
# Methods of Using Processor Cores To Optimize System Performance & Cost

# The Designer's Challenge

- How to partition between HW and SW to get the best cost and performance?
  - Hardware is fast and cost grows with complexity
  - Software is inexpensive and performance degrades with complexity
  - Some functions are naturally meant to be implemented in processors

# Accelerating Software Example

## Inverse Discrete Cosine Transform

- A typical software development starts as 100% “C”
- Performance profiling determines the critical paths
  - IDCT function is in the critical path
  - The system requirements are off by an order of magnitude

***Identify opportunities for system performance acceleration***

# If All Code is Run in Software...

Main Software
Calling IDCT Function

```
#include "mb_interface.h"

int main() {
   int indata[8], outdata[8];
   ...
  xil_idct (indata, outdata);
   ...
}
```

IDCT
Software Library

```
xil_idct (int *indata, int *outdata) {
   int i, j, k;
   int sum;

   for (j = 0; j < 8; j++) {
      sum = 0;
      for (i = 0; i < 8; i++)
         sum += (indata[i] * idct_constants[j][i]);

      outdata[j] = (sum >> 2);
      outdata[j] = (int) DESCALE (outdata[j], 10);
   }
```

- Entire IDCT function takes 1144 clock cycles to execute
- 896 clock cycles (14*64) reside in the inner loop

***This implementation is an order of magnitude too slow***

# Traditional Acceleration
## Define New User Instruction

Main Software
Calling IDCT Function

```
#include "mb_interface.h"

int main() {
  int indata[8], outdata[8];
  ...
  xil_idct (indata, outdata);
  ...
}
```

Add User-Defined
MAC Instruction

```
xil_idct (int *indata, int *outdata) {
  int i, j, k;
  int sum;

  for (j = 0; j < 8; j++) {
    sum = 0;
    for (i = 0; i < 8; i++)
      asm ("MAC %0, %1"
           : d (indata[i]) : d (idct_constants[j][i]));
    asm ("READMAC %0" : "=d" (sum));
    outdata[j] = (sum >> 2);
    outdata[j] = (int) DESCALE (outdata[j], 10);
  }
```

- Saving 3 clock cycles per iteration in the inner loop
- IDCT function runs only 1.2x faster - still too slow

***Large effort for a small return***

# Xtreme Processing Acceleration

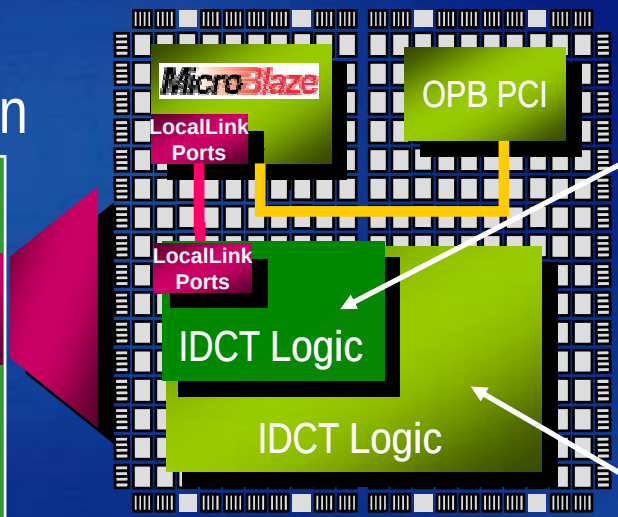
Main Software Calling IDCT Function

```
#include "mb_interface.h"

int main() {
    int indata[8], outdata[8];
    …
    xil_idct_hw(indata, outdata);
    …
}

void xil_idct_hw(int indata[8], int outdata[8] {
    microblaze_bwrite_datafsl (indata[0], 0);
    microblaze_bwrite_datafsl (indata[1], 0);
    microblaze_bwrite_datafsl (indata[2], 0);
    microblaze_bwrite_datafsl (indata[3], 0);
    microblaze_bwrite_datafsl (indata[4], 0);

    microblaze_bread_datafsl (outdata[0], 0);
    microblaze_bread_datafsl (outdata[1], 0);
    microblaze_bread_datafsl (outdata[2], 0);
    microblaze_bread_datafsl (outdata[3], 0);
    microblaze_bread_datafsl (outdata[4], 0);
}
```




IDCT Function in HW through LocalLink

Option 1:

Use small (200 LUTs) implementation of IDCT in HW

- only 107 clock cycles
- 10x faster

Option 2:

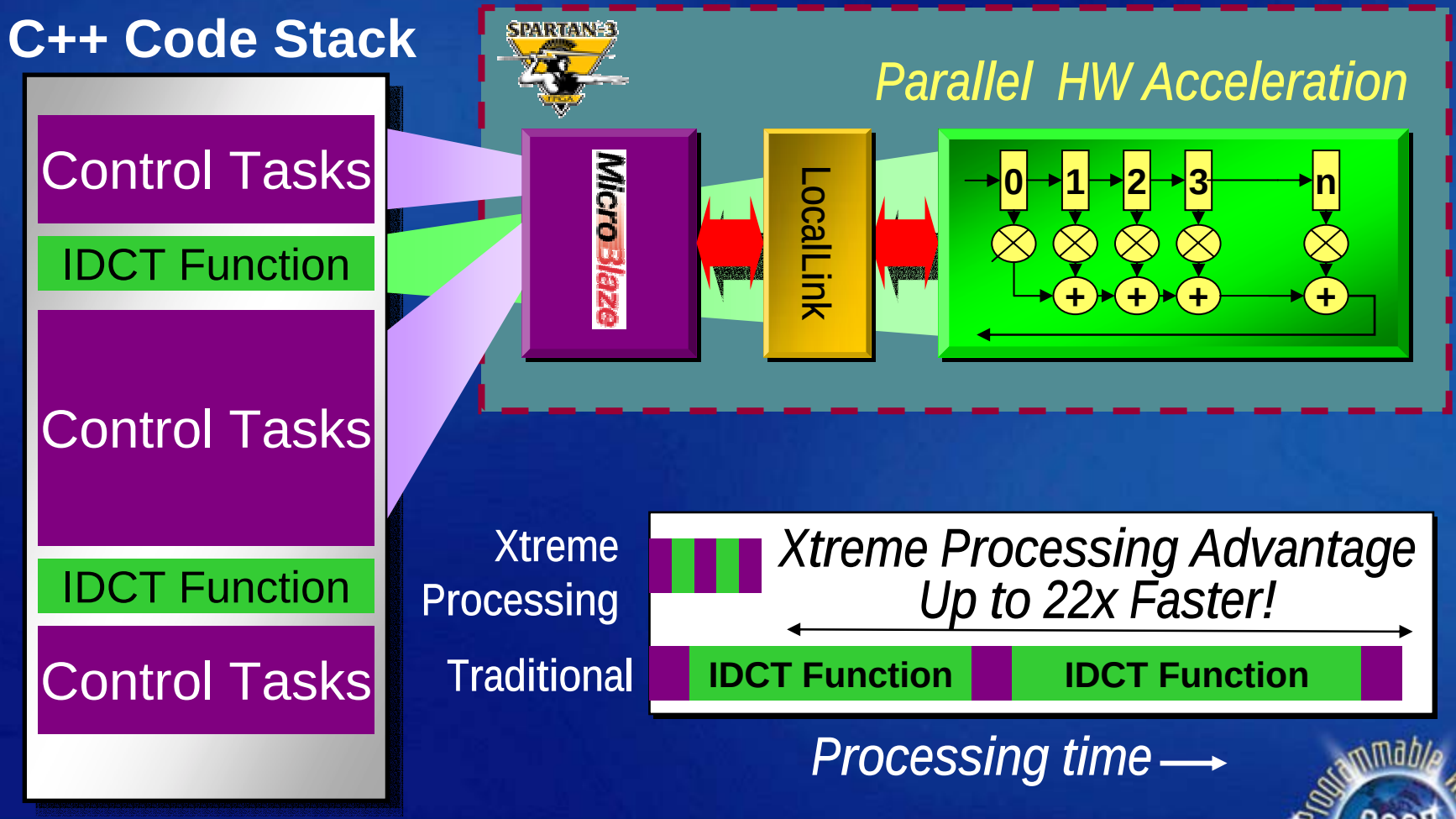
Use fast (1,600 LUTs) implementation of IDCT in HW

- only 52 clock cycles
- 22x faster

***This achieves our performance goals!***

# MicroBlaze LocalLink Accelerates System Performance

Processor(s) and HW on a single programmable platform




XILINX®

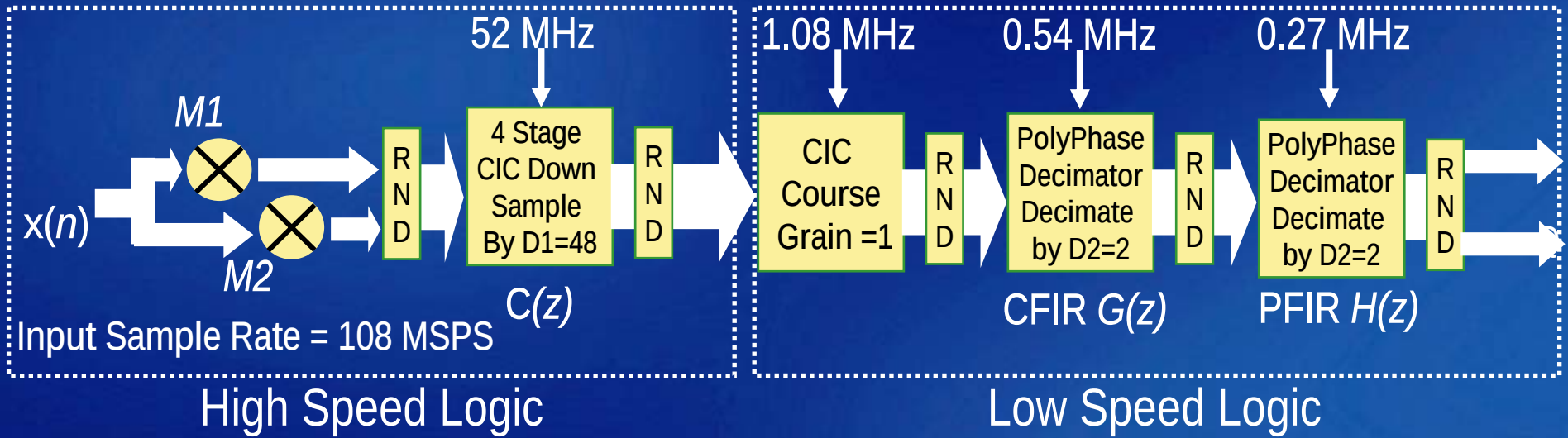
# Optimizing Hardware Example
## Digital Down Converter

- Hardware is ideal for high performance
  - Developing complex algorithms in hardware can be challenging
  - Low sample rate processing HW tends to be overkill
- Software is cost effective for low performance requirements
  - Time sharing of resources leads to dramatic silicon area savings
  - SW handles significantly more complex algorithms easier

***Identify opportunities for cost reduction***

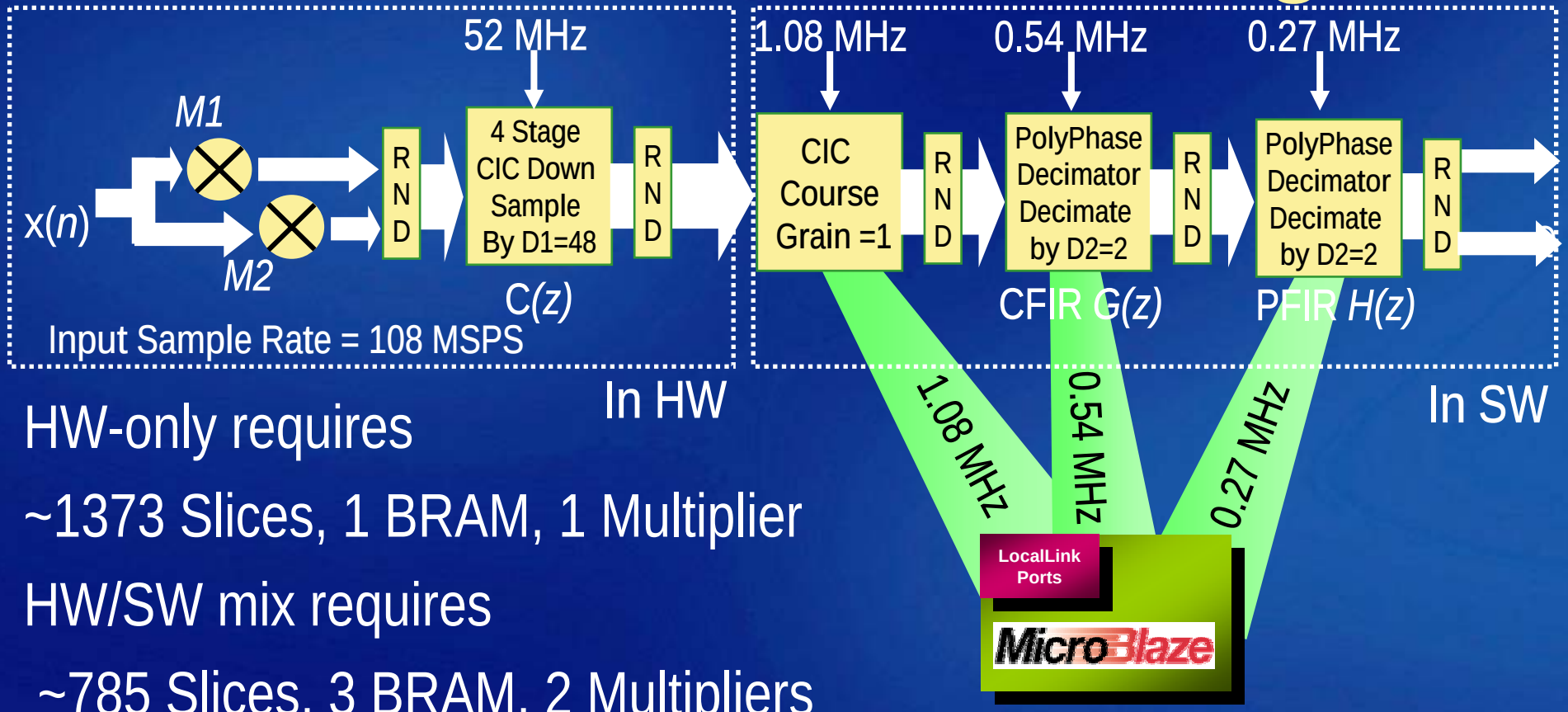
# Don’t Let High Speed Logic Drive Design Requirements

Digital Down Converter for GSM Base Station




- Low speed circuitry must be designed to high speed rules
- Half of the circuitry is 10x faster than required
- Wasted performance, power, and silicon cost

# Don't Run at 52 MHz When 0.27 MHz is Enough




HW-only requires

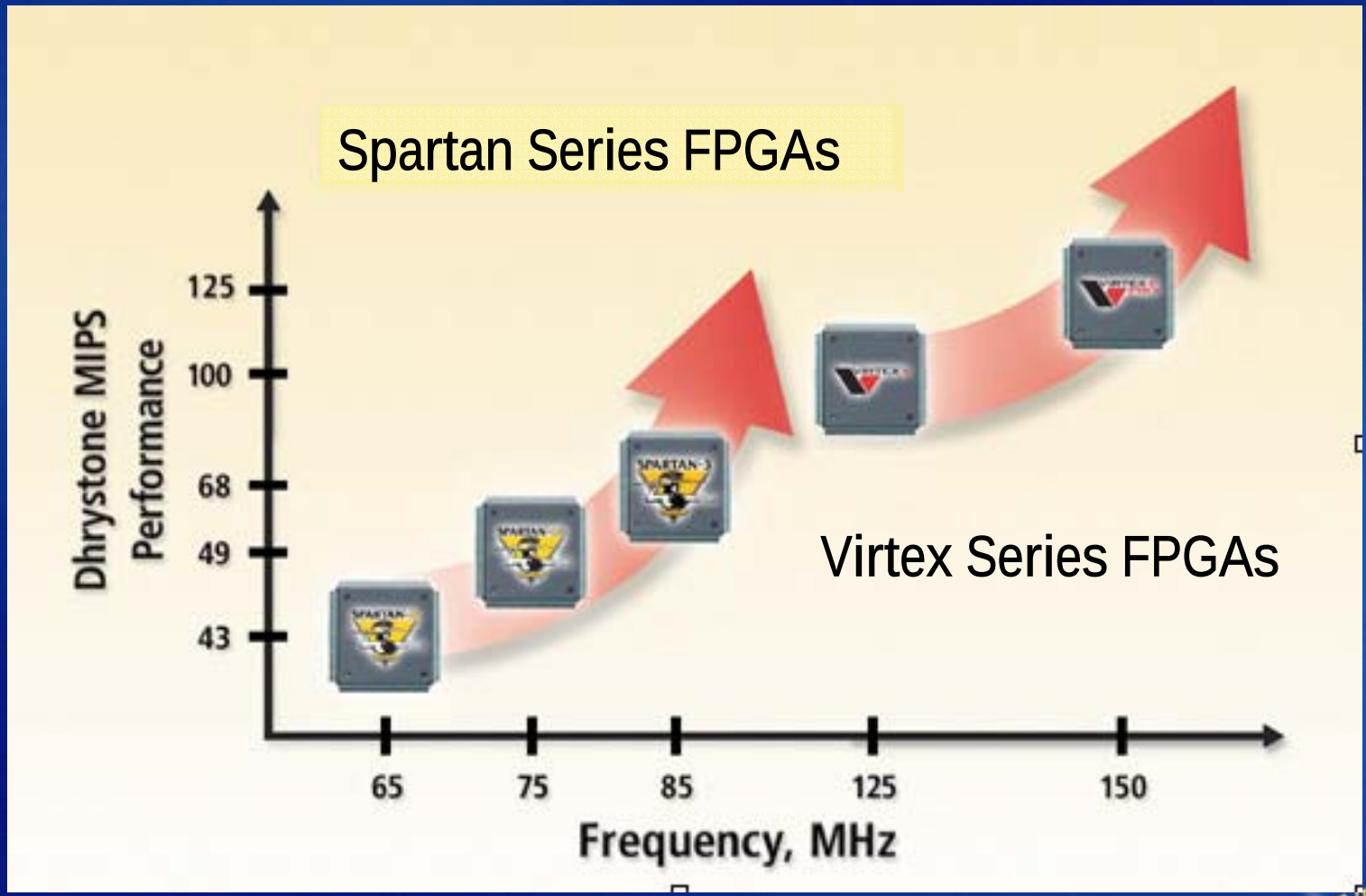
~1373 Slices, 1 BRAM, 1 Multiplier

HW/SW mix requires

~785 Slices, 3 BRAM, 2 Multipliers

***HW/SW mix is 34% smaller***

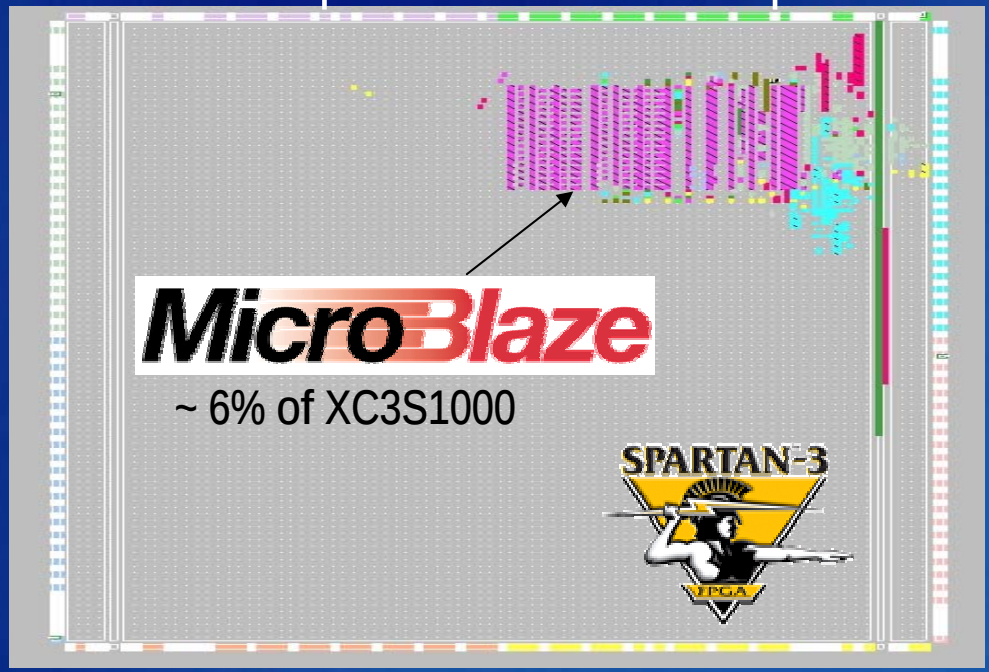
# MicroBlaze Device Support



# MicroBlaze Soft Processor

- 32-bit Harvard bus RISC architecture
- 32 general purpose registers
- 3 operand instruction format
- New features in EDK v3.2
  - Instruction and data caches
  - 32-bit barrel shifter
  - Hardware divider
  - LocalLink
  - MicroKernel
  - Hardware debug module
- Standard peripheral set
- GNU development tools

An Implementation Example




525 Slices in Spartan-3

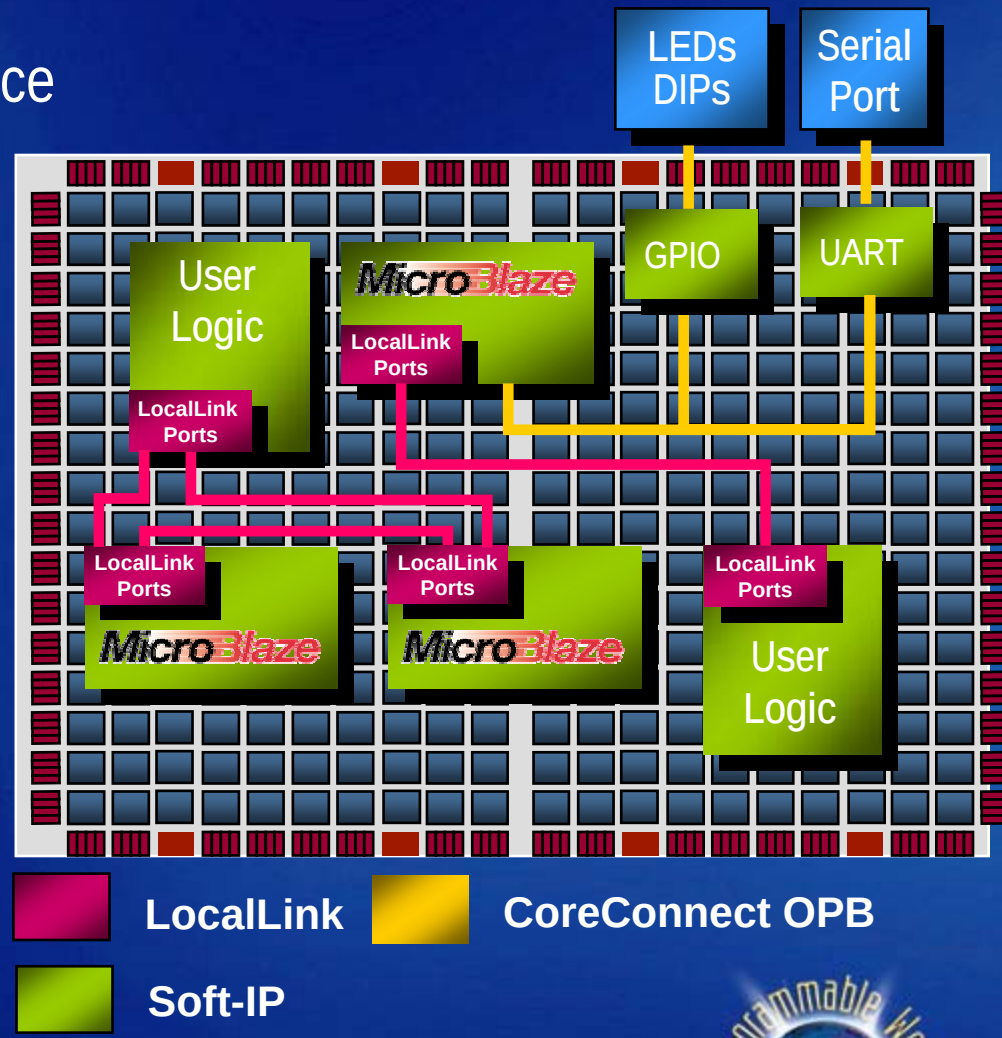
68 D-MIPS at 85 MHz

**Effective cost as low as $1.40***

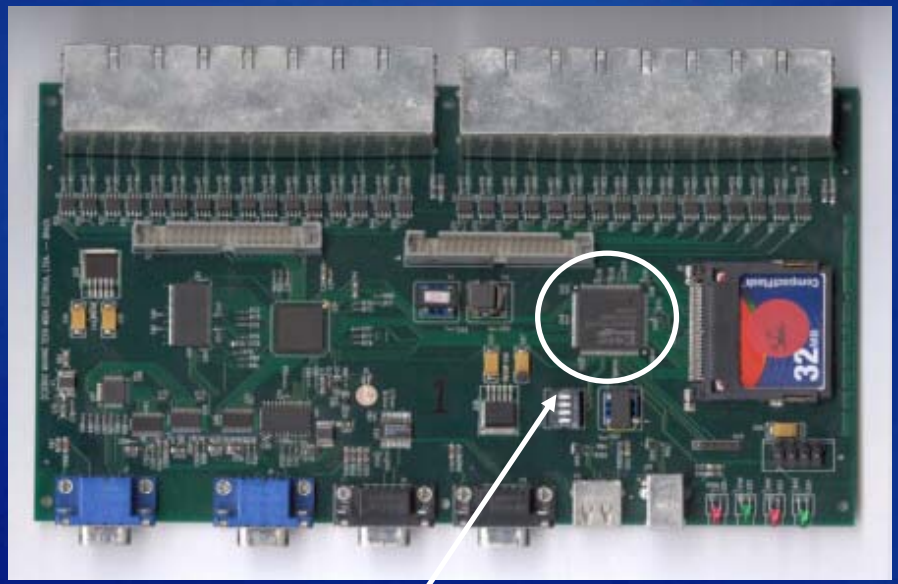
* Based on pricing for 2004, 250K units

# LocalLink with MicroBlaze

- 300MB/sec direct processor interface
- 2-cycle transfer
- Point-to-point connection
  - Inter-process communication
  - Custom functions & hardware
  - Co-processing functions
- Up to 32 input/output LocalLinks
- Configurable depth FIFOs on input/output LocalLink
- Use CoreConnect™ OPB bus, LocalLinks, or both



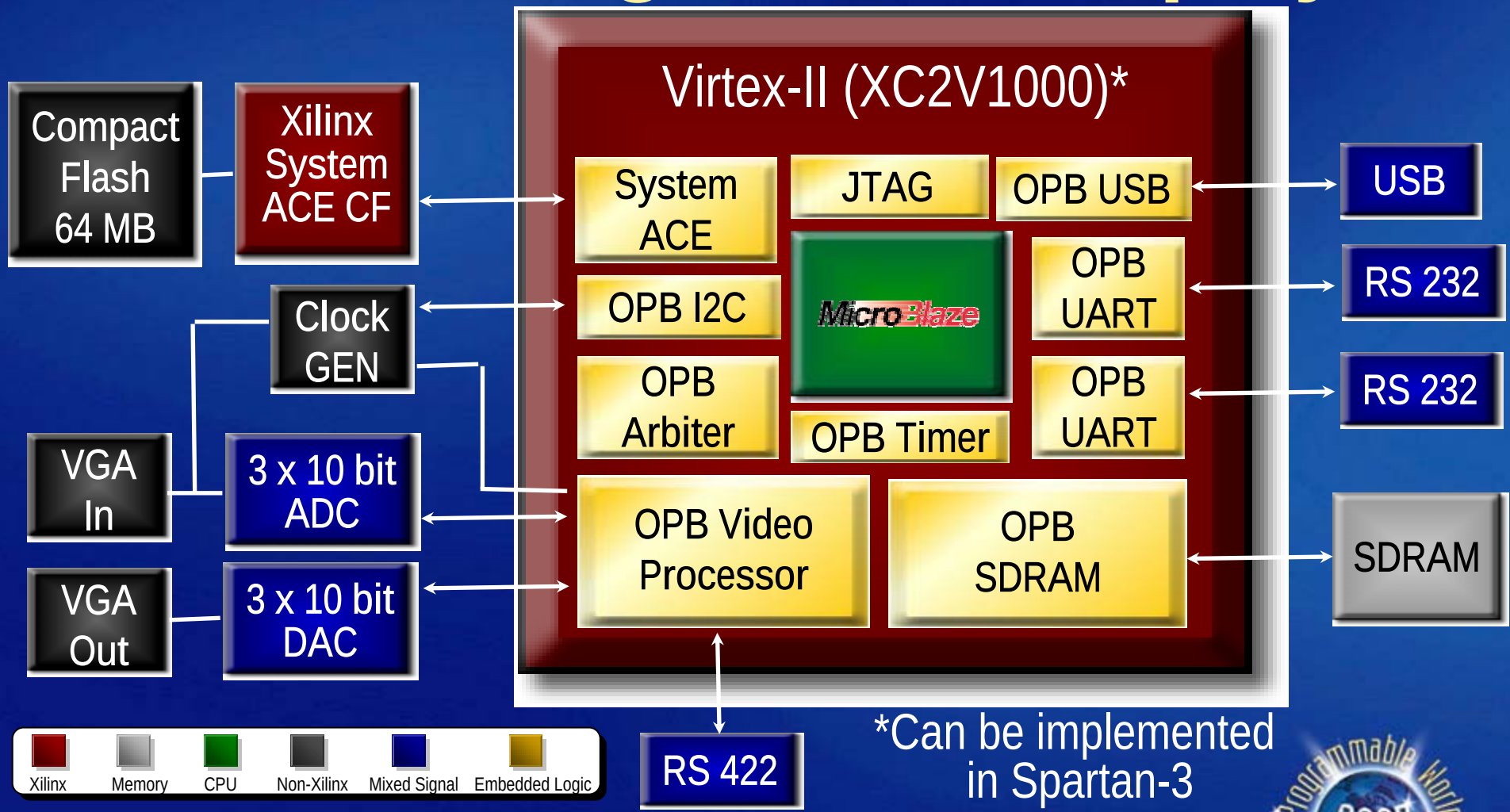
# Example: Birger Engineering 68 Billion Color LED Display



Xilinx FPGA featuring MicroBlaze for the video processor module


Large scale LED Display (4 x 5 meter)

# MicroBlaze Enables Video Processing in LED Display




*Can be implemented in Spartan-3

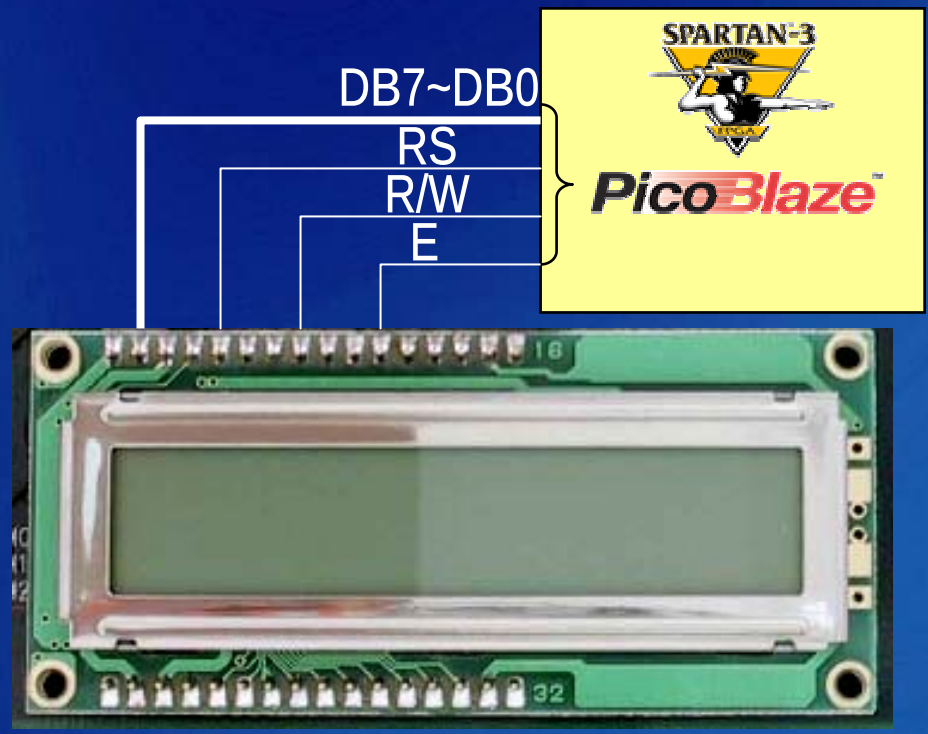
XILINX®

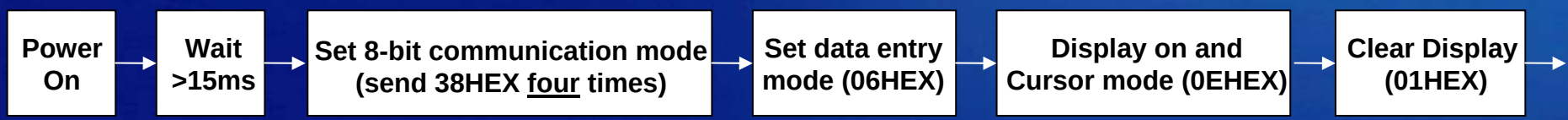
# When Speed Is Bad...

- Many peripheral components have been designed to be controlled by a processor
- Some peripherals require deliberately slow interfaces
- Implementing complex state machines in HDL with artificial timing constraints is challenging and inefficient

# Natural Processor Interface

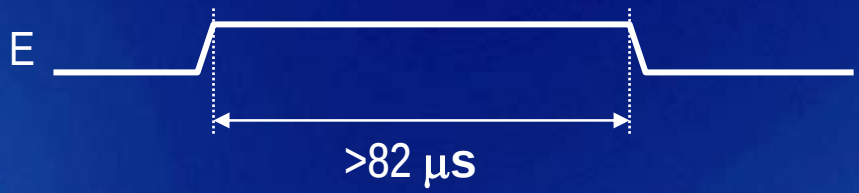
- Many peripherals are meant to be driven by a processor
  - Bus interface
  - Sequential initialization
  - Micro-second timing…



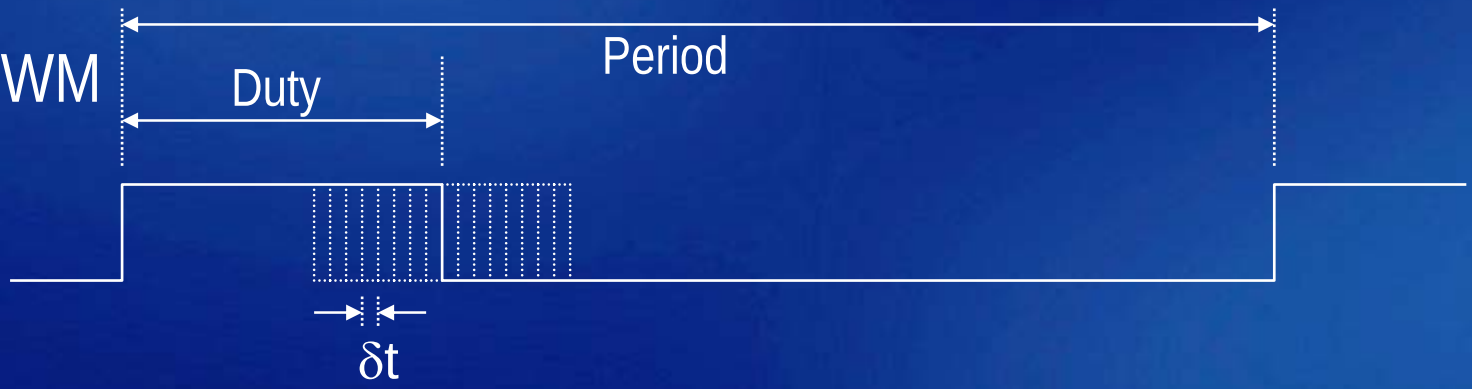



# Deliberately Slow Operations

LCD Module Timing




Clear Display and memory - wait 82μs to 1.64ms
Write data to character memory - wait 40μs



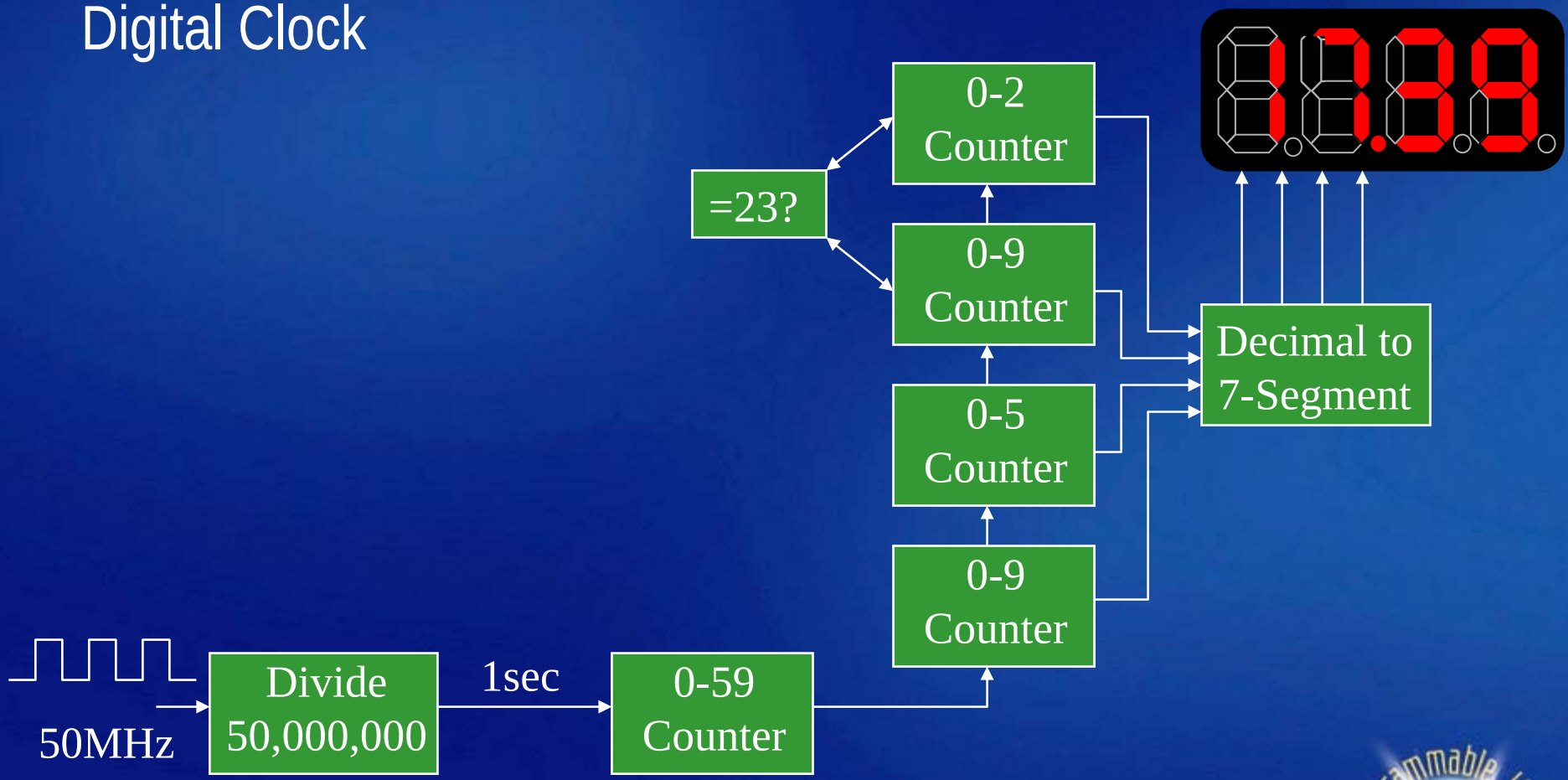

1% Duty Resolution
50 MHz Hardware Clock

| Rate | δt | Clock Cycles/ δt | Total Clocks |
|---|---|---|---|
| 150Hz | 66μs | 3330 | 333,000 |
| 1KHz | 10μs | 500 | 50,000 |
| 10KHz | 1μs | 50 | 5000 |

XILINX®

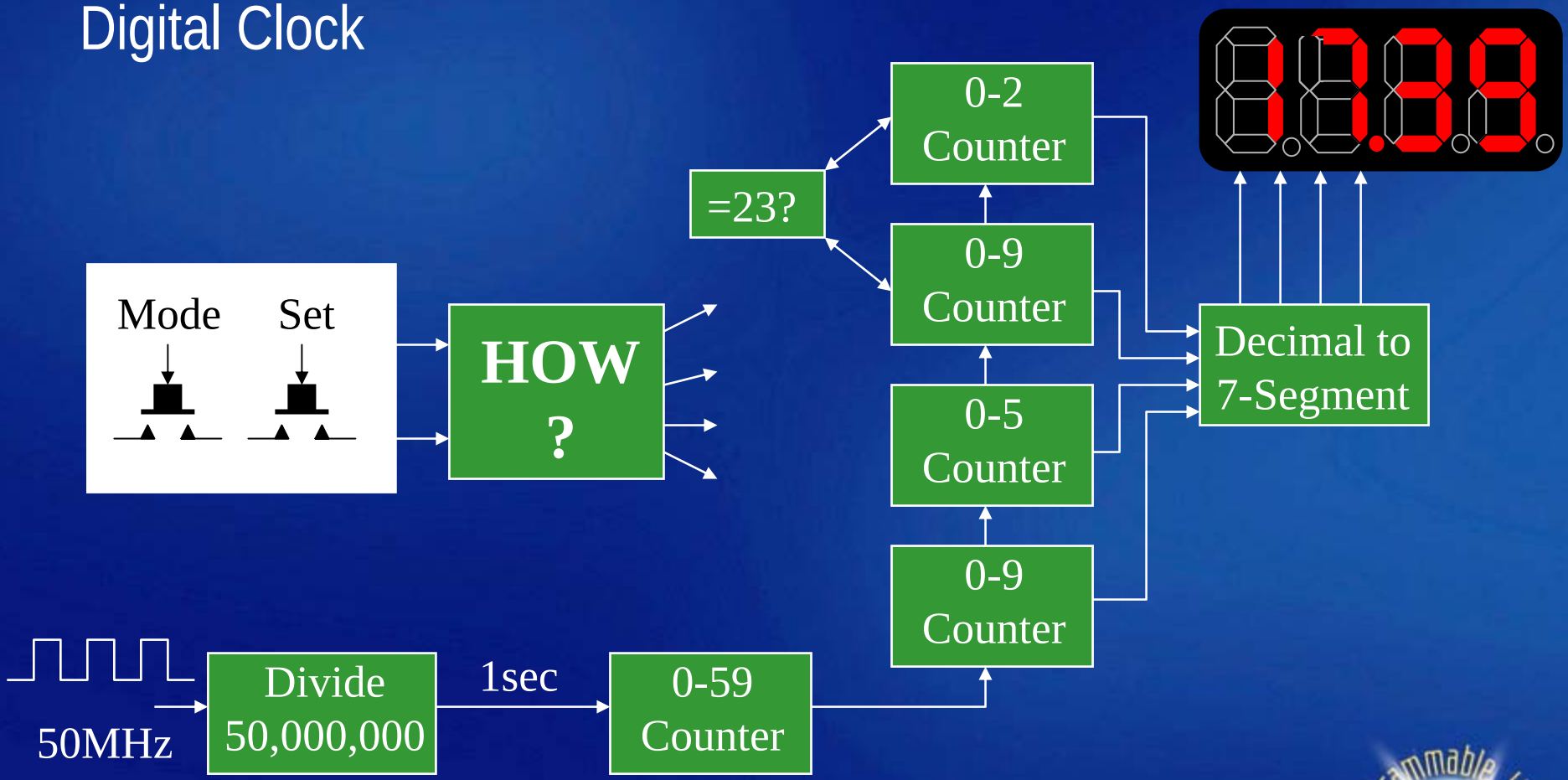
# Complex State Machines

Digital Clock



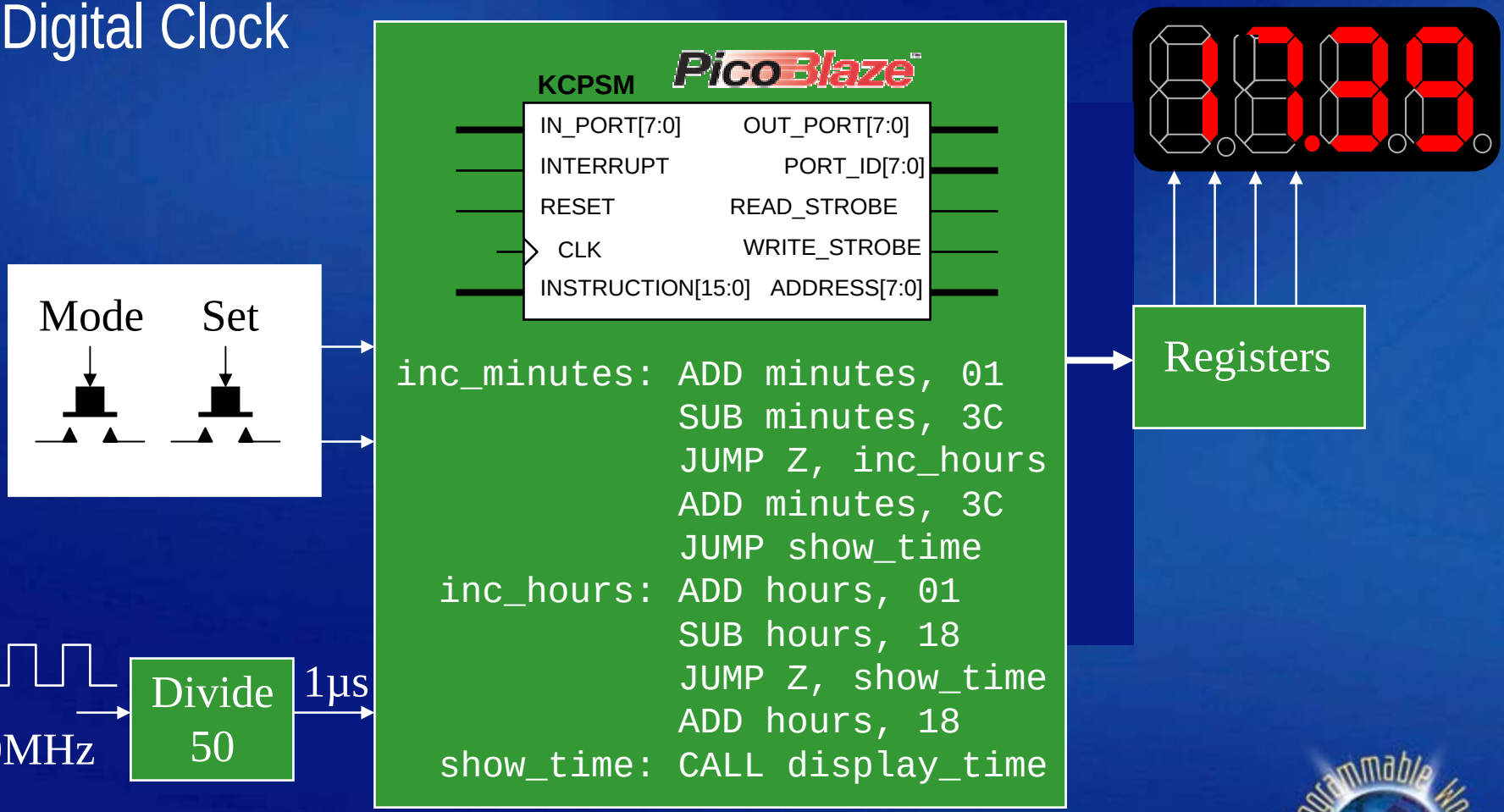
# Complex State Machines

## Digital Clock



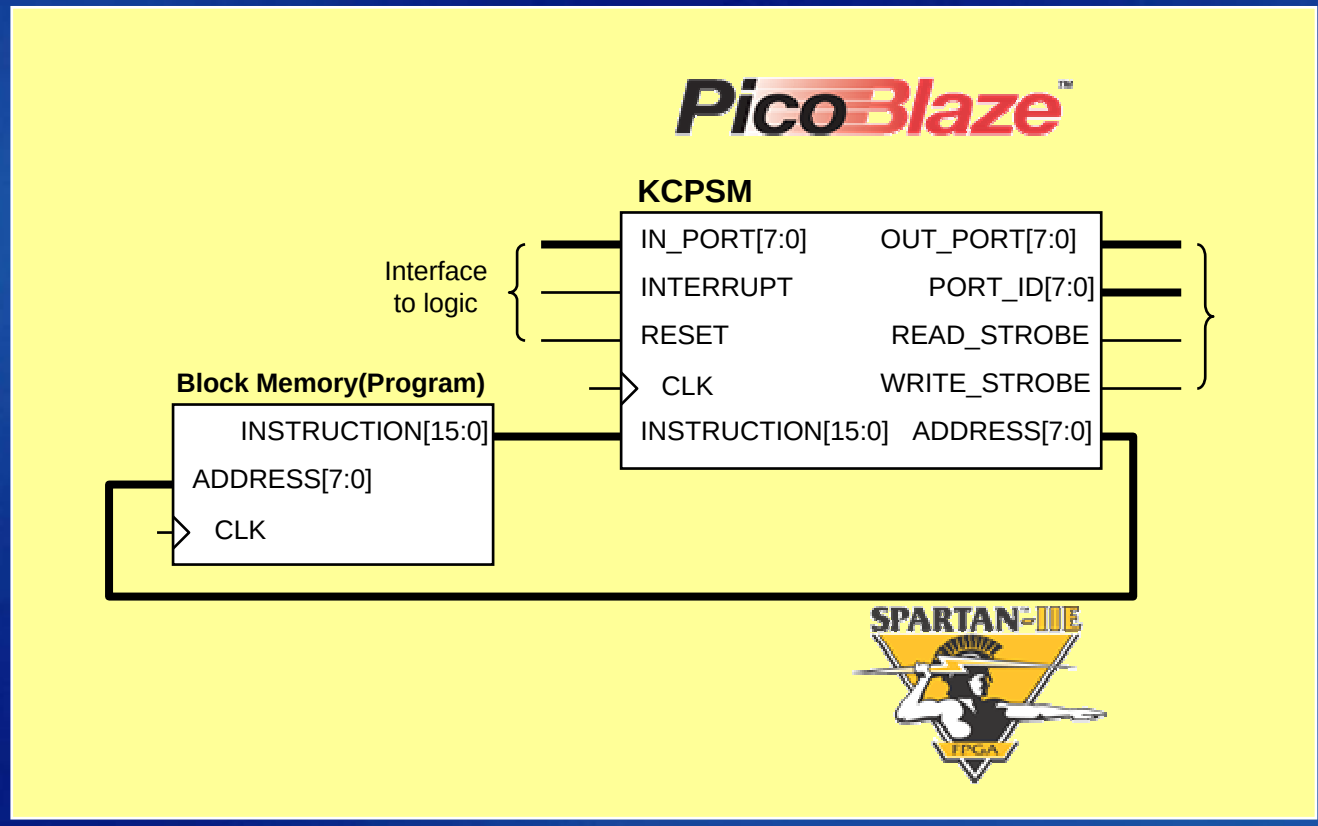
# Complex State Machines

## Digital Clock



# PicoBlaze Features

- 8-bit Data
- 16 Registers
- Interrupt
- Reset
- Built-In CALL & RETURN Stack
- Free Reference Design
- Source VHDL
- Program Stored in Block RAM



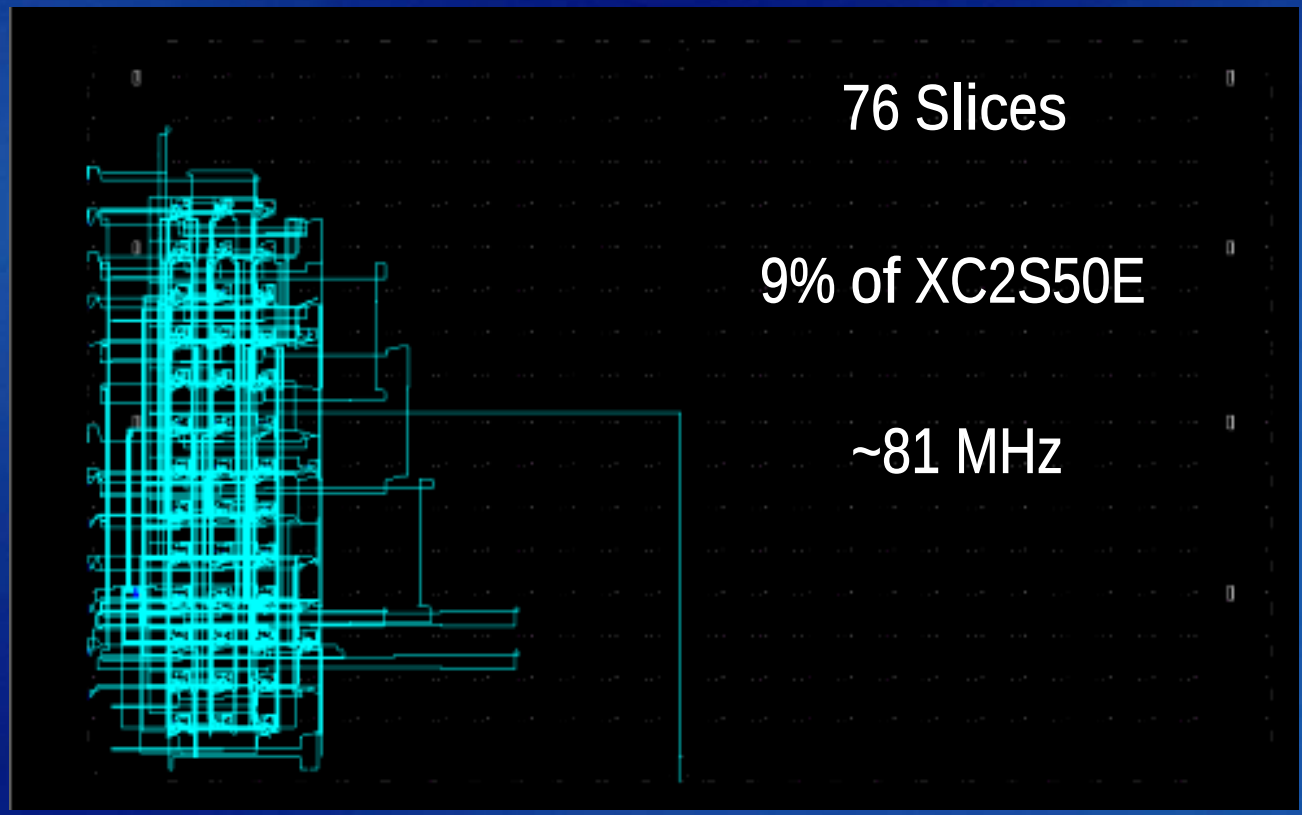

Spartan-II/E = 256 Instruction Code Space

Spartan-3 = 1,024 Instruction Code Space

XILINX®

# PicoBlaze Features

- 8-bit Data
- 16 Registers
- Interrupt
- Reset
- Built-In CALL & RETURN Stack
- Free Reference Design
- Source VHDL
- Program Stored in Block RAM




Spartan-II/E = 256 Instruction Code Space

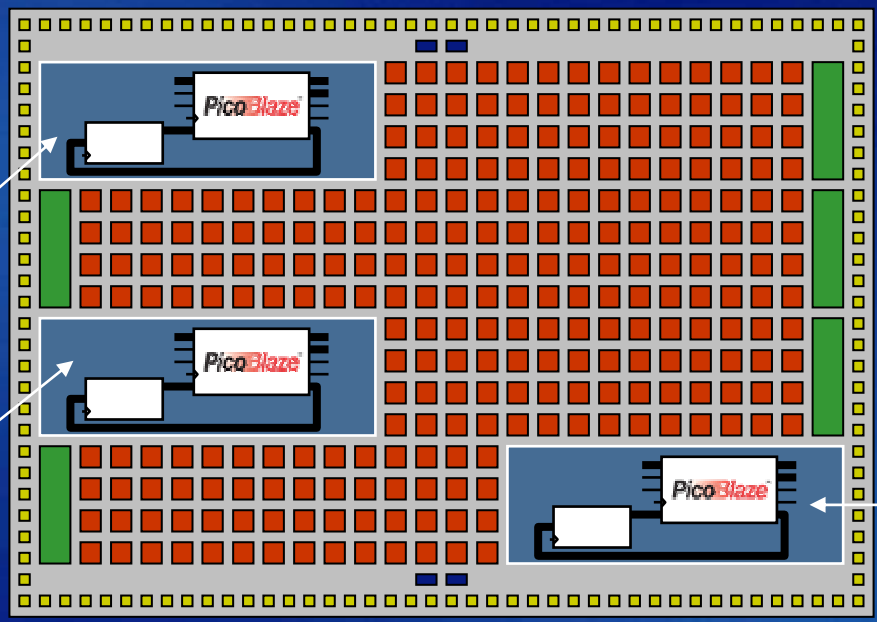
Spartan-3 = 1,024 Instruction Code Space

XILINX®

# PicoBlaze for Simple Processing

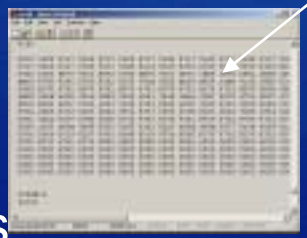
Key pad reader with DTMF dialing and tone generation

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
| * | 0 | # |




XC2S50E supports up to 8 PicoBlaze cores


UART communication providing remote diagnostic commands



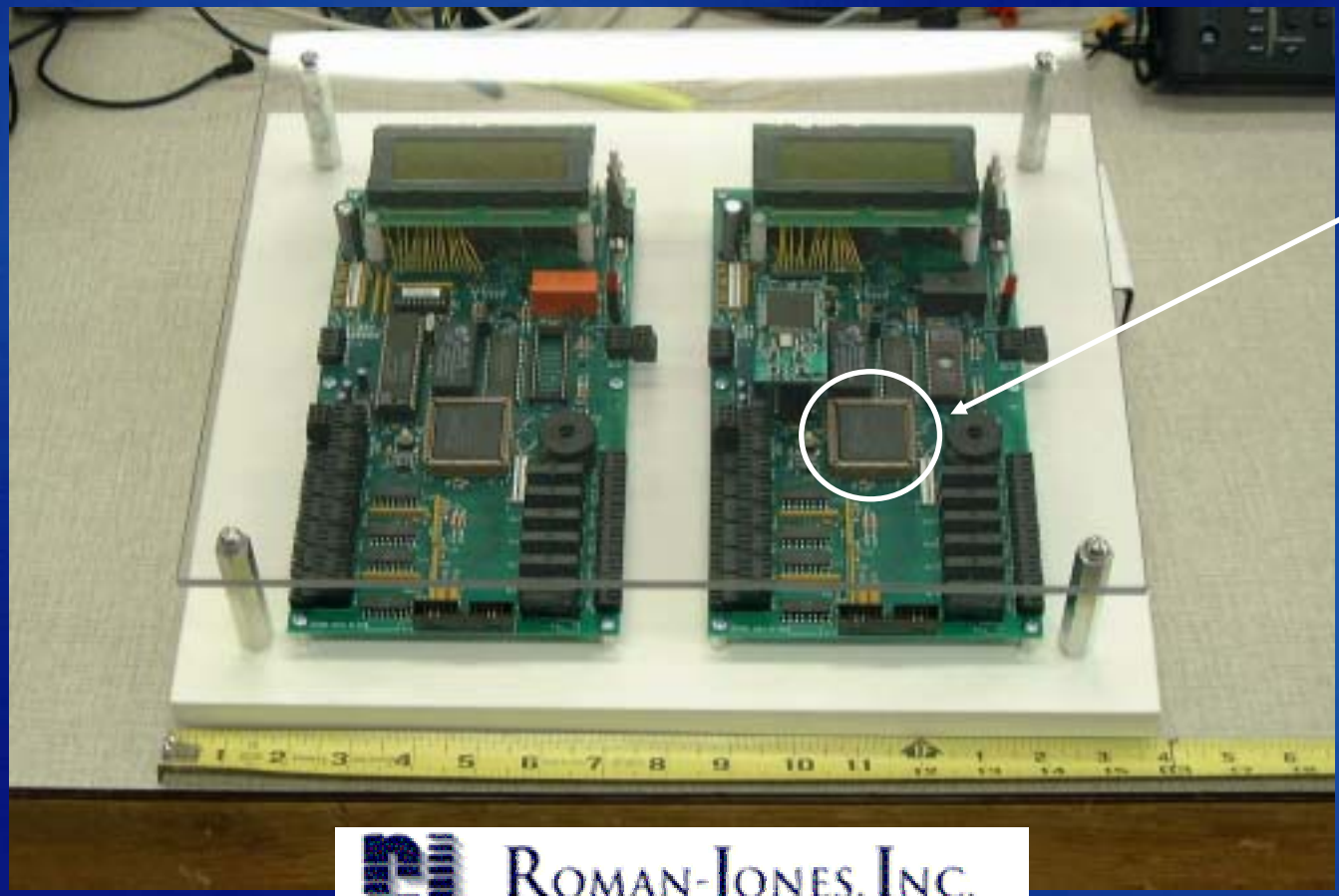
Display Control

"We found that having some intelligence in the hardware so close to the parts that actually do all of the work is remarkably efficient."

*Steve Brett - Technical Director - Pandora International Ltd*


XILINX®

# Emulated Embedded 8051 on PicoBlaze in Spartan-II



Spartan-II FPGA (XC2S30) featuring PicoBlaze to emulate 8051 operation

ROMAN-JONES, INC.

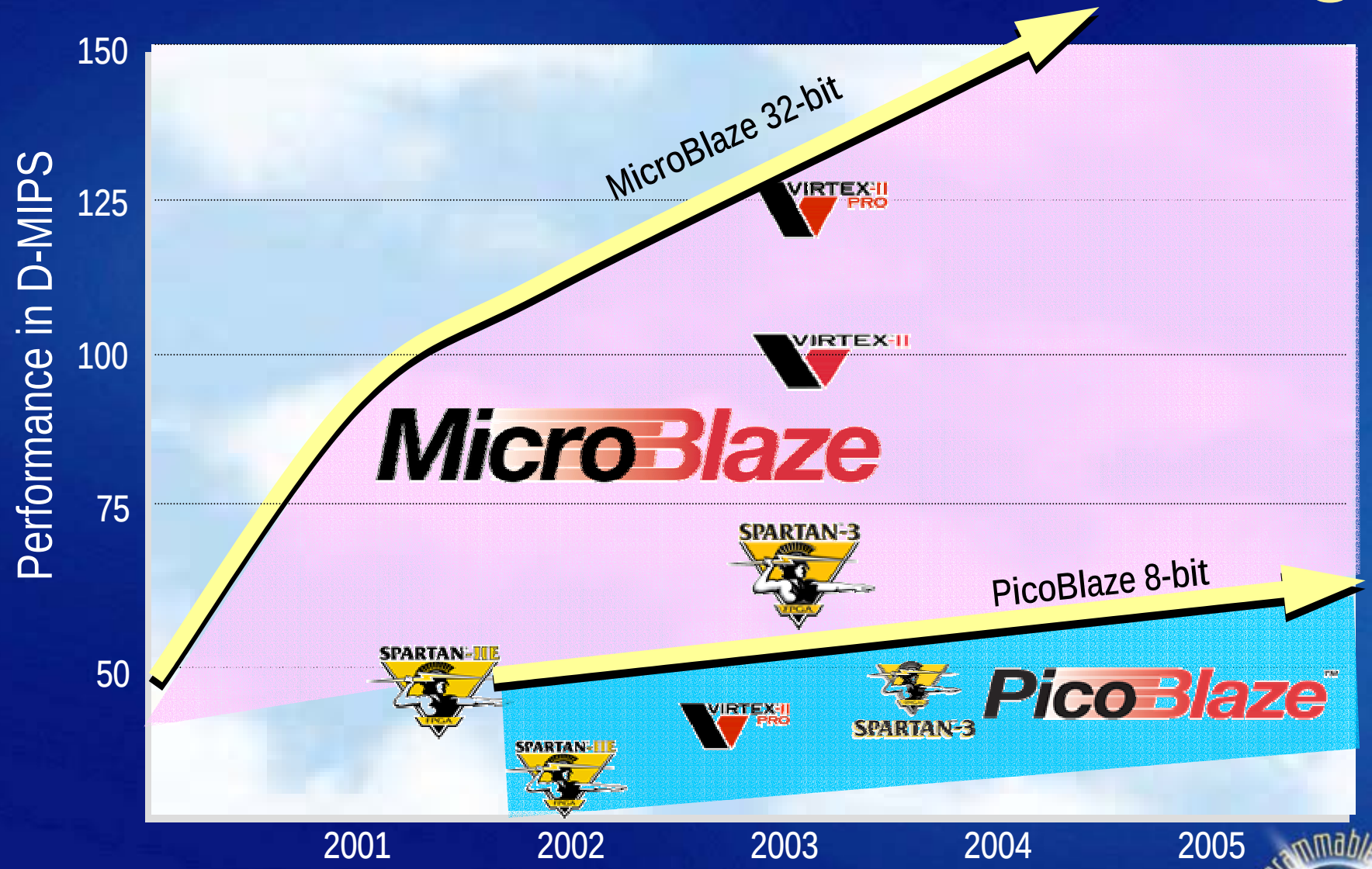
Consulting & Contract Engineering in the Electronic and Computer Disciplines

XILINX®



Programmable World 2003

# Xilinx Soft Processor Offerings



# The Designer's Solution

- Solve every problem the most cost-effective way
  - Some functions belong in HW
  - Some functions belong in SW
  - And some functions are just meant to interface to processors

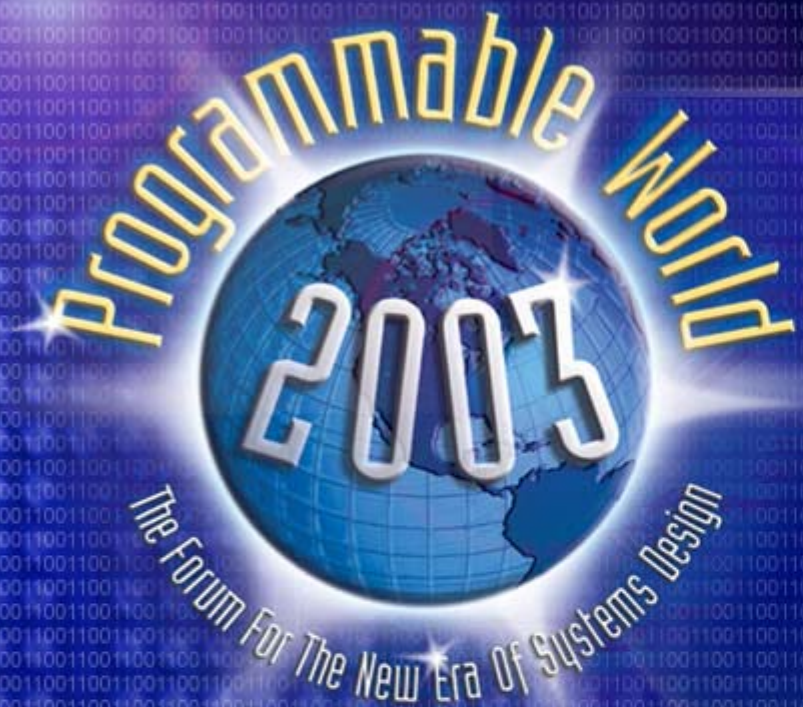
***Xilinx FPGAs With Soft Processor Cores Enable Flexible HW/SW Tradeoffs***


XILINX®
Programmable World
2003
The Forum For The New Era Of Systems Design
Thank You!
Celoxica
Synplicity
Texas Instruments
The MathWorks
Mentor Graphics
synopsys
NALLATECH
WIND RIVER
CMP
Electronics Group
XILINX®
IBM
Agilent Technologies
cadence
intel
Altium


# Reference Slides

# Need More Information?

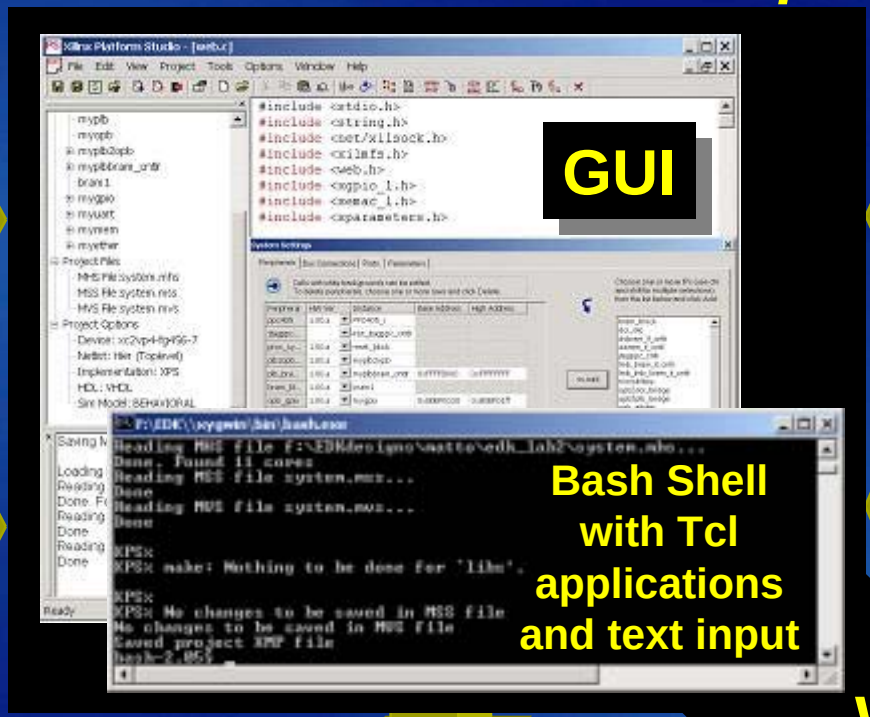
- Processor Central: www.xilinx.com/processor
- EDK: www.xilinx.com/edk
- MicroBlaze: www.xilinx.com/microblaze
- PicoBlaze: www.xilinx.com/picoblaze




# EDK 3.2 Software Tools Reference Slides

# Programmable Platform Design

## ISE Platform Studio IDE – *Your Desktop SoC Factory*




Your Code, Apps, OS…

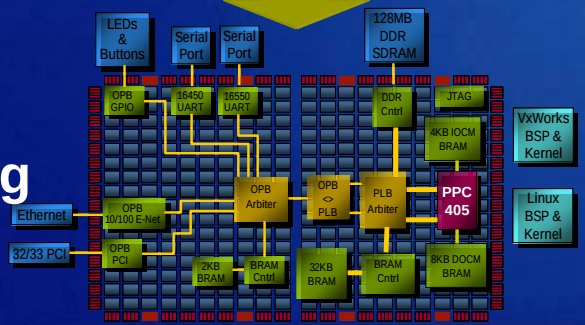
Optimized SW Libs and Drivers
libc.a
Xilinx Micro-Kernel

Your Custom HW

Parameterizable HW IP

Work the Way You Want – GUI, Command Lines or Both!

Your Custom Computing Platform



# Programmable Platform Design

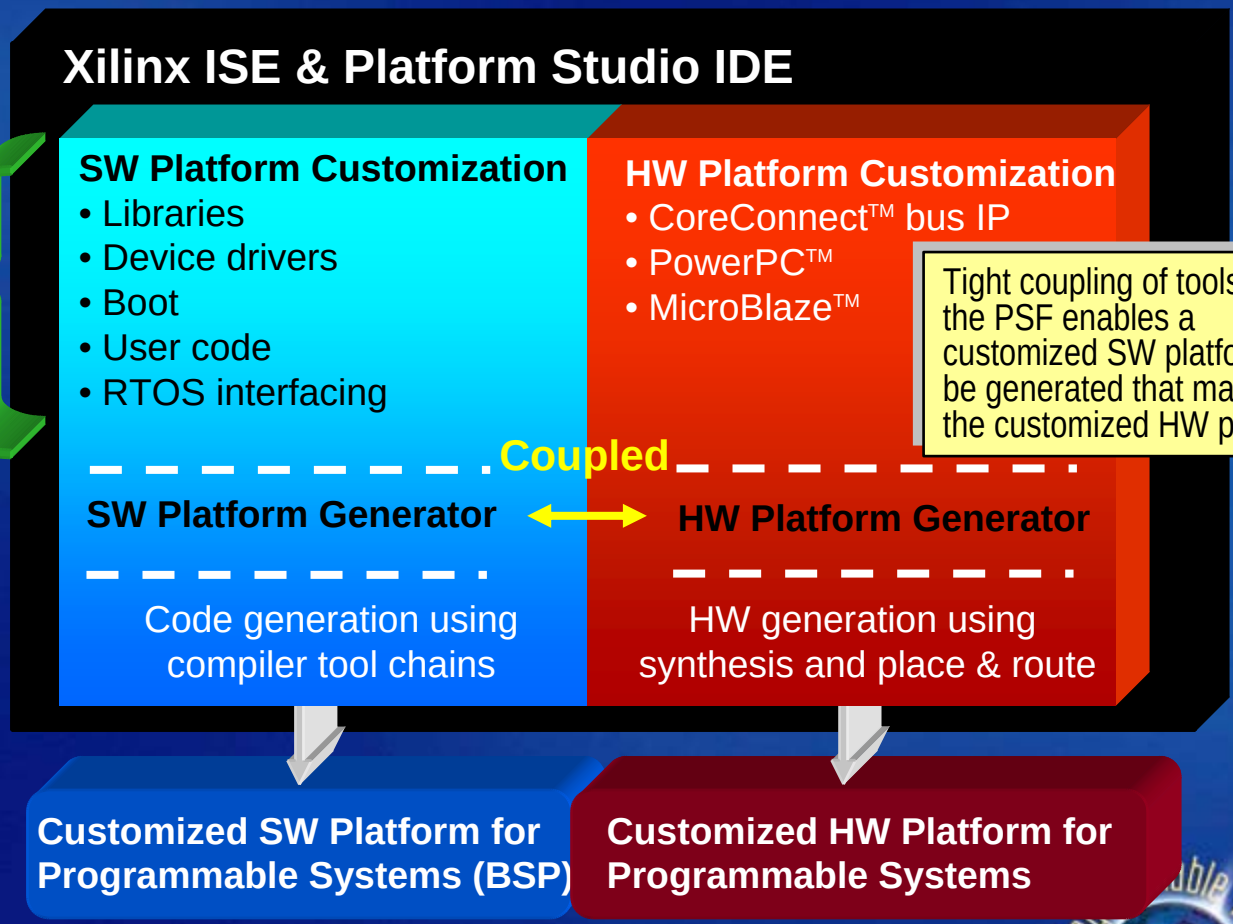
## Bridging the gap between the promise of single-language "co-design" and traditional "over-the-wall" HW/SW approaches

Enables traditional design teams (HW & SW) with a better SoC approach

Platform Specification Format (PSF)

An abstraction that enables the HW & SW domains to be coupled during the design process

- Does not require HW & SW to be "unified"
- Supports the traditional bottom-up approach that is familiar today



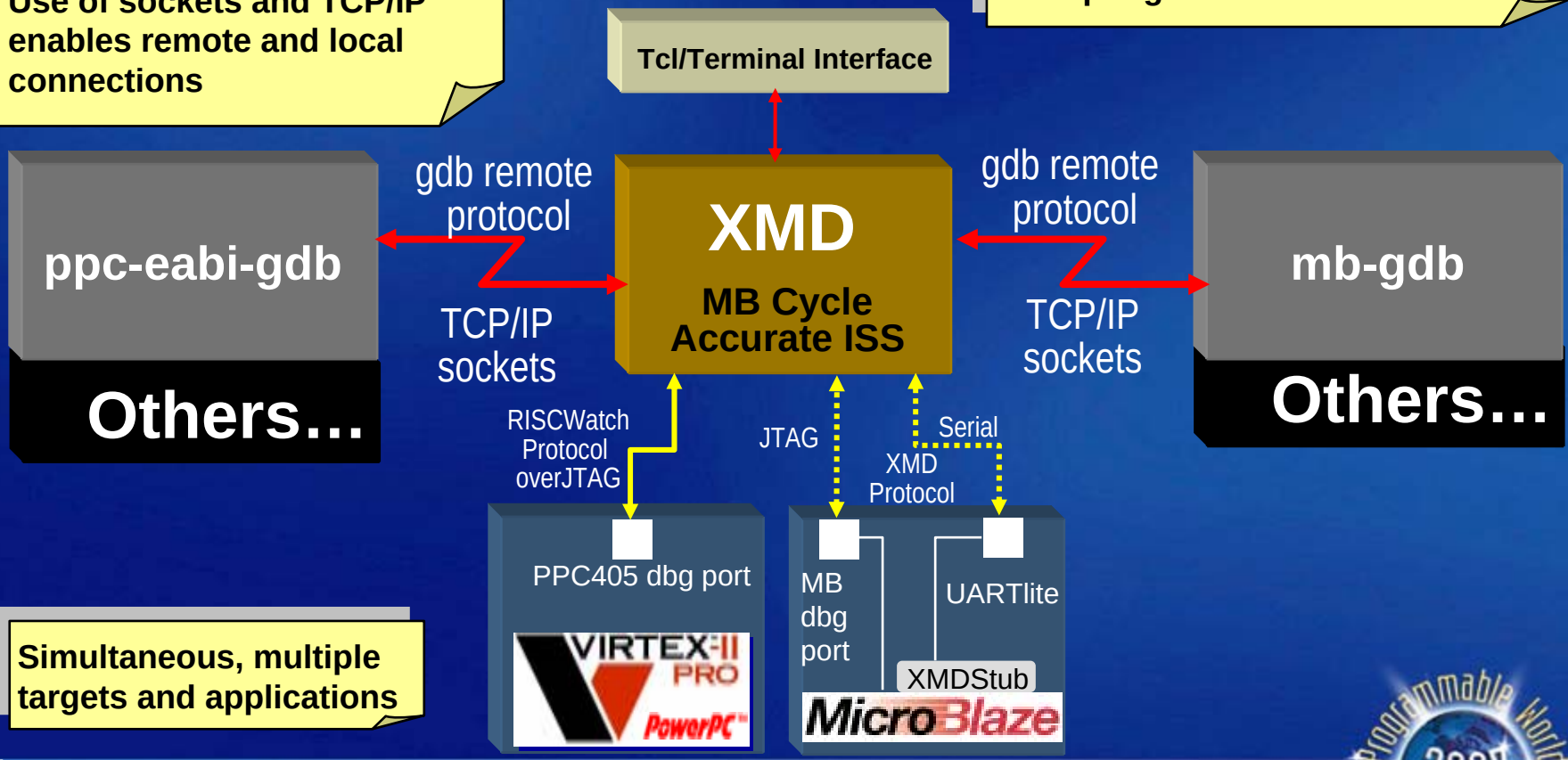

Provides the foundation for automated Architectural Synthesis

# Xilinx Microprocessor Debug (XMD)

**Plumbing and synchronization between host-side application and:**
- **Other host-side applications**
- **Actual targets**

**Use of sockets and TCP/IP enables remote and local connections**

**Tcl Interface enables:**
- **Command line control of debugging using Tcl capabilities**
- **Complex verification and analyses scripting**




**Simultaneous, multiple targets and applications**

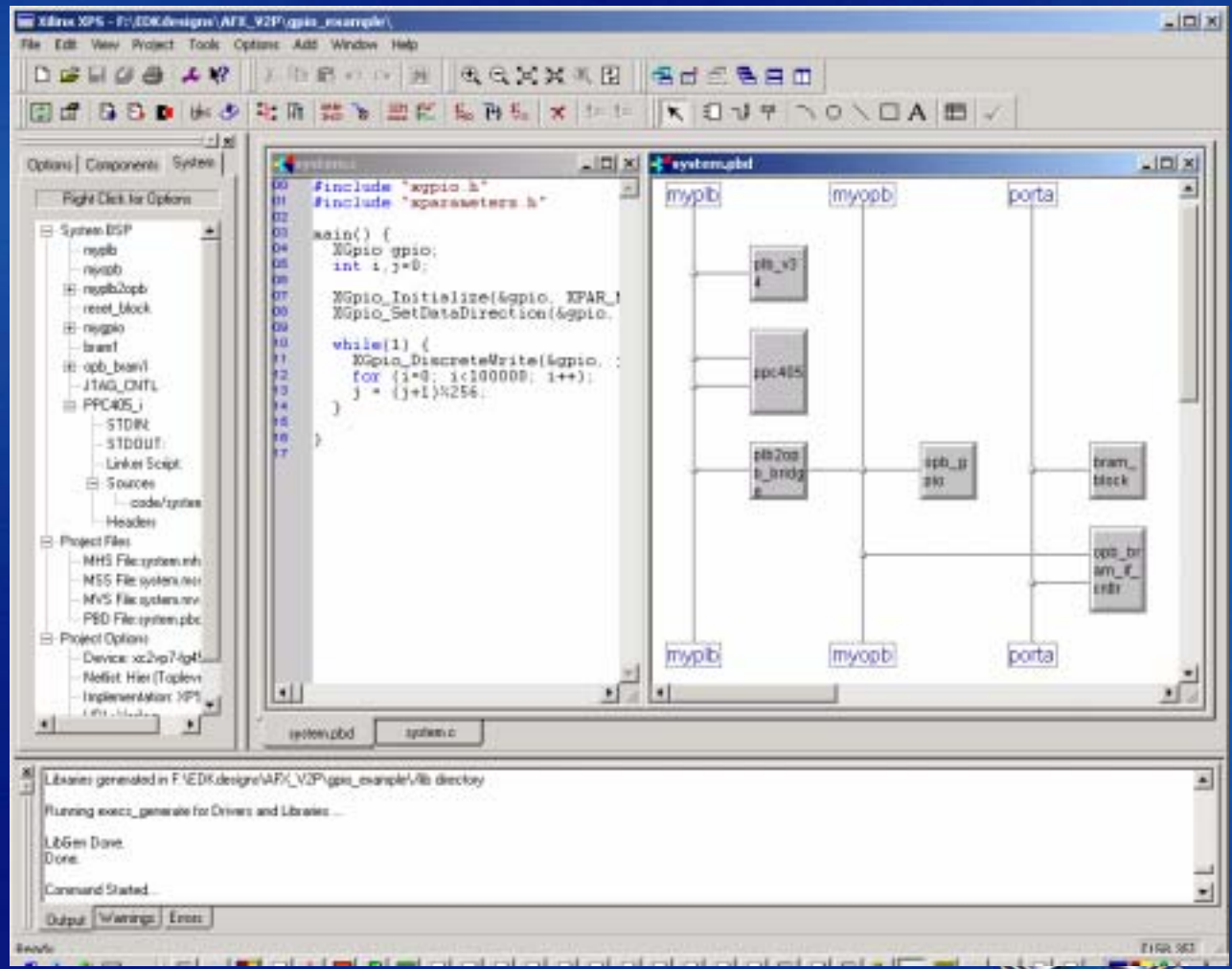
XILINX®

# EDK 3.2 Release

- Xilinx Platform Studio
  - Platform Block Diagram Editor
  - GUI on Solaris
  - Improved database facilities
  - User Toolbar
- Xilinx Microkernel libraries
  - Highly modular, OS-like services
- MicroBlaze
  - I-Cache and D-Cache
  - Hardware Divide Instruction
  - HW Debug module
  - LocalLink for Processors
- Requires ISE 5.2i, SP1

# Xilinx Platform Studio 3.2

- Platform Block Diagram Editor
  - Integrates System Generator Pro functionality
  - Provides editing as well as generated diagrams



# Xilinx Platform Studio 3.2

- Full GUI on Solaris
- New Platform Specification Format database facilities
  - Foundation for improved and more robust rules checking of design before generation
  - Expanded SW platform generation capabilities
- User Toolbar
  - Launches user-identified applications, scripts, etc.

# Xilinx Microkernel Libraries

- Goal: Provide highly modular, OS-like facilities for Xilinx embedded CPU systems
  - Utilize “the FPGA way” in HW design for SW design
    - Only build-in exactly what you need
    - Minimize resource usage and footprint size
  - Supports PPC405 and MicroBlaze
- Existed in EDK 3.1, as an unsupported capability – in 3.2, this is now supported
  - User changes made to source are not supported
- Source included, open license model
  - No exposure of your IP, customization, etc. is required
  - Free to do with as you will

# New Debug Capabilities in 3.2

- New MicroBlaze HW debug module
  - Connects via JTAG just like PowerPC 405 core
  - Includes runtime control and/or trace and/or profiling data
  - No memory space for debug kernel needed
  - No debug kernel to crash/more robust debug
  - Runs with Xilinx Parallel IV cable and EDK-supplied SW debugger


XILINX®
Programmable World
2003
The Forum For The New Era Of Systems Design
EDK 3.2
Processor IP
Reference Slides
Celoxica
Synplicity
Texas Instruments
The MathWorks
Mentor Graphics
synopsys
NALLATECH
WIND RIVER
CMP
Electronics Group
XILINX
IBM
Agilent Technologies
cadence
intel
Altium

# Released Processor IP (EDK)

- Shipped as Clear text VHDL Source
  - OPB SDRAM Controller
  - OPB DDR Memory Controller
  - OPB EMC Memory Controller
  - OPB Block Memory Controller
  - OPB ZBT Memory Controller
  - PLB Block Memory Controller
  - PLB DDR Memory Controller
  - PLB SDRAM Memory Controller
  - OPB Uart-Lite, OPB JTAG Uart
  - OPB Timer / Counter, OPB Watchdog Timer
  - OPB GPIO, OPB SPI
  - OPB Interrupt Controller
- Additional Included IP
  - OPB System ACE controller
  - OPB IPIF Interface
  - PLB IPIF Interface
- Shipped as Evaluation (Includes Eval License)
  - OPB Uart-16450 & OPB Uart-16550
  - OPB HDLC
  - OPB IIC
  - OPB Ethernet 10-100 EMAC
  - OPB Ethernet-Lite 10-100 EMAC
  - OPB ATM Master Utopia Level 2
  - OPB ATM Slave Utopia Level 2
  - OPB PCI 32 Bridge
  - OPB ATM Master Utopia Level 3
  - OPB ATM Slave Utopia Level 3
  - PLB ATM Master Utopia Level 2
  - PLB ATM Slave Utopia Level 2
  - PLB GMAC Ethernet
  - PLB RapidIO (Installs in to EDK)

New

Open Source for Infrastructure Cores (EDK SP2)

# IP Evaluation In The EDK

## For LogiCORE IP that Xilinx Licenses ($$)

- Evaluation IP in the EDK (EDK SP1)
  - To Evaluate $$ IP, Customer Must Buy the EDK
  - OPB 10/100 Ethernet MAC, OPB2PCI 32/33 Bridge, IIC, UART 16450/550, ATM Utopia 2 & 3
- EDK 3.2 Install Includes Evaluation IP
  - Installed with 10 month Evaluation License
- Evaluation Core can be Treated in Every Way Like its Licensed Counterpart
  - Evaluation Core can be Processed Through MAP, PAR and Through Bitstream Generation
  - Logic Added to Disable Some Aspect of the LogiCORE's Functionality (Controlled by License File)
  - Evaluation Core will Function in Hardware for 6-8 Hours

# PCI 32/33 is not “Free”

- To Evaluate PCI32 OPB Bridge, You Buy the EDK 3.2
- To get Full License for PCI32, You a PCI32 LogiCORE Product
  - Full OPB2PCI32 Install and License is in the PCI32 Lounge

Same Process for ATM, Ethernet, RapidIO, Etc

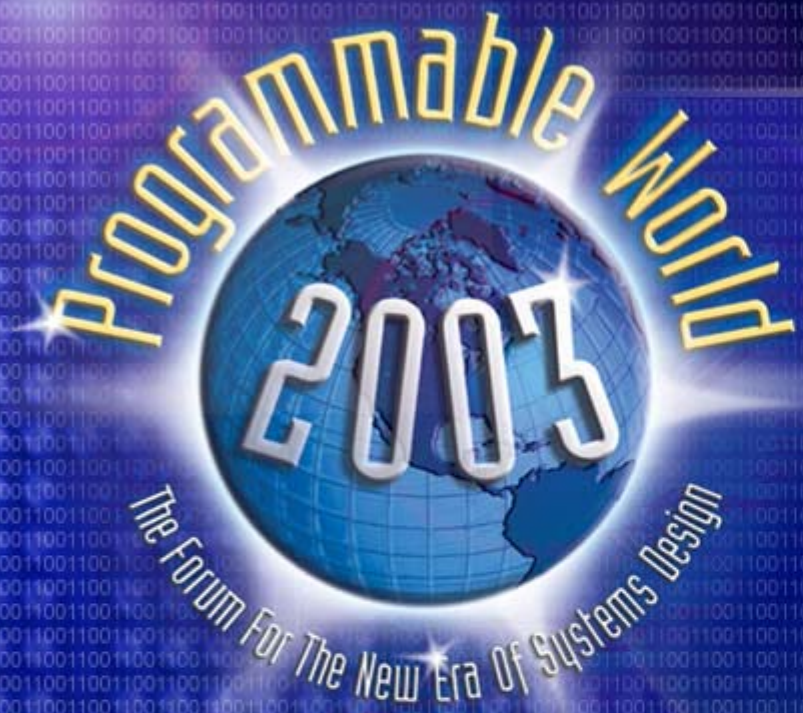
# Interfacing to FPGA Fabric

- Key to efficient user model is interface to FPGA fabric
  - FIFOs cross clock domains between MicroBlaze and other FPGA Logic
    - Input and Output FIFOs efficiently handle data transfer to/from MicroBlaze and other logic in the FPGA
      - Simplifies the users data flow control
      - Proven solution that matches user mindset
  - H/W support for simple “semaphore” operation
    - Simplifies “connecting” FPGA hardware operations with software task
    - Synchronize hardware to software processes.
  - MicroBlaze LocalLink Input and Output Ports for sampling and driving signals

# Processing Requirements

- DSP systems require varying levels of processing power
- HW runs fast
  - Great for higher rate processing, usually lower complexity algorithms
  - Developing complex algorithms in hardware is very challenging
  - Downside: Low sample rate processing H/W tends to be overkill and leads to significantly higher cost solutions.
- SW runs much slower
  - Suitable for low sample rate processes
  - Time sharing of resources leads to dramatic silicon area savings
  - Upside: handles significantly more complex algorithms easier

An ideal solution has support for BOTH HW and SW!




# LocalLink for Processors Reference Slides

# LocalLink vs BUS

- Unidirectional point to point communication
- Unshared non-arbitrated communication mechanism
- Supports any arbitrary connection topology
  - Star topology
  - Pipelined unidirectional flow network
  - Bi-directional flow
  - Ring topology etc.

# LocalLink & BUS: Complementary

- Support for any imaginable data flow requirements
- Match interconnect architecture to data flow requirements of user program

# LocalLink with MicroBlaze (1)

- Available today on MicroBlaze !
- New instructions
  - Get : Read from LocalLink into register
  - Put : Write register contents to LocalLink
- Blocking Get and Put instructions
  - Stall until the instructions succeed
- Non-blocking Get and Put instructions
  - No stalling of processor pipeline
  - Carry bit set if instruction succeeds

# LocalLink with MicroBlaze (2)

- Configurable depth input FIFO on input LocalLink
- Configurable depth output FIFO on output LocalLink
  - Get and Put MicroBlaze Instruction
  - get fromInputFSL M, toReg N
  - put toOutputFSL M, fromReg N
- 5 bit opcode describing get and put instructions
  - 32 input and 32 output LocalLink Registers
- Input and Output LocalLink width same as datapath width of MicroBlaze (32 bit)

# Software Programming Model

- Direct access to input and output LocalLink via inline assembly of opcodes C/C++ program
- Macros for LocalLink access within C code
  - XilReadFsl (value, fslN) - read from input LocalLink N into C program variable ‘value’.
  - XilWriteFsl (value, fslN) - write C program variable ‘value’ into output LocalLink N
  - fslN : integer value between 0 and 32
- Fully supported by Assembler
- Automatic inference and scheduling not yet supported by compiler

# LocalLink Performance (1)

- No latency beyond input FIFO delay assuming non-empty FIFO (data available)
- No latency beyond output FIFO delay assuming non-full FIFO (data ready)
- Get/Put instructions operate at Fmax of processor
  - Adding logic over LocalLink does not affect processor speed
- 2 cycles to execute get instruction
  - One cycle to test if data available on input FIFO
  - One cycle to transmit data from input FIFO to register

# LocalLink Performance (2)

- 2 cycles to execute put instruction
  - One cycle to test if output FIFO not full
  - One cycle to transmit data from register to output FIFO
- Data transfer rate over LocalLink on MicroBlaze: 300Mbyte/sec
  - assuming 150 MHz processor speed (VII Pro)
  - 32 bit data path
  - assuming no stalls due to empty or full FIFO
  - 300 Mbyte/sec = (32/8 byte)*(150M/sec)*(1/2 cycles)

# Streaming Data Performance: LocalLink vs BUS

- LocalLink
  - Data availability tests performed in hardware for LocalLink
- BUS
  - Requires busy loop for polled bus transfers
  - Requires interrupt handler for interrupt driven bus transfers
  - Larger overhead associated with data availability tests and arbitration for buses

# LocalLink : Direct Communication

- Communication between two MicroBlaze cores with data transfer rates of 300 Mbyte/sec
  - Point-to-point communication with no arbitration overhead
- Communication between hardware processing engines implemented on the FPGA
  - Data transfer rate = 600 Mbyte/sec
  - assuming 150 MHz clock, and 32 bit data
- LocalLink not associated with processor can clock at a much higher rate ~600 MHz
  - Data transfer rate = 2.4 Gbyte/sec
  - Multiple clock domains not supported yet

# Hardware System Implementation

- Number of input and output LocalLink on configurable on MicroBlaze
- MHS based interconnect specification
- PlatGen based interconnect generation
- Can be used in conjunction with buses (LMB/OPB/PLB)