

Mentor Graphics Veloce™ Systems

- Up to:
 - 10,000 times the performance of software simulation
 - 512Mgate designs
 - 4 I/O boxes
 - 15 pure asynchronous clock domains
- Hardware
 - Crystal Chip™: 96K Configurable Programmable Block (CPB) able to emulate up to 500K user gates
 - 16 Crystal Chips combined on a board → emulate 8Mgates
 - Up to 16 boards in a system (PSU ECE Velocesolo1 has one board)
 - Family of products built with common board, software, universal user interface
 - Veloce: Maximus (512Mgates), Grande(256Mgates), Quattro (128Mgates), Solo (16Mgates), Trio
 - Veloce2 family (2012) includes a Double Maximus, Maximus, and Quattro
- Trace system: up to 64k cycles of data before upload (can be selective)
- Software: Compilers, runtime system, debuggers, ...

•1

Online reference material: Mentor Graphics Veloce User's Guide

- Ch 1: Introduction
- Ch 2: Getting Started
- Ch 3: Graphical User Interface
- Ch 4: Project Creation/Design Preparation
- Ch 5: Design Compilation
- Ch 6: Triggers
- Ch 7: RTLC Synthesis
- Ch 8: Integrated Flows and Solutions
- Ch 9: Emulation
- Ch 10: Debugging
- Ch 11: Reference Guide for Compiler and Runtime Options
- ...

2

StandAlone Mode

Emulation using Mentor Graphics Veloce

Compile/synthesis on velocesolo
Emulation on velocesolo1

•

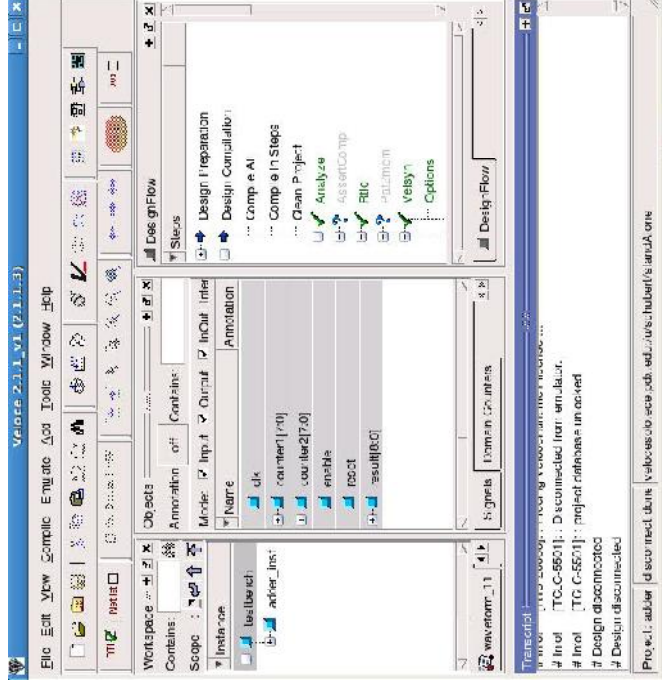
velocegui usage flow

- Create new project/open project
 - Add files to project
- Compile design
 - Multiple steps (generic to emulator board specific)
- Connect to Emulator
 - Run Emulation
 - Upload traces
- Disconnect from Emulator
- View results using waveform viewer
- Save project and quit

Many Mentor Graphics Emulator products and configurations

- Maximus, Grande, Quattro, Solo, Trio

Common software (velocegui) will require platform specific parameters



- Windows can be undocked
- Actions not currently enabled appear faded
- Some duplication of functions between, main menu functions, action buttons and DesignFlow submenus






Docked Windows

- Workspace
- Objects
- Design Flow
- Transcript

Other windows

- Emulation Control/Debug
 - Emulation-Emulation Control
 - multiple tabs
- Waveform viewer
 - View-Debug Window-Wave

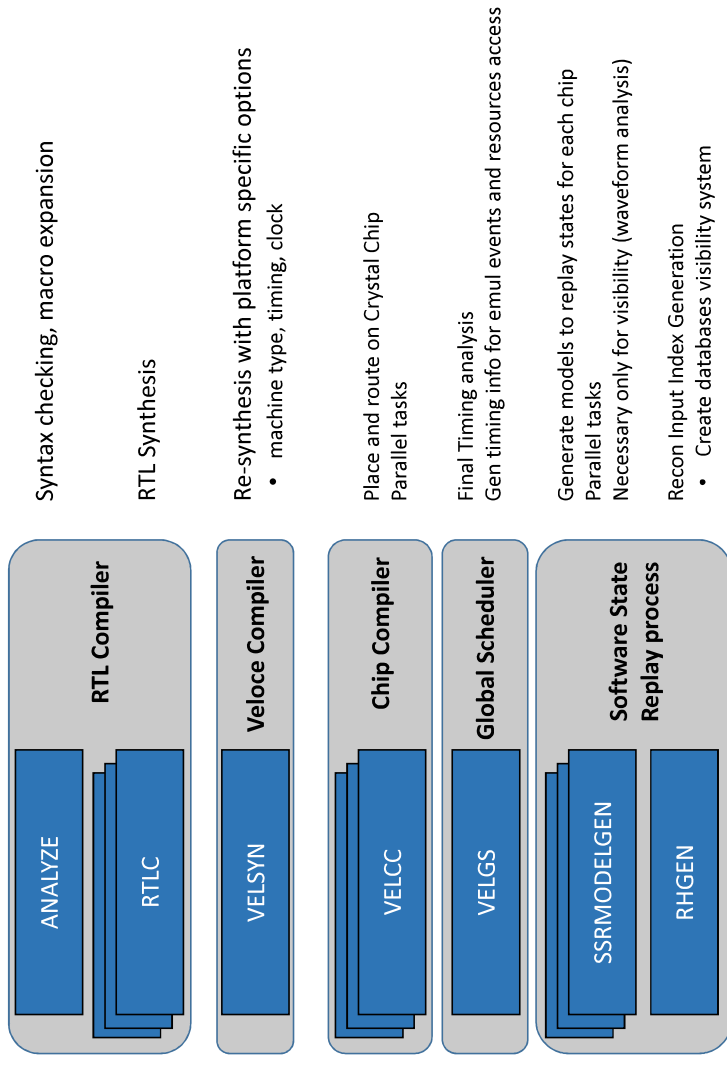
Action Buttons

-  Connect/Disconnect
-  Emulation Control
-  Waveform upload
-  Waveform viewer
-  Stop the emulation

Design Flow Window

- **Design Preparation**
 - Create Project
 - Open Project
 - Save Project
- **Design Compilation**
 - Analyze:
 - Rtlc:
 - Velsyn:
 - Velcc
 - Velgs
 - Ssrcc
- **Emulation**
 - Emulation Setup
 - Connect/Disconnect
 - Configure
 - Run

Veloce Compile Flow



6

Design Compilation

- Analyze: syntax, macro expansion
 - error only occurs the first time run, before you define a clk
Warning [velcomp-2553]: Unable to open file vmw.clk
Error accessing vmw.clk: No such file or directory
- Rtlc: RTL synthesis
 - need to set *testbench* as top in workspace window
- Velsyn: system specific settings needed:
 - Options
 - Target: Solo(comodel(D1SC))
 - Board #1
 - Timing Setup
 - Clock Domain: Add clock domain, add clock (clk) and set waveform
 - Data I/O timing: select reset, enable, and result[8:0]
- Velcc
- Velgs
- Ssrc

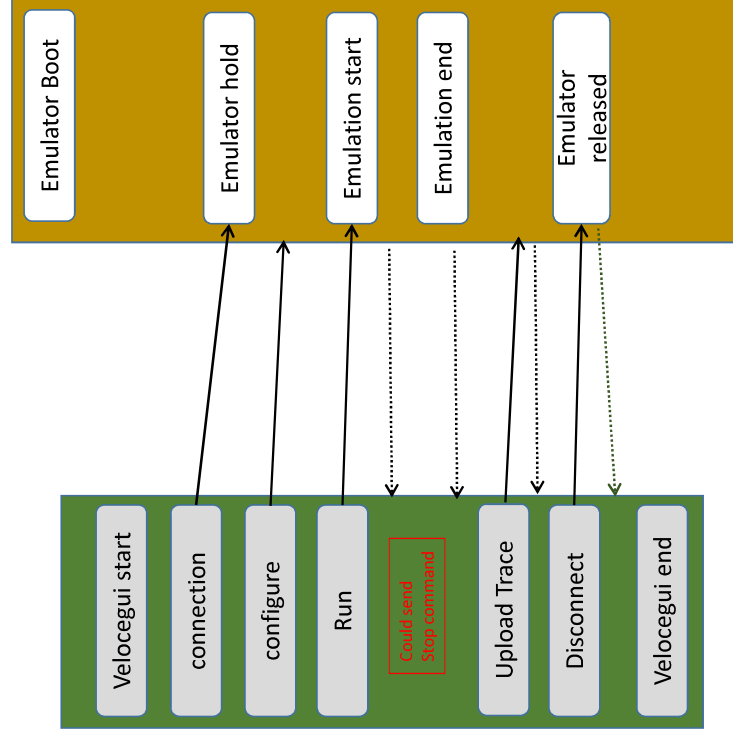
7

Emulation

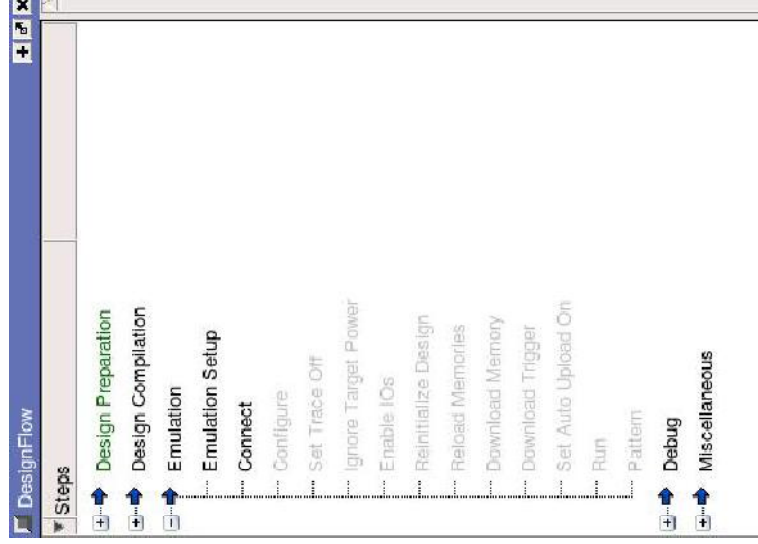
- Setup
 - Set emulator name: velocesolo1
- Connect
- Configure
- Now type commands in transcript window
 - Run a while, force reset and enable, run for a while, force reset low. Run for 60000 cycles
- Upload trace
 - Select emulate->emulate control
 - Select wave tab, click upload waveform, give a name, upload
- **Disconnect**
 - Once disconnected, the option will be in light grey

8

Server-Emulator communication



9

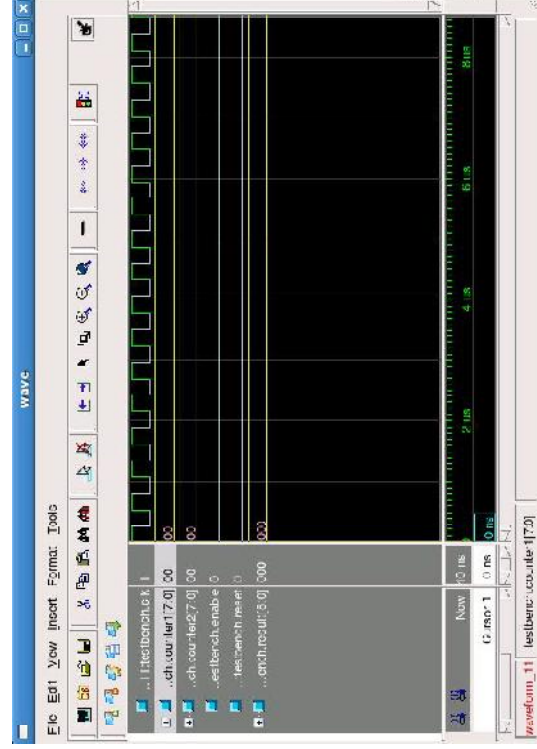


10

Waveform Viewer Window

- To view all signals in the waveform viewer:
 - Go Main menu and select: View-debug windows-wave
 - In Wave window select: File-specify dataset< select trace
 - In Objects: select *all objects* >add to wave-selected signals>

< left click
> right click



11

```

module adder (result, a, b, clk, reset, enable);
    input [7:0] a;
    input [7:0] b;
    input clk;
    input reset;
    input enable;

    output [8:0] result;
    reg [8:0] result;

    always @(posedge clk)
    begin
        if (reset)
            result = 0;
        else if (enable)
            result = a + b;
    end
endmodule

```

12

```

module testbench (clk, reset, enable, result );

```

```

    input clk;
    input reset;
    input enable;
    output [8:0] result;

    reg [7:0] counter1;
    reg [7:0] counter2;

```

```

    always @(posedge clk or posedge reset)
    begin

```

```

        if (reset)
            begin
                counter1 = 8'b0000_0000;
                counter2 = 8'b0000_0000;
            end
        else if (enable)
            begin

```

```

                if (counter1 == 255)
                    counter1 = 0;

```

```

                else
                    counter1 = counter1 + 1;
                    if (counter1 == 0)
                        counter2 = counter2 + 1;

```

```

            end // else if (enable)
        end // always

```

```

        adder_adder_inst (.result(result), .a(counter1), .b(counter2),
            .clk(clk), .reset(reset), .enable(enable));

```

```

    endmodule

```

```

> run 100 // Run 100 cycles
> reg setvalue testbench.reset 1 //Force reset=1
> reg setvalue testbench.enable 0 //Force enable=0
> run 100 // Run 100 more cycles
> reg setvalue testbench.reset 0 //Force reset=0
> reg setvalue testbench.enable 1 //Force enable=1
> run 60000 // Run 60000 cycles

```

13