System-on-Programmable-Chip Design (SOPC)

John Tramel CSULB CECS460 Lecture 2

July 25, 2016

ref: Using System-on-a-Programmable-Chip Technology to Design Embedded Systems
J.O. Hamblen/T.S. Hall IJCA, Vol. 13, No. 3, Sept. 2006

System-On-a-Programmable-Chip (SOPC)

* Traditional SOC designs require the development of a custom IC or Application Specific Integrated Circuit (ASIC)
* Over the past years the cost of developing an ASIC have dramatically increased - easily runs > $1M
* Players now typically include a few high volume products which can support extended ASIC development time and high costs
* The end result is fewer ASIC “starts” are occurring
* An alternate approach, SOPC, now offers cost-effective solutions for developments that can not justify the long times/high costs

SOPC Approach - Advantages

* SOPC utilize large FPGAs to contain logic designs, memory arrays, and processor cores
* SOPC provides a means to rapidly implement a computer with custom hardware for use in embedded systems
* SOPC designs are reconfigurable with a much shorter development cycle
* SOPC provide a path to fast time to market
* SOPC designs are easier to revise and upgrade than SOC

SOPC Approach Disadvantages

* SOPC by definition will have lower processor performance than that achievable in an SOC design
* The recurring costs of an FPGA-based SOPC design will be greater than that of an SOC design
* The power demands will also be greater in an SOPC design than a SOC design

SOPC Compared to ASIC/ Processor Design

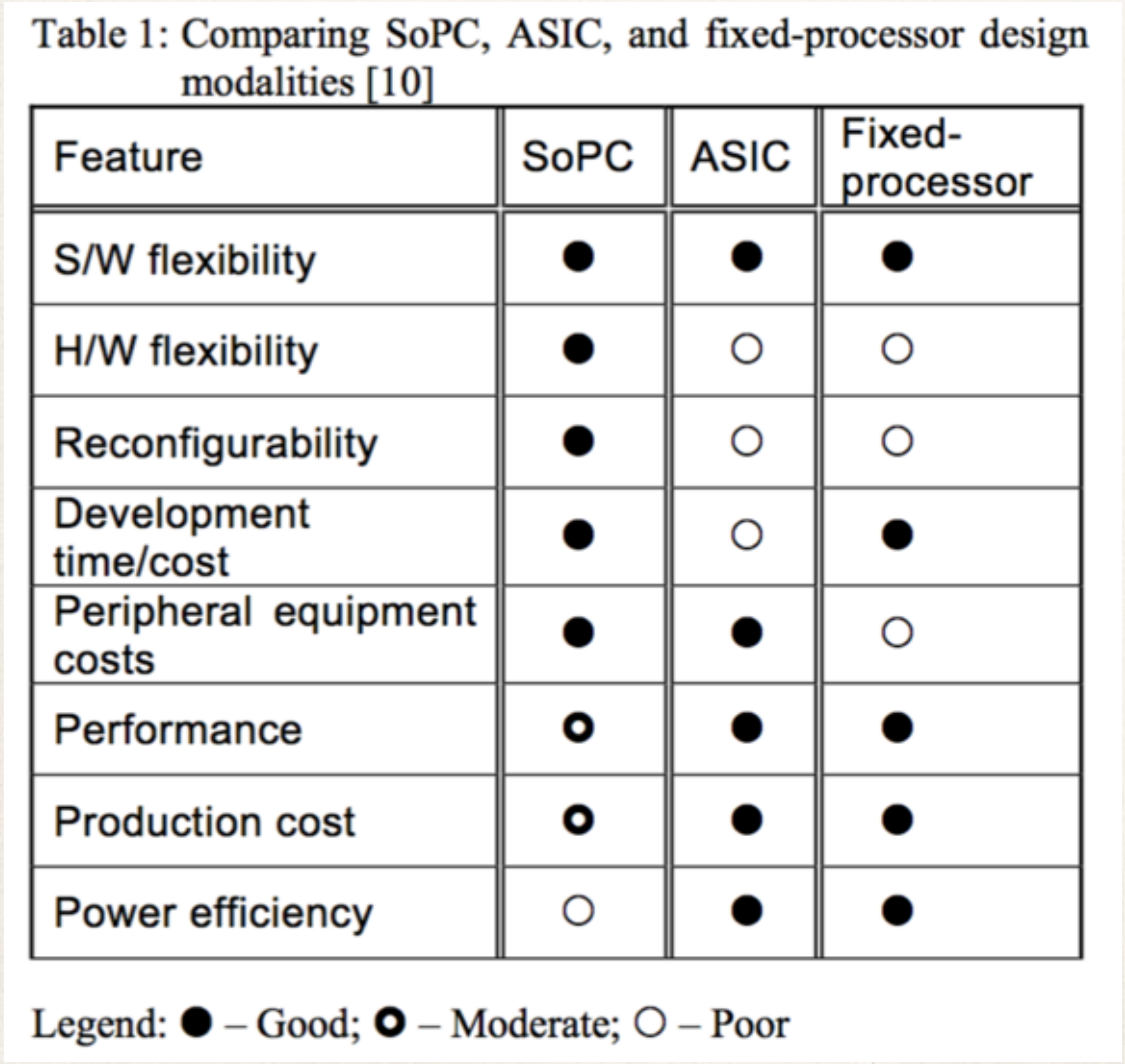


Table 1: Comparing SoPC, ASIC, and fixed-processor design modalities [10]

Feature	SoPC	ASIC	Fixed-processor
S/W flexibility	●	●	●
H/W flexibility	●	○	○
Reconfigurability	●	○	○
Development time/cost	●	○	●
Peripheral equipment costs	●	●	○
Performance	◉	●	●
Production cost	◉	●	●
Power efficiency	○	●	●

Legend: ● – Good; ◉ – Moderate; ○ – Poor

- If you compared SOPC to SOC the relative comparisons would hold
- The basic difference is that in the first approach the fixed-processor is on the PCB and in the SOC it is within the ASIC, now referred to as SOC

Hardware/Software Design Options

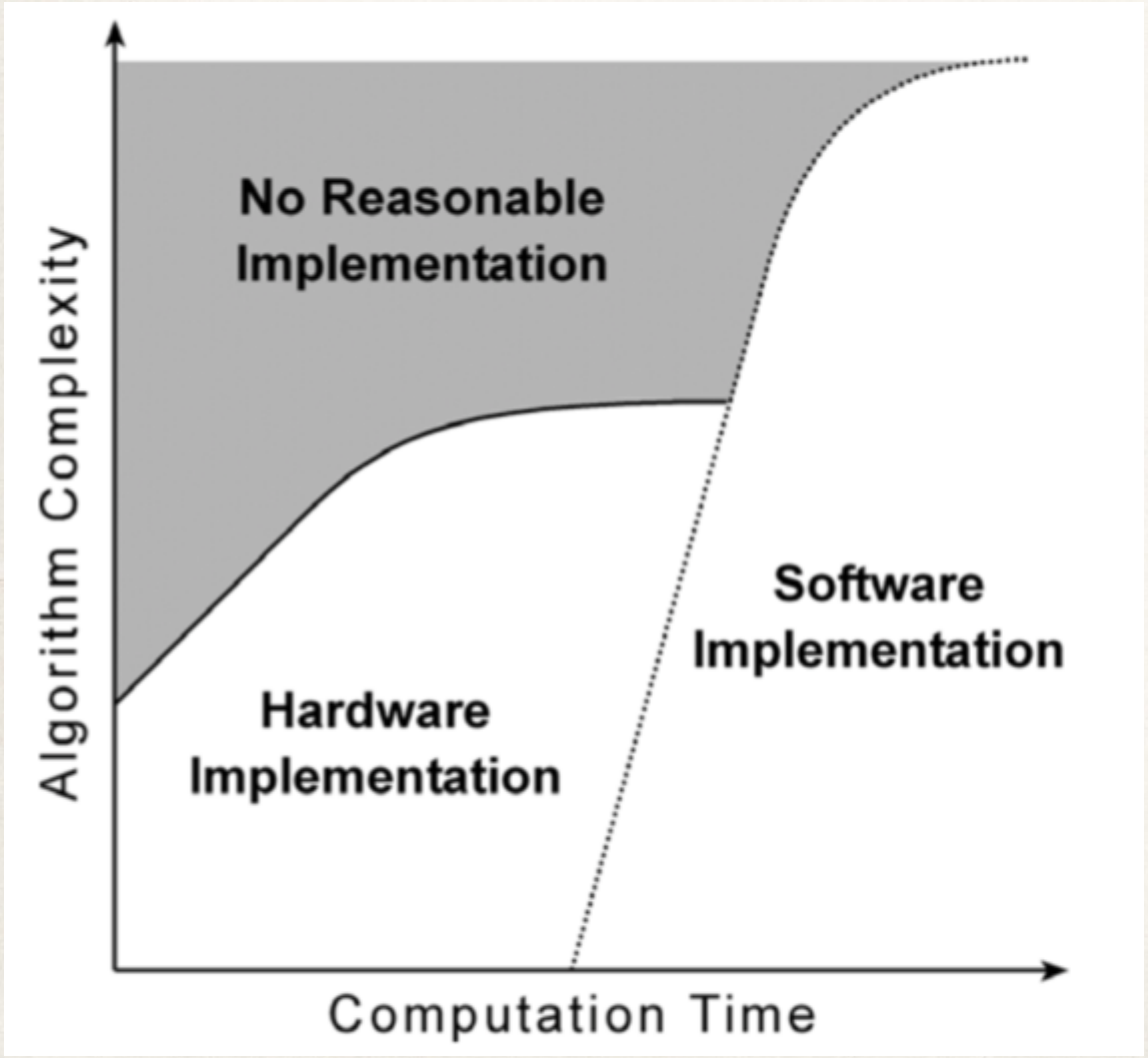
* SOPC provides a means to explore the issues that arise when considering which algorithms should be implemented in software or in the surrounding custom logic design

* Software approaches provide the capability of reconfiguring the design easily but at a cost of slower performance when compared to the custom logic

* Hardware performance issues also arise requiring alternate approaches including pipelining and parallel processing

* SOPC provides a platform for 'at speed' comparisons of different architectural approaches not achievable in SOC designs

SOPC Hardware/Software Tradeoffs

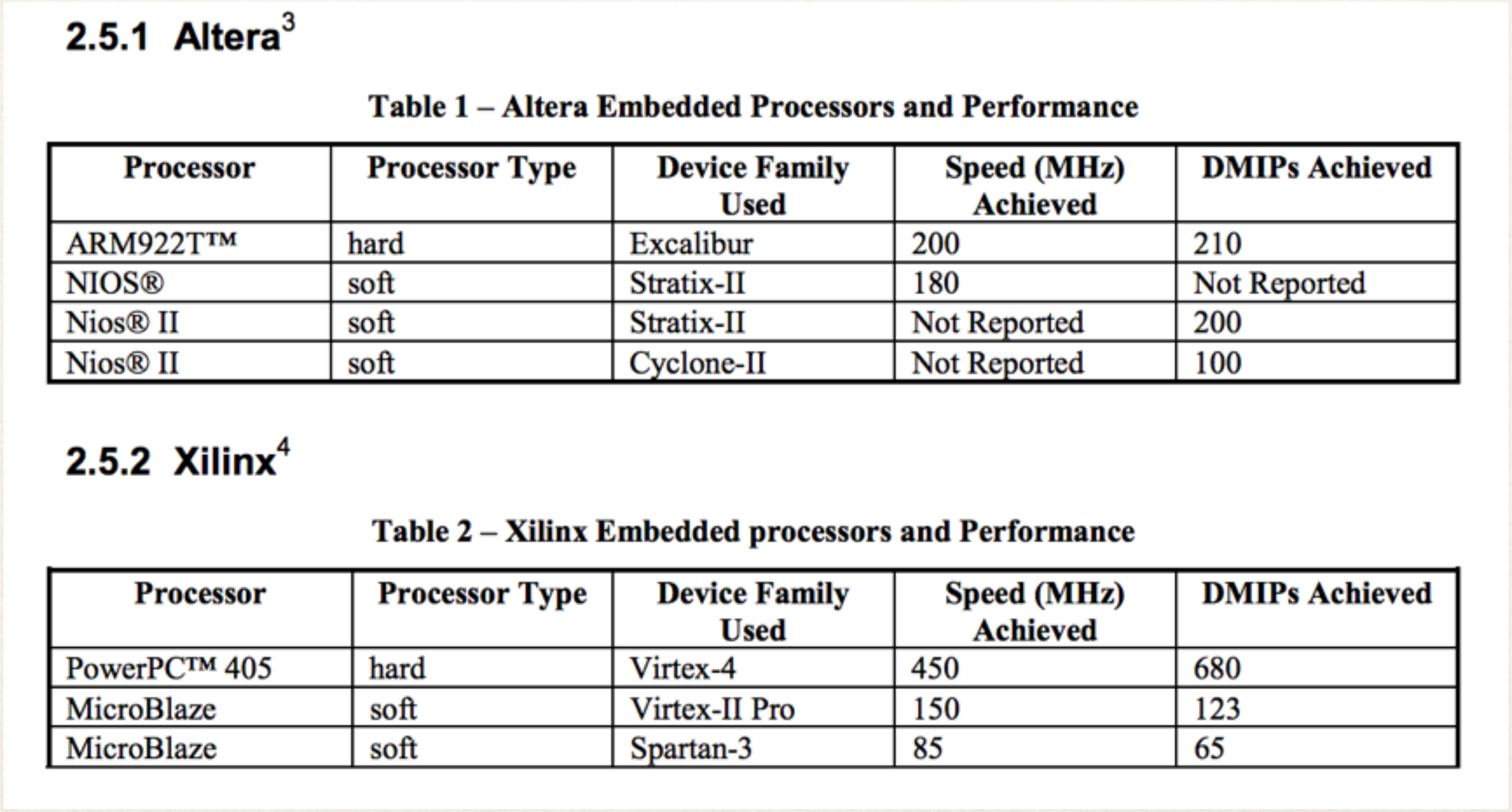


- Generalizations
  - Hardware implementations will run quickly but heavily tax modifications
  - Software implementations will run slower but will heavily tax performance
- The architect is responsible for driving these tradeoffs to the most optimal solution

SOPC Processor Cores

- SOPC designs require an FPGA with a processor core
- The processor core may be either “hard” or “soft”
  - Hard cores are built in to the fabric of the FPGA and are not easy to configure - the benefit of a hard core is that it should have a better performance than a soft core
  - Soft cores are RTL designs that share the available resources within the FPGA - soft cores typically are slower and more power-hungry than hard cores
  - Soft cores offer the the design team the capability to modify the processor design to customize it to their needs (within reason)

SOPC Processor Cores - Examples



2.5.1 Altera[3]

Table 1 – Altera Embedded Processors and Performance

Processor	Processor Type	Device Family Used	Speed (MHz) Achieved	DMIPs Achieved
ARM922T™	hard	Excalibur	200	210
NIOS®	soft	Stratix-II	180	Not Reported
Nios® II	soft	Stratix-II	Not Reported	200
Nios® II	soft	Cyclone-II	Not Reported	100

2.5.2 Xilinx[4]

Table 2 – Xilinx Embedded processors and Performance

Processor	Processor Type	Device Family Used	Speed (MHz) Achieved	DMIPs Achieved
PowerPC™ 405	hard	Virtex-4	450	680
MicroBlaze	soft	Virtex-II Pro	150	123
MicroBlaze	soft	Spartan-3	85	65

460: SOPC Processor Core History

- 460 has historically utilize the Xilinx PicoBlaze processor
- 8-bit microcontroller supporting following features:
    - 16-byte wide general purpose data registers
    - 1K instruction memory
    - Byte-wide ALU with CARRY and ZERO flags
    - 64-byte internal scratchpad RAM
    - 256 input and 256 output ports
    - 31-location CALL/RETURN stack
    - Single interrupt capability

460: SOPC Processor Core - Diagram

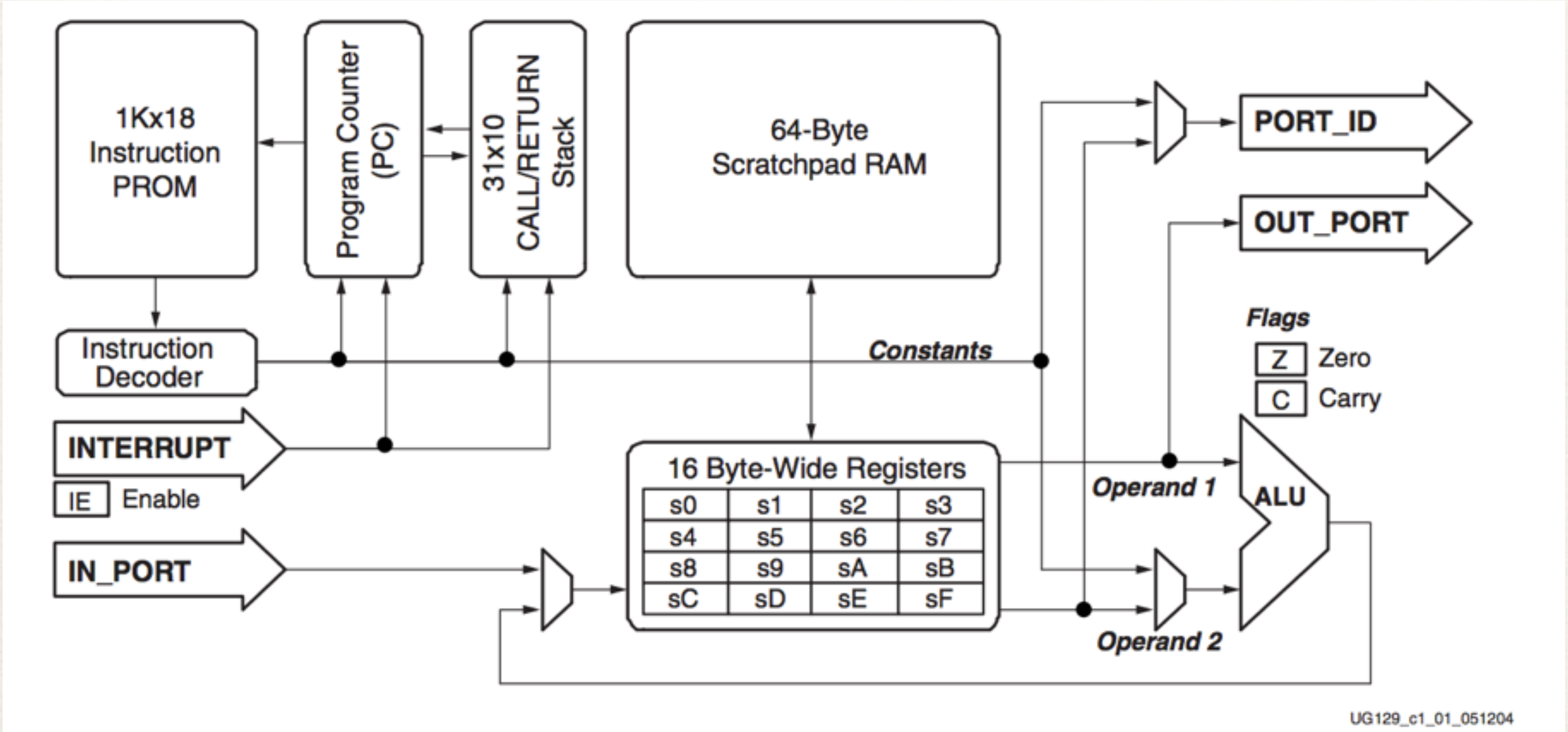


Figure 1-1: **PicoBlaze Embedded Microcontroller Block Diagram**

PicoBlaze Limitations

- 1K instruction memory limits amount of software that may be developed
- Targeted Spartan 3/Spartan 6 devices but has not continued to follow into the Artix devices
- Software development tools were limited and required the utilization of a DOS shell to assemble the executable code
- Design not easily modified to add capabilities to the processor

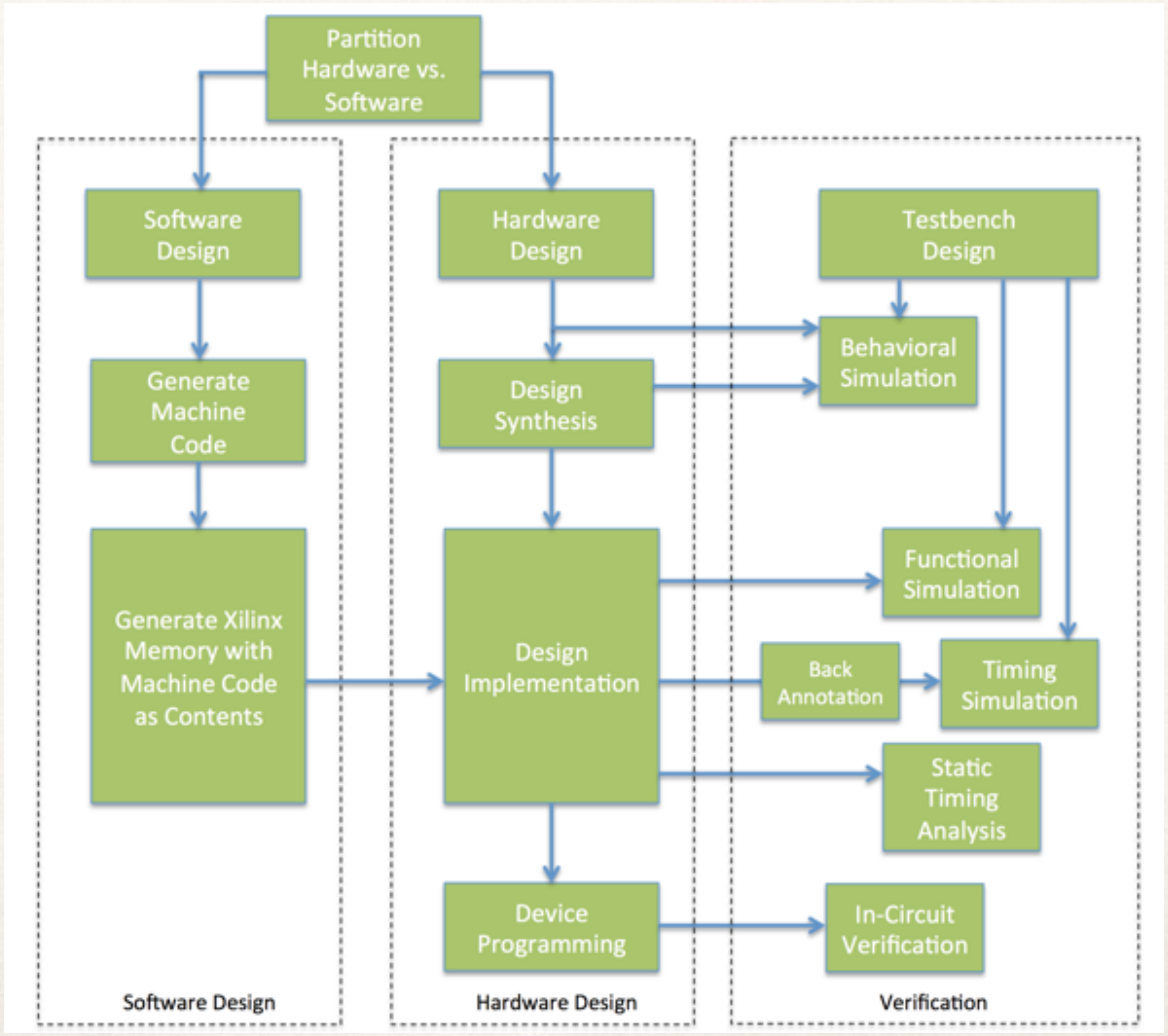
TramelBlaze

- The TramelBlaze is a 16-bit processor core designed to emulate the Xilinx PicoBlaze
- The instruction set is exactly the same as the PicoBlaze instruction set - only 16-bit operands instead of 8-bit
- The TramelBlaze is a soft-core written in Verilog
- It comes with a Python assembler for developing source code

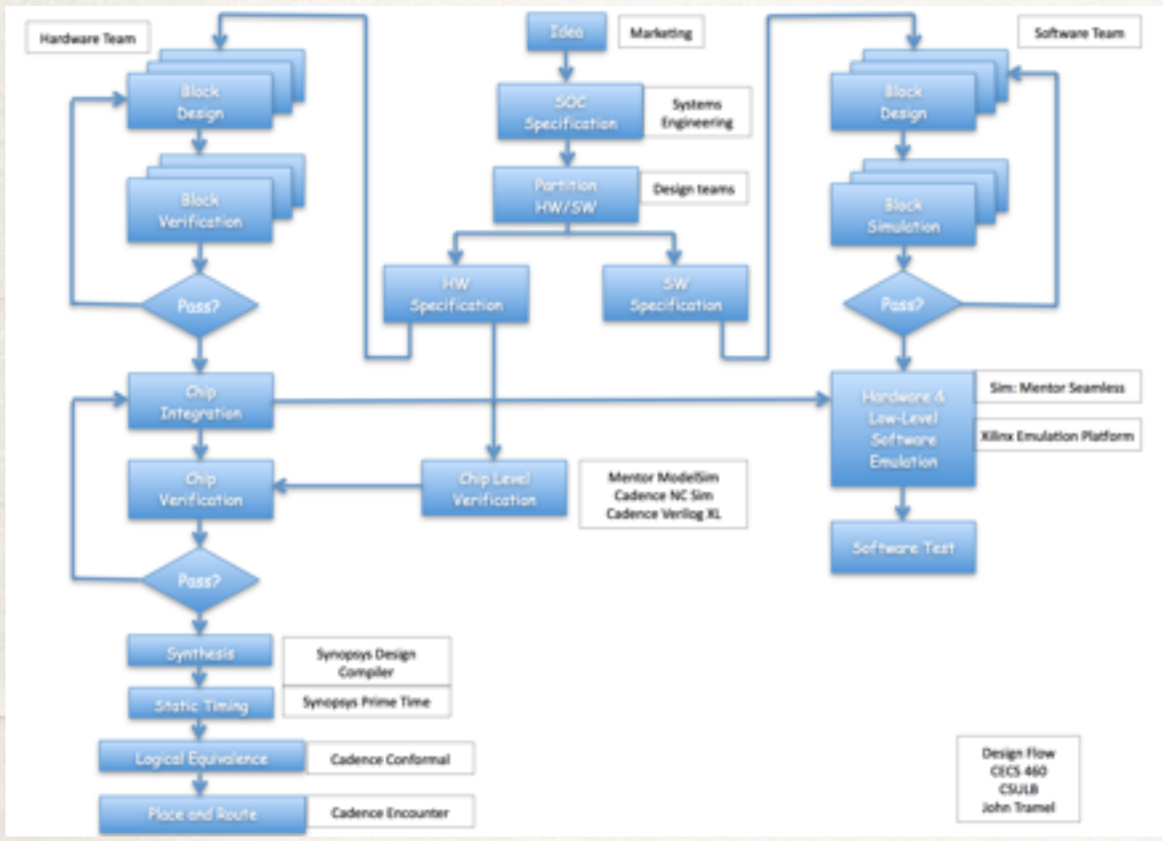
TramelBlaze Differences from PicoBlaze

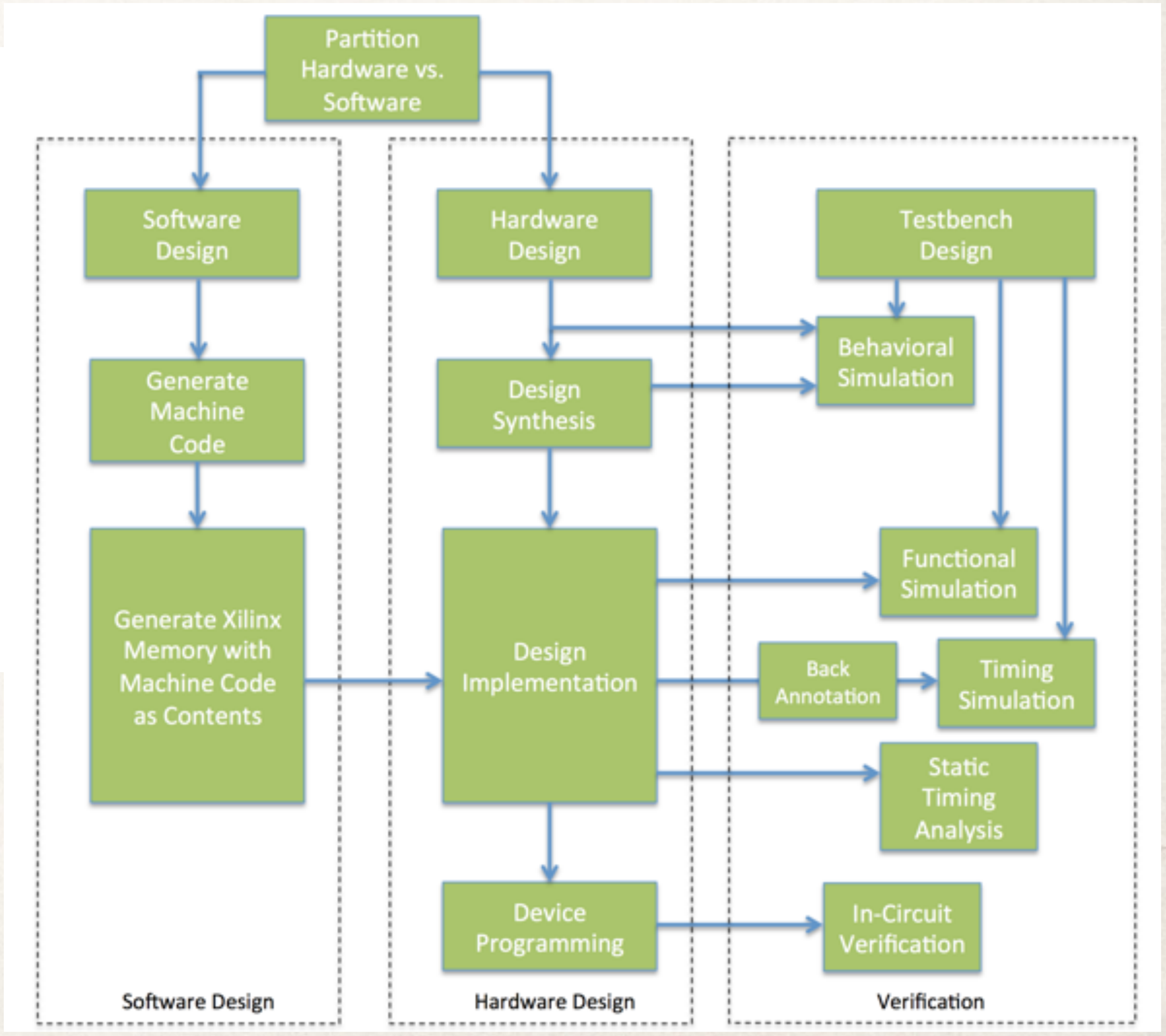
* Scratchpad RAM - 512 x 16
* Stack RAM - 128 x 16
* Instruction RAM - 4K x 16
  * Some instructions single word
  * Some instructions double word
* 64K input / output port capability

SOPC Flow - Xilinx Centric



SOC Flow Compared to SOPC Flow





Design Entry

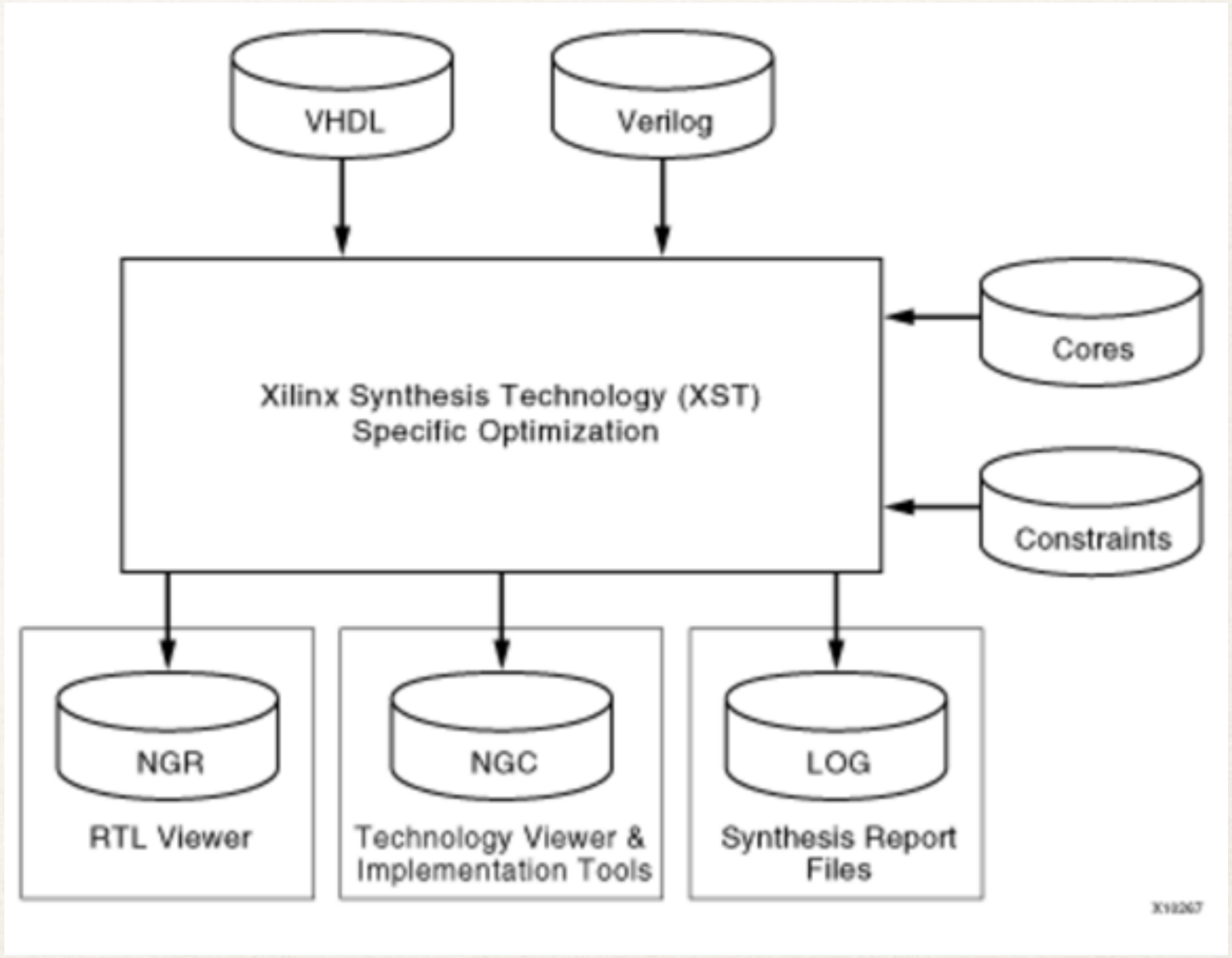
- Create an ISE project
- Create your hierarchical design
  - Instantiate IP (i.e. TramelBlaze)
  - Design your own functional blocks
- Add all blocks to the project
- Create a User Constraints File (UCF)
  - Assign timing constraints
  - Assign pin assignments
  - Assign area constraints (if any)

Functional Verification

- Create a testbench to thoroughly exercise your design
- Simulate your RTL design by running behavioral simulation - all propagation delays are ideal
- After synthesis run the gate-level functional simulation - all propagation delays intrinsic to models
- Back annotate delays using SDF file and run the gate-level functional simulation - all delays are estimated by the Xilinx place and route tool

Design Synthesis

- Referred to as "map to gates" and also "generate bit file"
- Ensure that the tools know the target technology in order to match the development board


VHDL
Verilog
Cores
Xilinx Synthesis Technology (XST) Specific Optimization
Constraints
NGR
RTL Viewer
NGC
Technology Viewer & Implementation Tools
LOG
Synthesis Report Files
X10267

Design Implementation

- Translate will merge the incoming net lists and constraints into a Xilinx design file
- Map will fit the design into the available resources on the target device
- Place and route will place your logic onto the FPGA resources and interconnect (route) all of the blocks under the constraints of the defined timing
- Programming file generation will create a bitstream (bit) file that can be downloaded to the device

Timing Verification

- The timing of the design can be verified at different points in the design flow
- Run the static timing analysis after mapping or after place & route
- Review the generated reports to uncover any timing constraint violations
  - Path delay violations
  - Setup/hold violations

Device Programming

- Using the bit file created program the FPGA
- Test the design at speed in order to confirm the design is functioning properly
- Board may be programmed to retain the configuration or it may be reprogrammed every time

Device Test

- Up till now all verification activity has been utilizing various abstractions of the design (simulation models, etc.)
- Now the device is programmed and the design must be proven to work properly by testing
- Testing runs the application at-speed with all interfaces running
- Assistance in test may be achieved through the use of test equipment - oscilloscopes, logic analyzers, Special Test Equipment (STE)
- This is a mandatory step to proving the proper operation

The Challenge of 460

- This aspect of 460 proves to be a challenge to many students
- Proper simulation is a tremendous asset to proving the correctness of a design but just because simulations pass it is not assured that the hardware will work
- Following proper methodology in creating a design is essential to properly functioning hardware
- Following proper debug techniques is essential to a quick resolution of any uncovered bugs