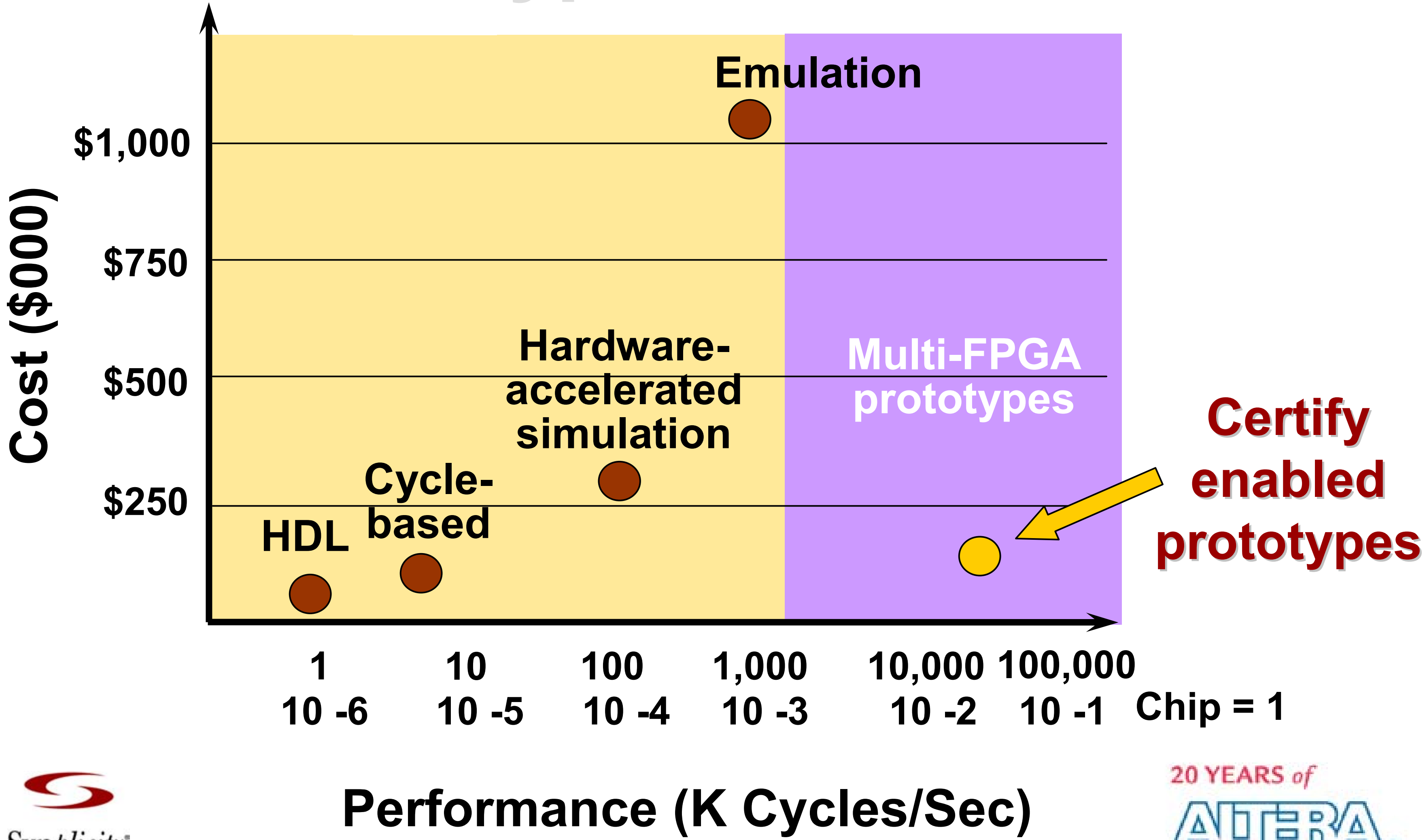
“In just our first six weeks with the product, we used it to find and fix bugs in three of our designs. In each case the process was completed within a day. It would have taken 10 to 20 times longer using traditional test-bench methods.”

-Richard Grenier

Director of ASIC Development

Multi-FPGA Prototyping (Certify)

Higher Speed & Lower Cost using FPGA Prototypes



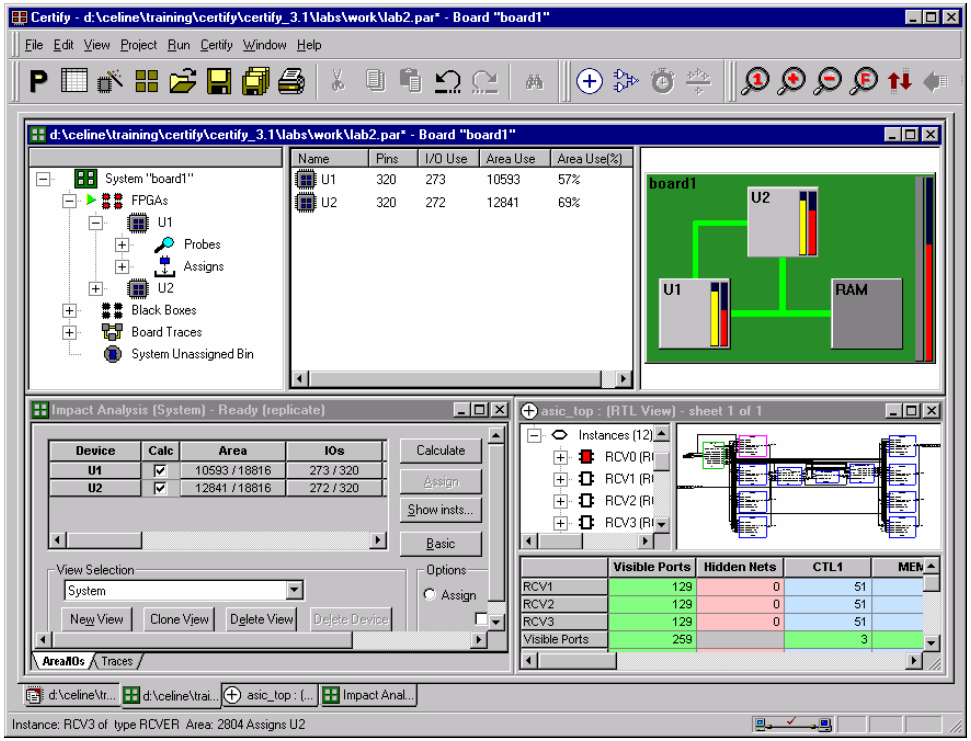
Synplicity
Simply Better Results

20 YEARS of ALTERA INNOVATION

Prototyping Challenges

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

FPGA-Based ASIC Prototypes



Certify - d:\celine\training\certify\certify_3.1\labs\work\lab2.par* - Board "board1"

d:\celine\training\certify\certify_3.1\labs\work\lab2.par* - Board "board1"

- System "board1"
  - FPGAs
    - U1
      - Probes
      - Assigns
    - U2
  - Black Boxes
  - Board Traces
  - System Unassigned Bin

Name	Pins	I/O Use	Area Use	Area Use(%)
U1	320	273	10593	57%
U2	320	272	12841	69%

Impact Analysis (System) - Ready (replicate)

Device	Calc	Area	IOs
U1	☑	10593 / 18816	273 / 320
U2	☑	12841 / 18816	272 / 320

asic_top : (RTL View) - sheet 1 of 1

- Instances (12)
  - RCV0 (R
  - RCV1 (R
  - RCV2 (R
  - RCV3 (R

	Visible Ports	Hidden Nets	CTL1	MEN
RCV1	129	0	51	
RCV2	129	0	51	
RCV3	129	0	51	
Visible Ports	259		3	

Instance: RCV3 of type RCVER Area: 2804 Assigns U2

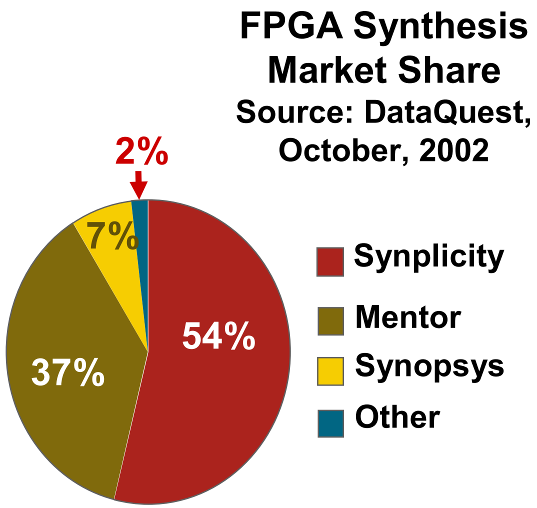
- Highest performing ASIC prototypes for
  - Functional verification
  - HW/SW co-verification
- Automatic RTL partitioning, I/O sharing, and more
- Supports all prototyping hardware including off-the-shelf boards
- Adopted by Philips, TI, LSI Logic and others

Synplicity
Simply Better Results

20 YEARS of ALTERA INNOVATION

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



Synplicity
Simply Better Results

20 YEARS of ALTERA INNOVATION

Synplicity®

Simply Better Results

END