System On a Programmable Chip (SOPC) based Power Electronic Controller

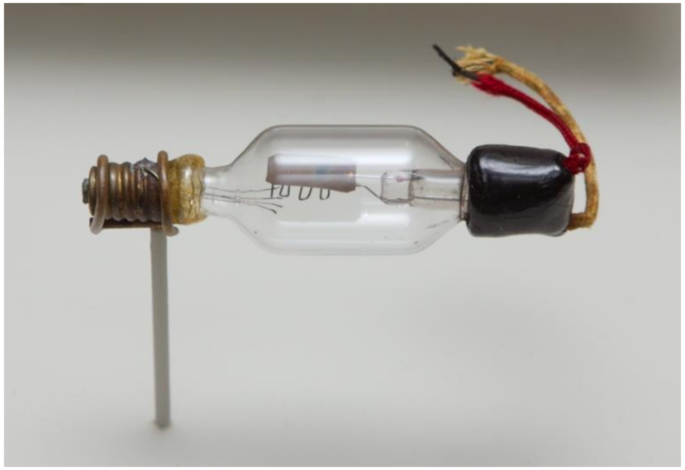
By

AJEESH A
CDAC-Thiruvananthapuram
ajeesh@cdac.in

Some Calculations

- A Processor with one million switching devices

- Use Vacuum tubes (Triodes) ,not MOS Transistors



Technical issues

Area

Assume the triode occupies an area of 5cm X 5cm including some space between adjacent tubes

- ❑ Total area of 50 X 50 =2500 m^2
- ❑ With multilayer area = 22m X 22 m

Technical issues

Power consumption

Assume

- ❑ Taking 5W per triode

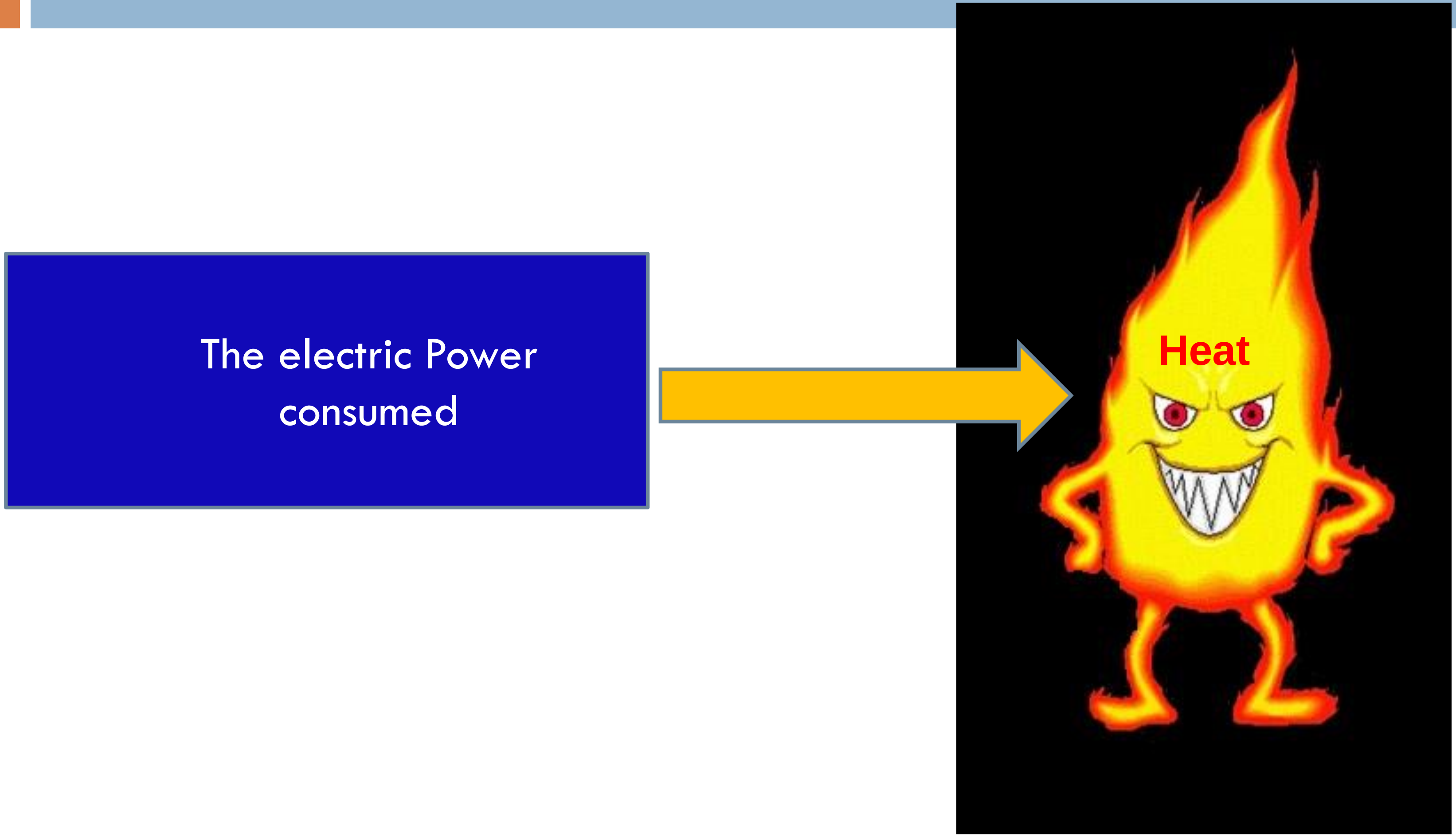
- ❑ Half of the triodes are conducting at any time

- ❑ The total power consumed $= 0.5 \text{ X } 10^6 \text{ X } 5$

  $= 2.5 \text{ MW}$

Technical issues

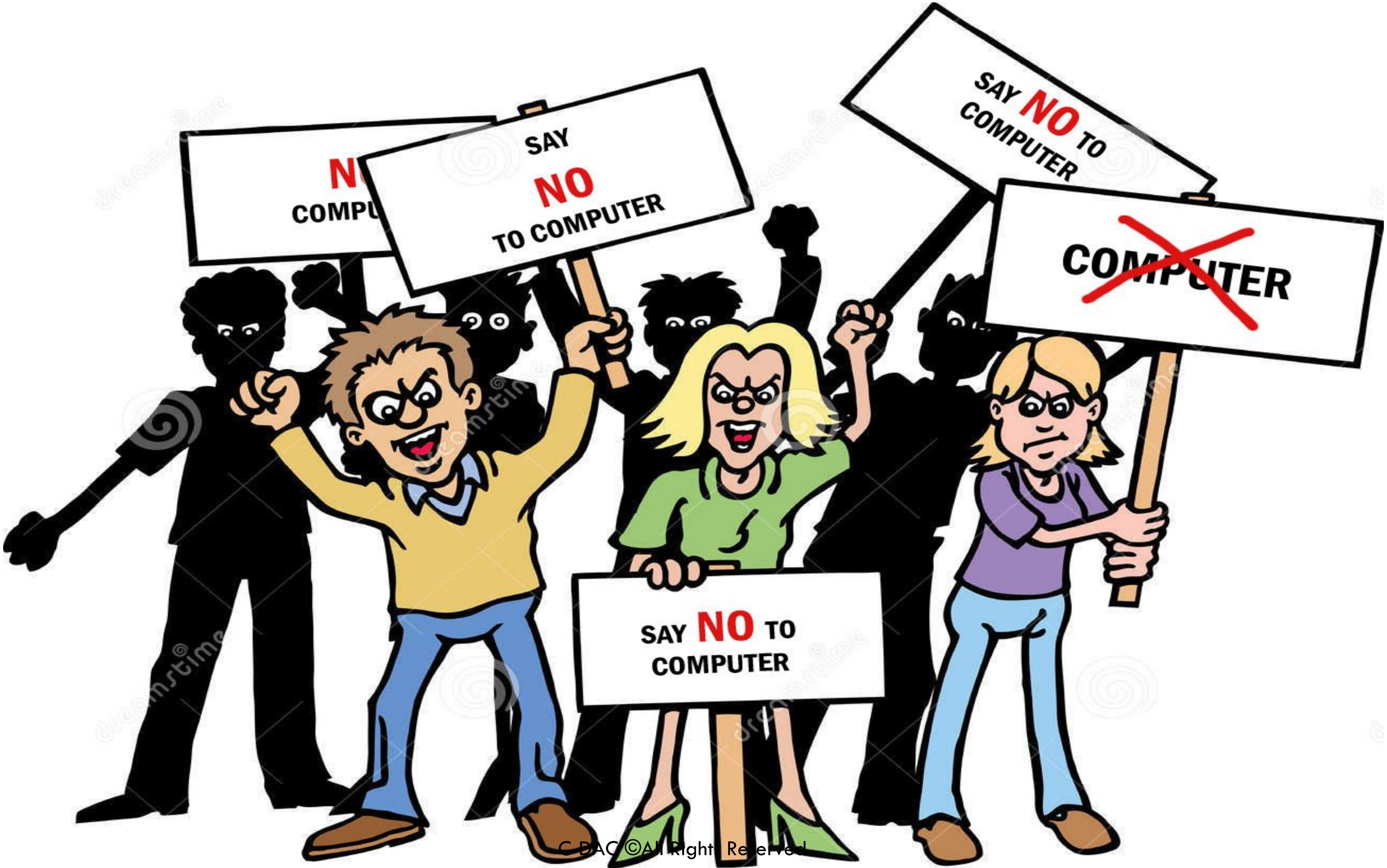
Heat dissipation



The electric Power consumed

Technical issues

Heat dissipation



Technical issues

Reliability

Let 'T' – Life of a transistor in hours

- *Replace*

on an average one triode every $T/10^6$ hr

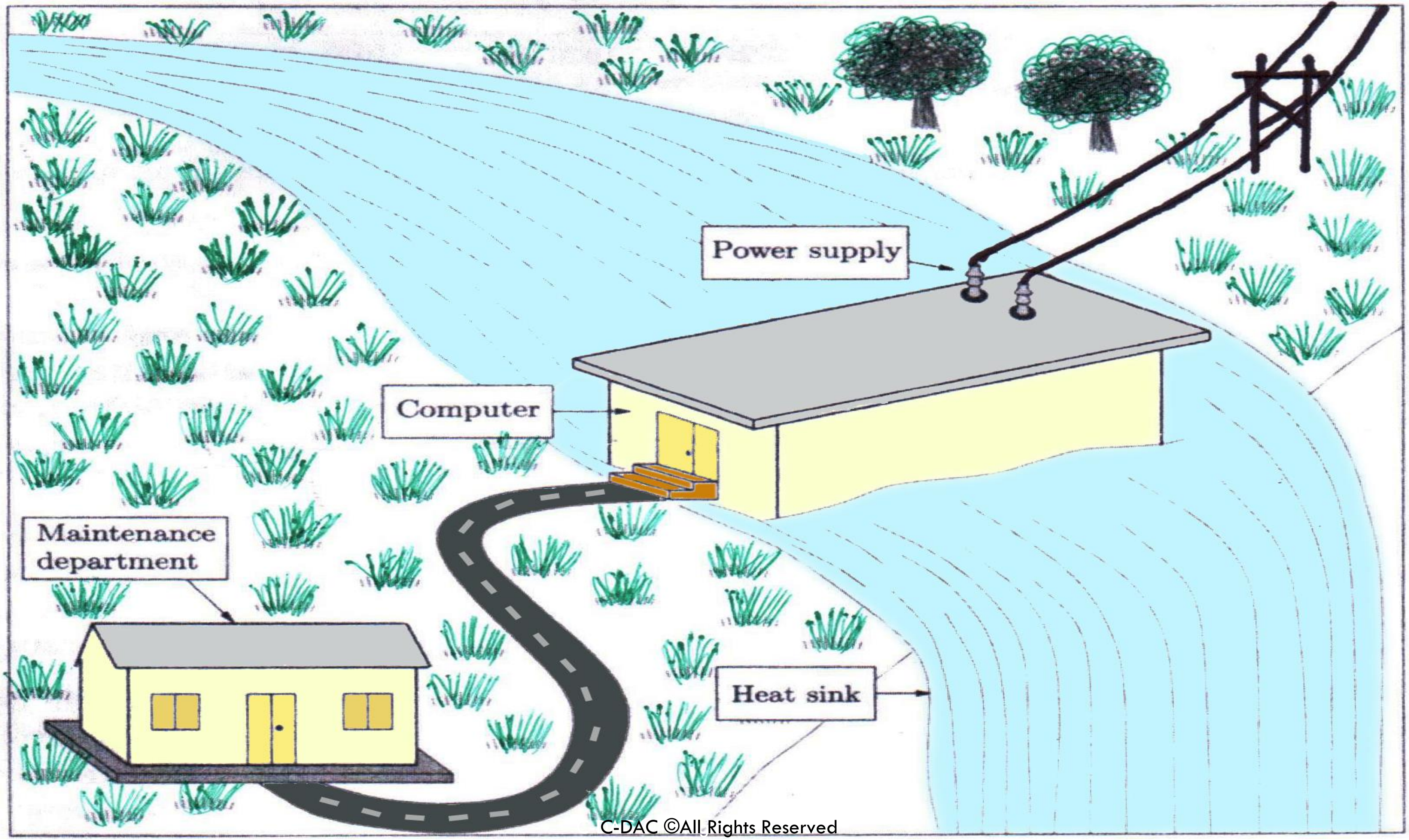
Grant T = 10^6

Then we need to replace a triode every

"one hour"

24X7 maintenance

The system



Technical issues

Speed

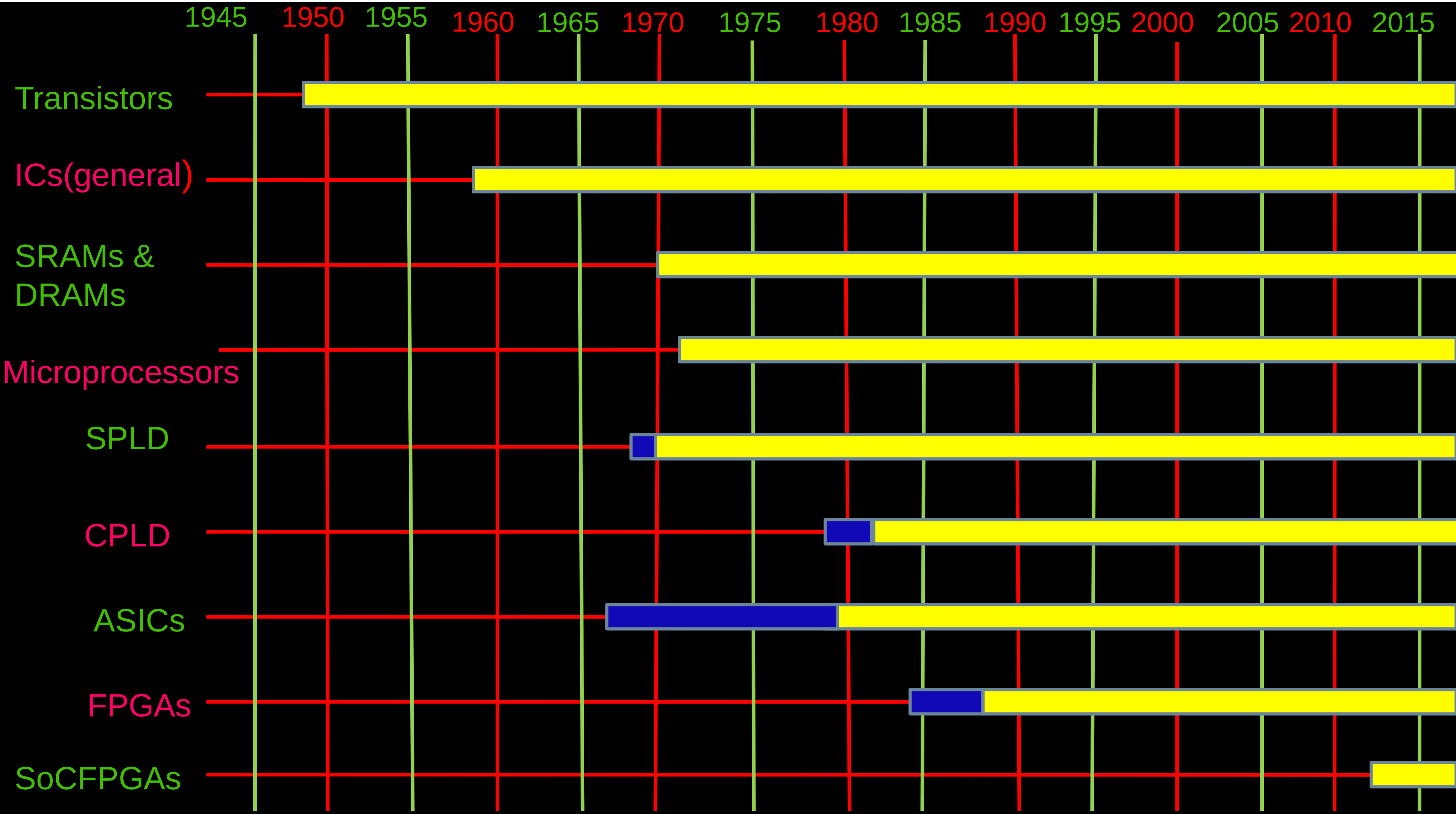
Below 1 MHz due to parasitic wiring capacitance and Inductance

Longer development time

Summary

- Area/Size
- Power Consumption
- Reliability
- Speed
- Time to market
- Cost

Programmable chip



FPGAs

1980s

PLDs

The Gap

ASICs

➢configurable
➢Fast design and modification time
➢Could not support large or complex functions

support large or complex functions
Expensive and Time consuming
➢Frozen in Silicon

FPGAs

1984s

PLDs

FPGAs

ASICs

- ➢Highly configurable
- ➢Fast design and modification time
- ➢Could not support large or complex functions

- ➢support large or complex functions
- ➢Expensive and Time consuming
- ➢Frozen in Silicon

Xilinx Introduced FPGAs in Market
CMOS – SRAM Based

FPGAs

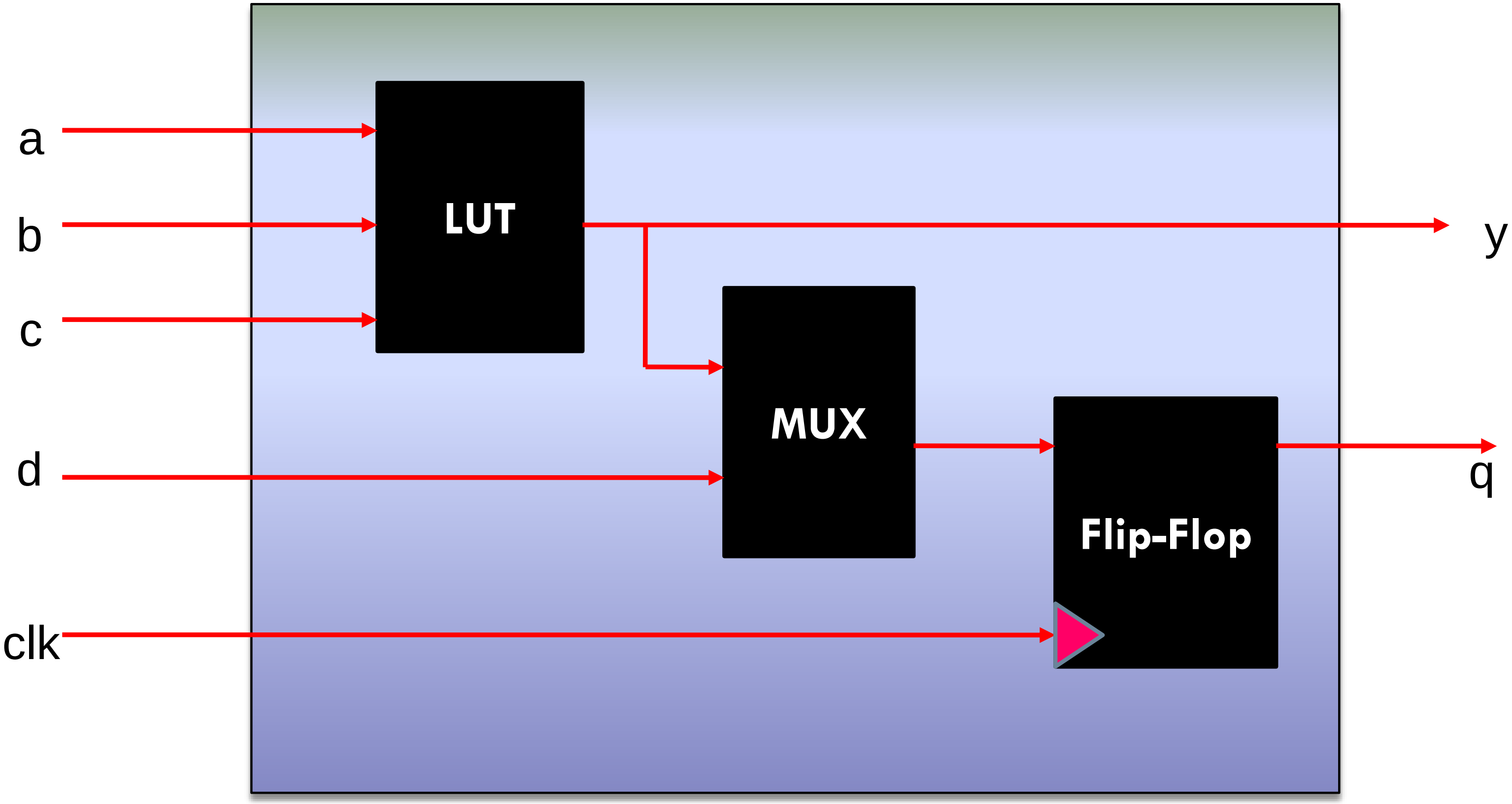
Programmable logic Blocks

- ➢ A 3 input Lookup table (LUT)
- ➢ Flip-Flop or Latch
- ➢ A Multiplexer

Modern FPGAs Contains complex versions of “Programmable logic Blocks”

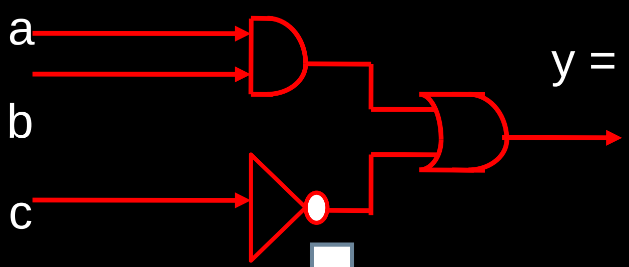
Using SRAM every logic block in the device can be programmed

PLB



Configuring LUT

Required Function



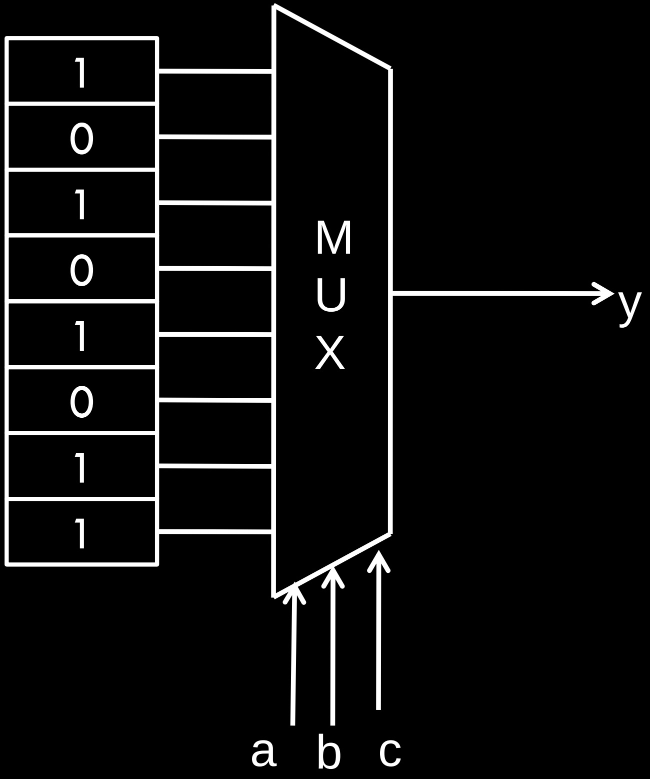
y = (a AND b) OR !C

Truth Table

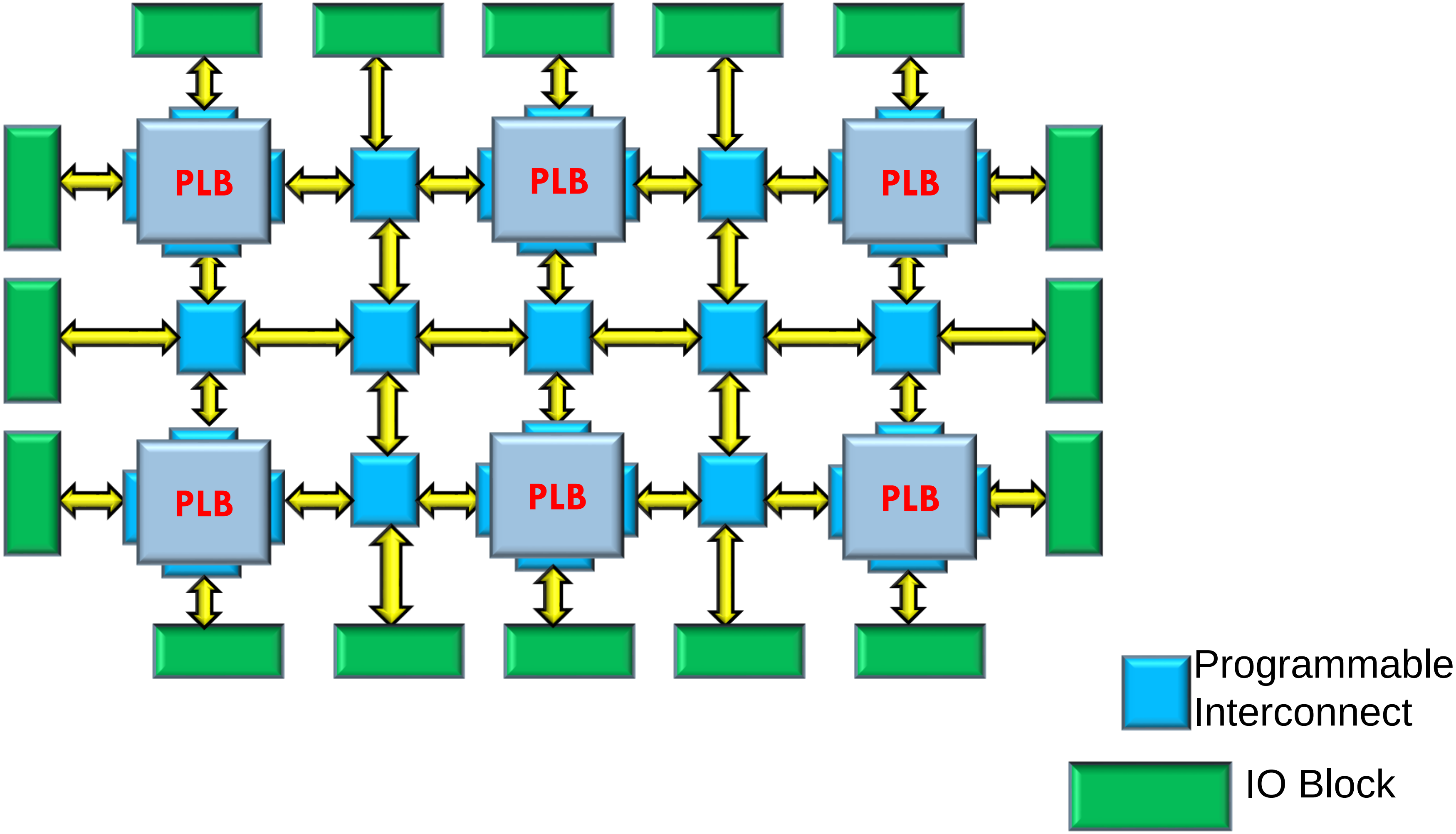
a b c	y
0 0 0	1
0 0 1	0
0 1 0	1
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	1

Programmed LUT

SRAM CELLS

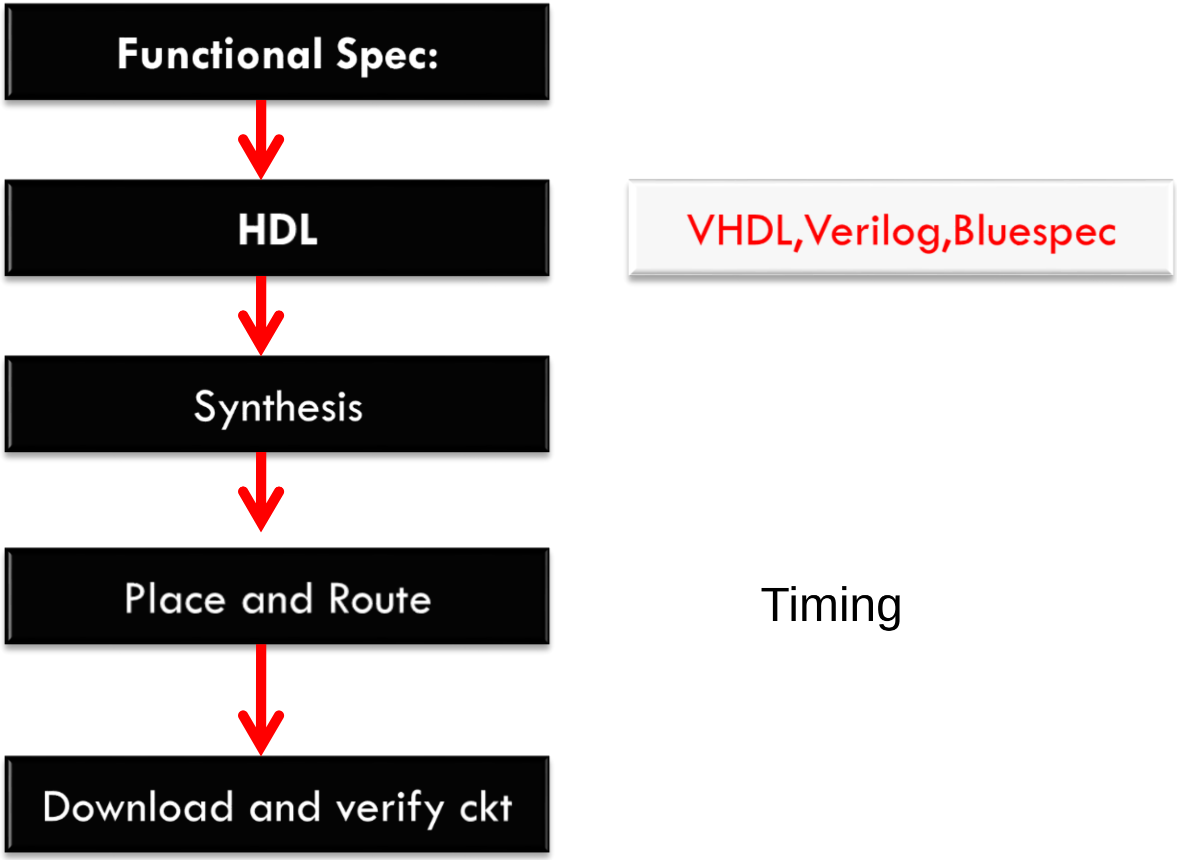


FPGA Structure



Complex architecture today

FPGA Design Flow



Functional Spec:

HDL

VHDL,Verilog,Bluespec

Synthesis

Place and Route

Timing

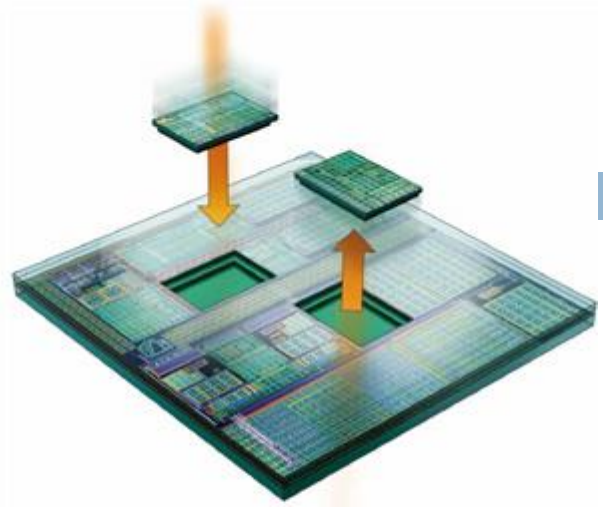
Download and verify ckt

PE Control System Design

❑ Analog – Using OP-AMPS, Resistors, Capacitors etc.

❑Digital – Using Processors and software programming

Logics and numerical methods – DSPs



“What if have a Processor so that user defines Instructions, Internal Modules, Speed, Memory, pins etc. ???”

Obsolescence free design!!!

SOPC

- “Building embedded system into a single programmable integrated circuit like FPGAs”

- Generally involves utilization of a large FPGA
  - Processor cores (Soft / Hard)
  - Memory(RAM and ROM)
  - Intellectual property (IP) ,Custom hardware blocks/IPs

“have all (or majority) of the components on a single programmable chip”

Need for Development

Long term support for Hardware and software

Industrial PE systems, Power system controls etc. (5 -10Y)

Railways, Defense Application (10-30Y)

- ***Processor Obsolescence*** in 4 - 6 years
- ***Next generation Processor :*** Software porting overhead according to Processor Hardware changes

***soft* processor IP core in general purpose FPGA**

Single chip Solution:

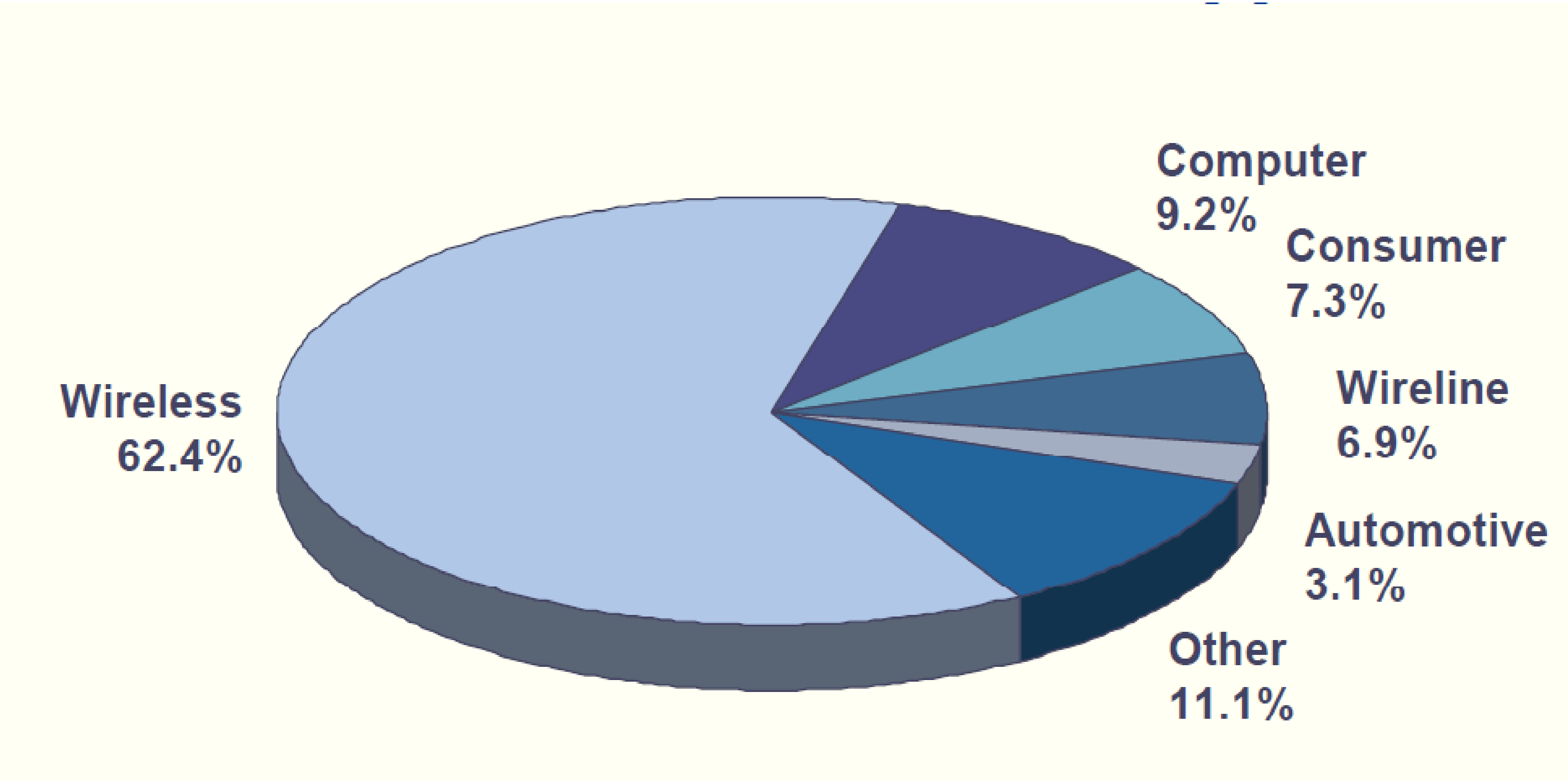
Difficult to find a processor with all interfaces/reconfigurable for the application

Custom made IPs with the CPU core

Single core to multi core : with increased cost, may require hardware redesign

same FPGA Hardware for single to multi core application

Need for Development
DSP Revenue



Nobody is going to use a mobile more than 20 Years

Need for Development

Faster Controller : Sampling of 5 -10 microsec or less

- ***Use Parallel processing*** *– multi-core processor integration*
- ***Use Hardware accelerators*** - Control blocks in dedicated hardware to perform functions faster than a CPU (PI, Filters, PLL, PWM)
- ***C2H Compilers:*** Inspect the application 'C' code and try to convert time consuming software algorithms into equivalent hardware implementation

Hardware performs faster than software

- ***Hardware Parallel Processing than Software pipelining***

8201 NIOS II cycles equivalent to 24603 TigerSHARC cycles

(200 MHz vs 600 MHz)

Better Time to Market criteria

- ❑ ***Same FPGA Board***
- ❑ ***Proven CPU and IP core Set***

handles a wide range of applications

Need for Development

Higher reliability

- *Higher level of integration and resources utilization of a single FPGA*

Board Design

On chip interconnections : ***Industry standard (AVALON, AMBA AXI)***

Internal PLLs : ***Remove the need to distribute high speed clocks round a PCB***

Reduced EMI

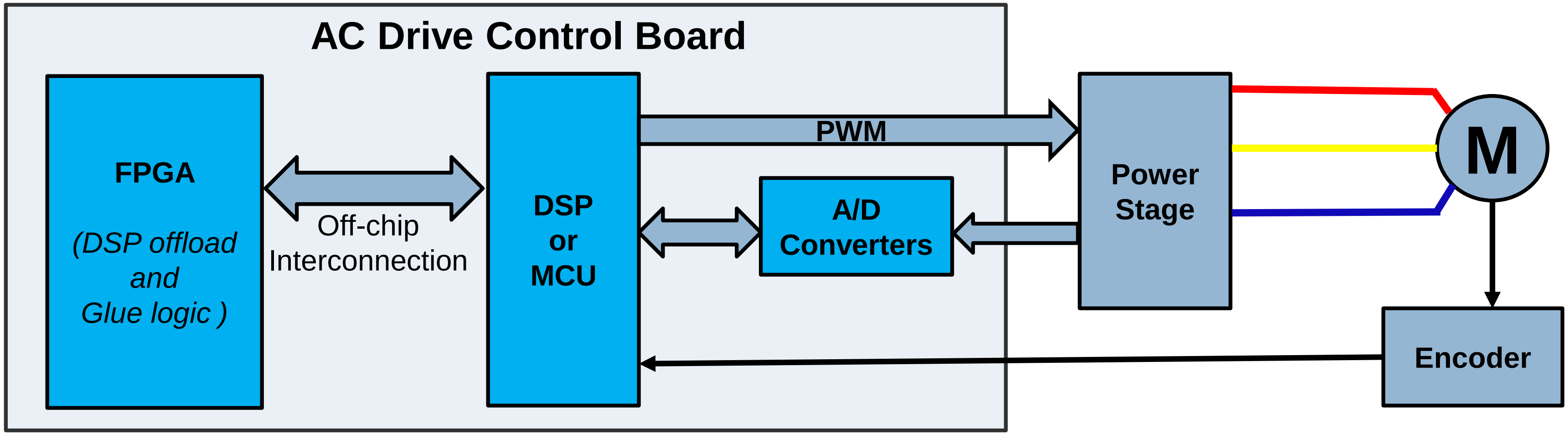
“PCB can be smaller, consume less power, be easier to get through EMC tests”

Board Testing

“Dedicated test designs can be loaded into the FPGA to automate Board testing”

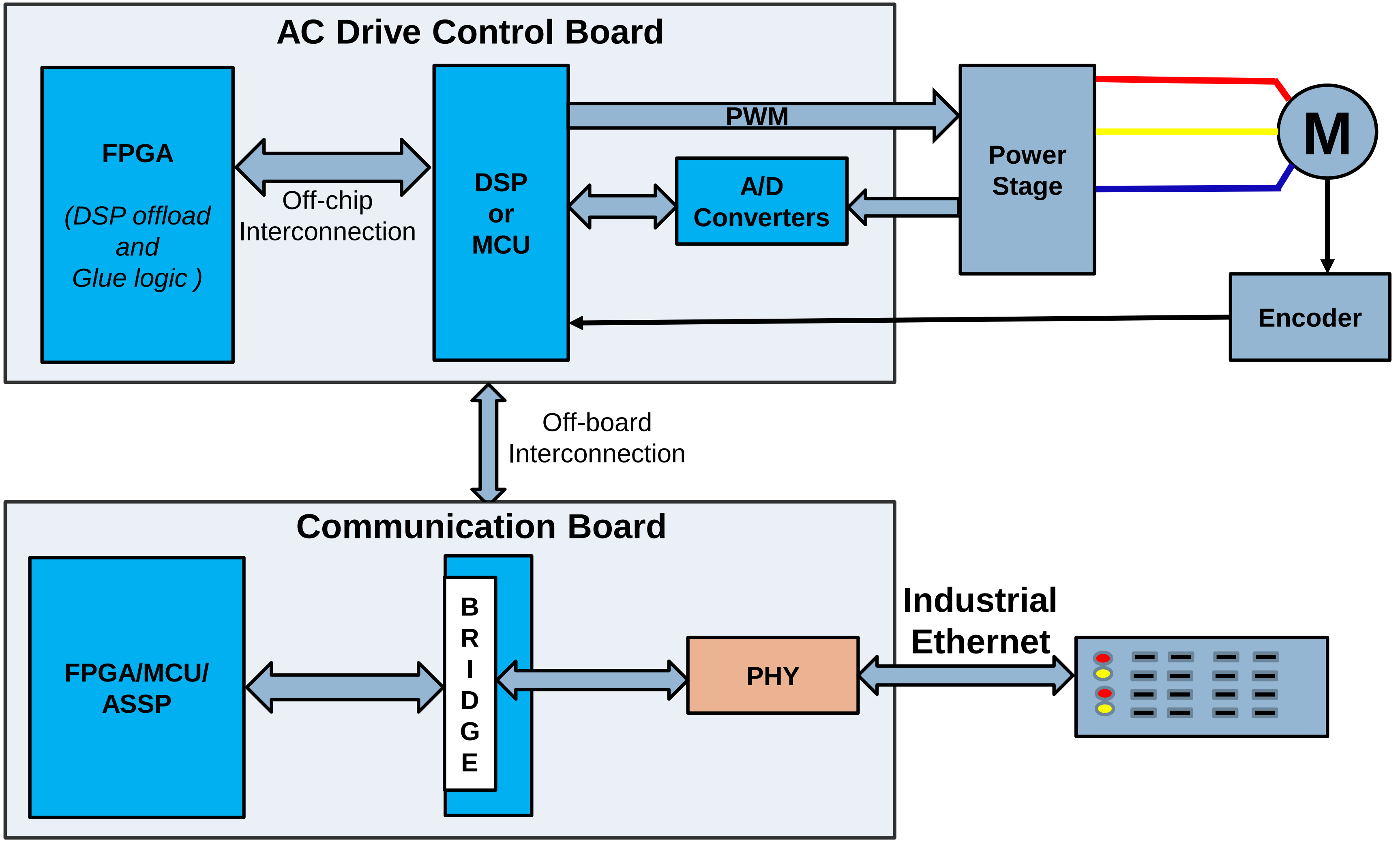
Conventional Controller Design

CONVENTIONAL CONTROLLERS

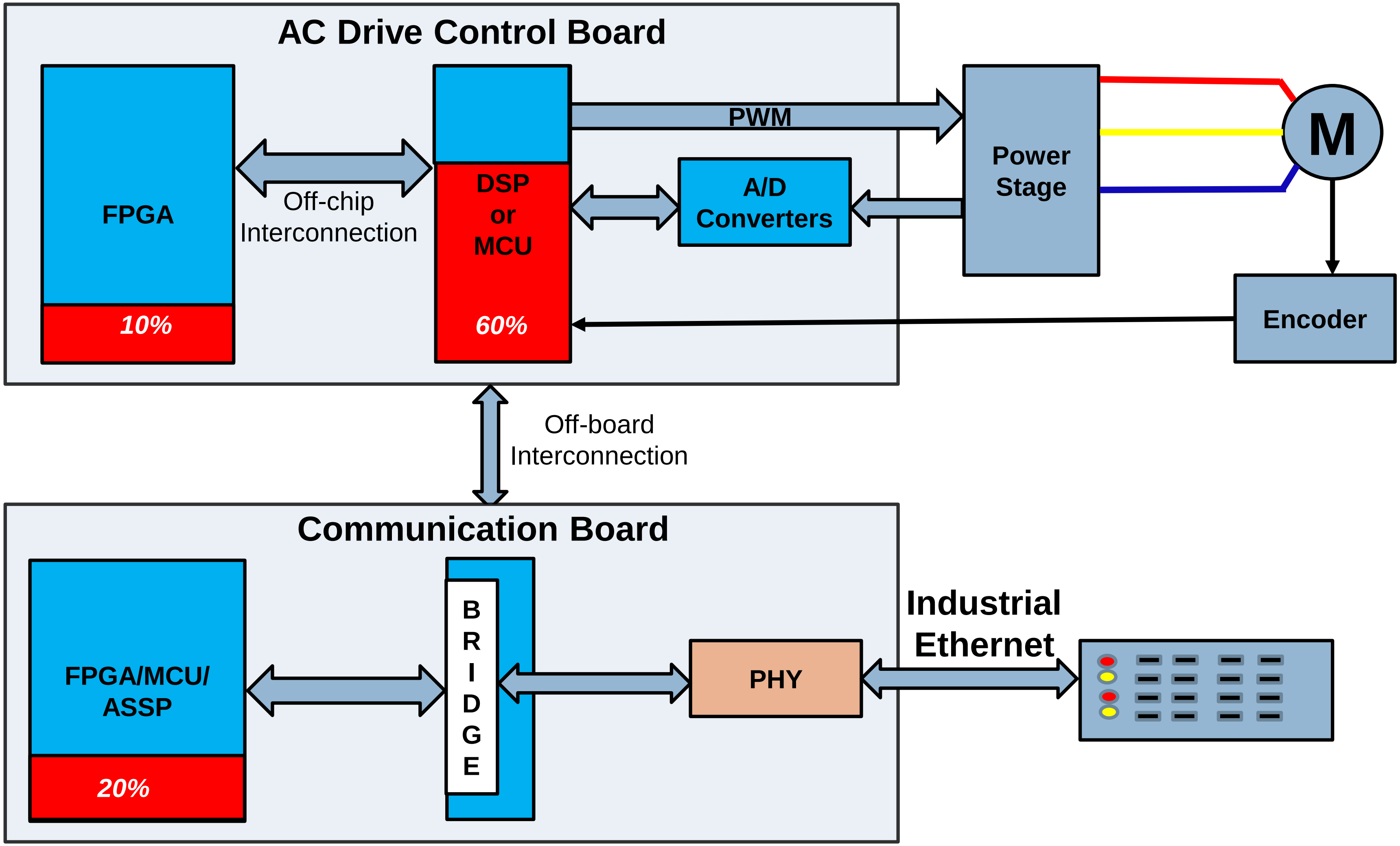


Conventional Controller Design

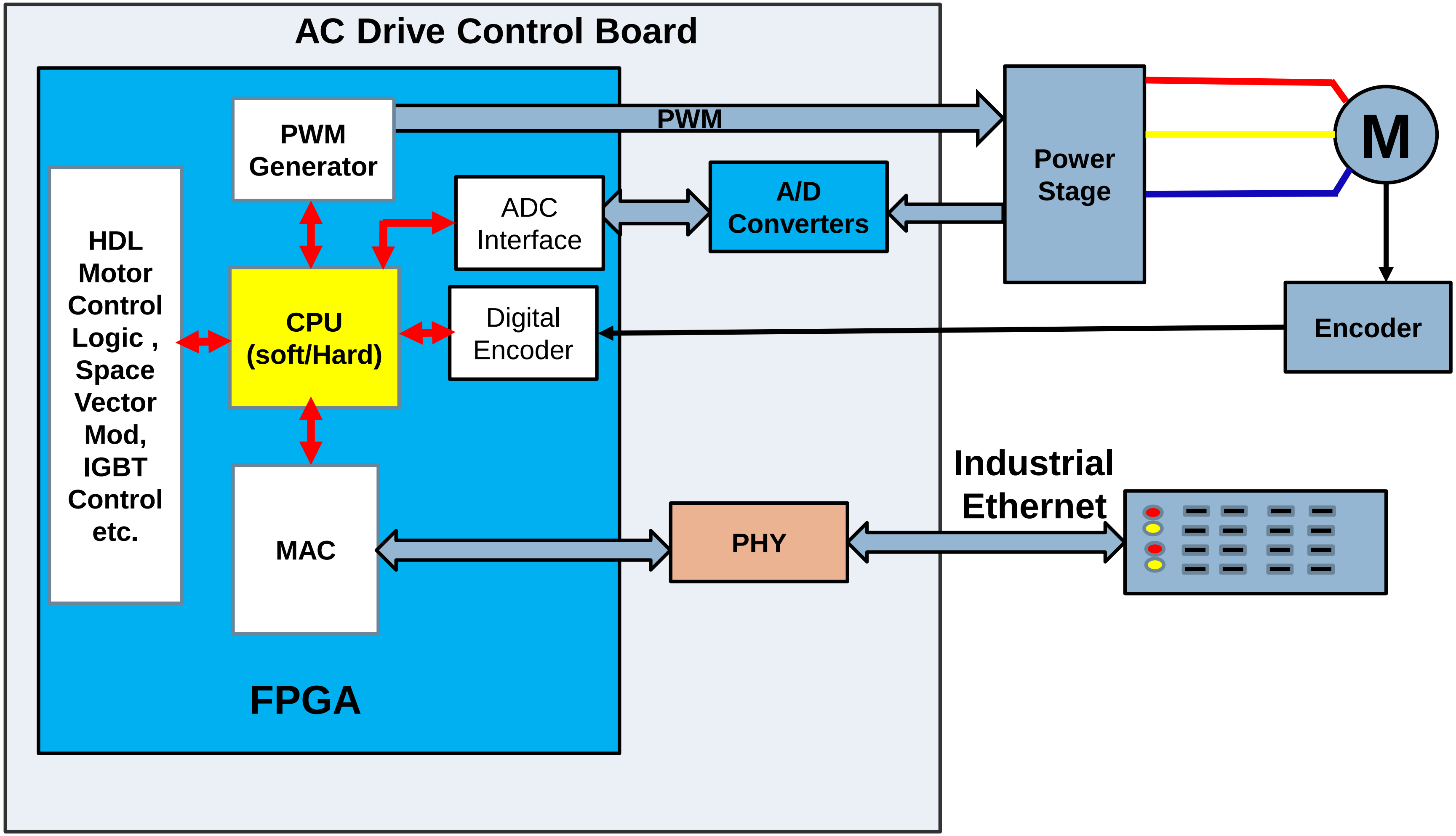
CONVENTIONAL CONTROLLERS



Resource Utilization

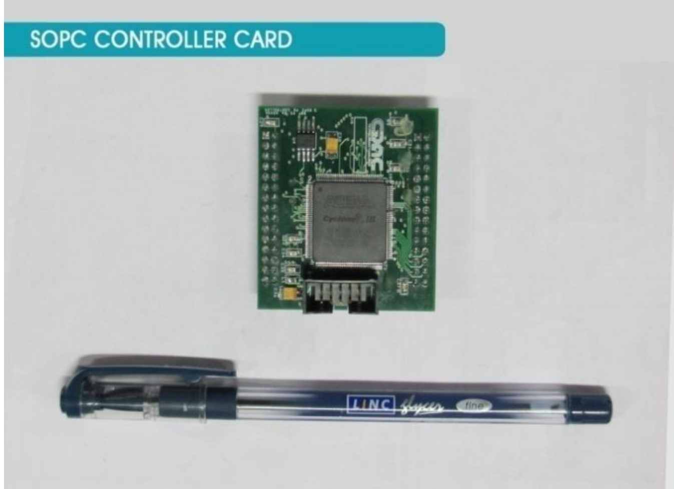


SOPC PE Controller

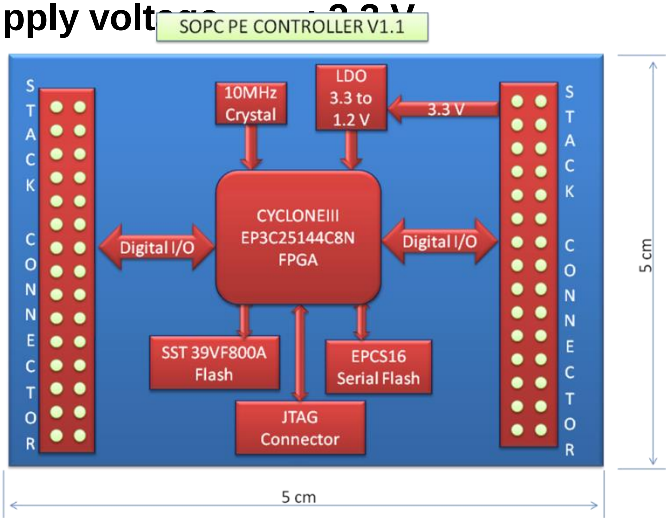


FPGA as a Drive Controller Chip

SOPC PE Controller Board V_1.1

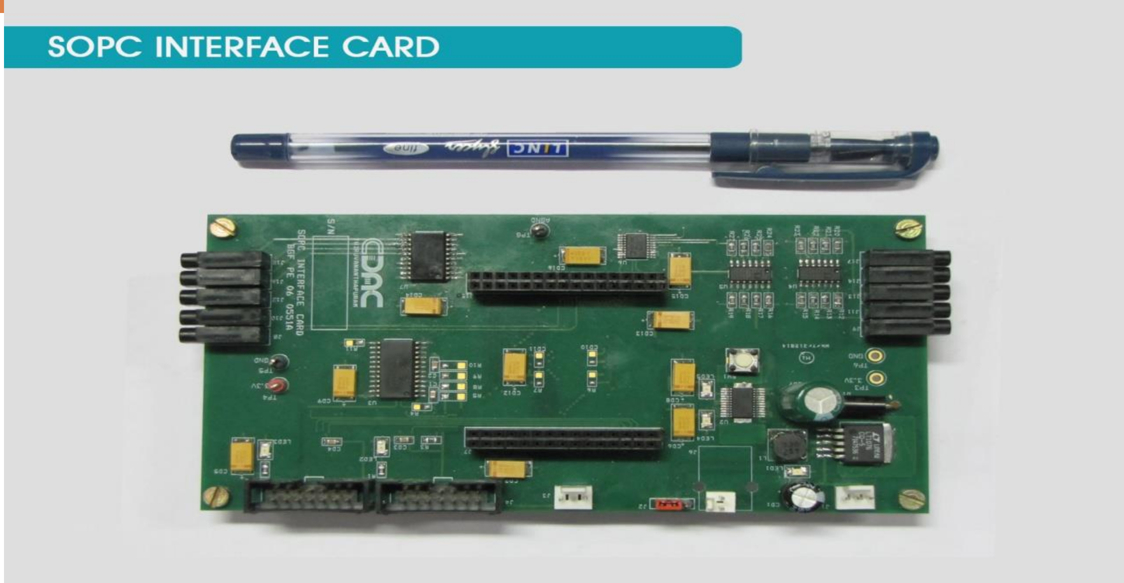


	Specifications
❑FPGA	: Cyclone III EP3C25E144C8N (24,624 LEs)
❑On chip Memory	: 64 kBytes (Inside FPGA)
❑Flash memory	: 2 MB
❑Digital I/Os	: 56 No.s 3.3-V LVTTL - configurable for ADC/DAC, PWM ,I2C interfaces
❑Host interface	: JTAG
❑Supply volt	



- ❑ This Generic Board can handle Most of the PE applications
- ❑ The Hardware inside the FPGA is Configurable depending on PE Application
- ❑ The I/Os are Configurable

Peripheral Interface Card



Targeted Application

• DC-DC converter control
•AC drive applications
•Front end converter etc.

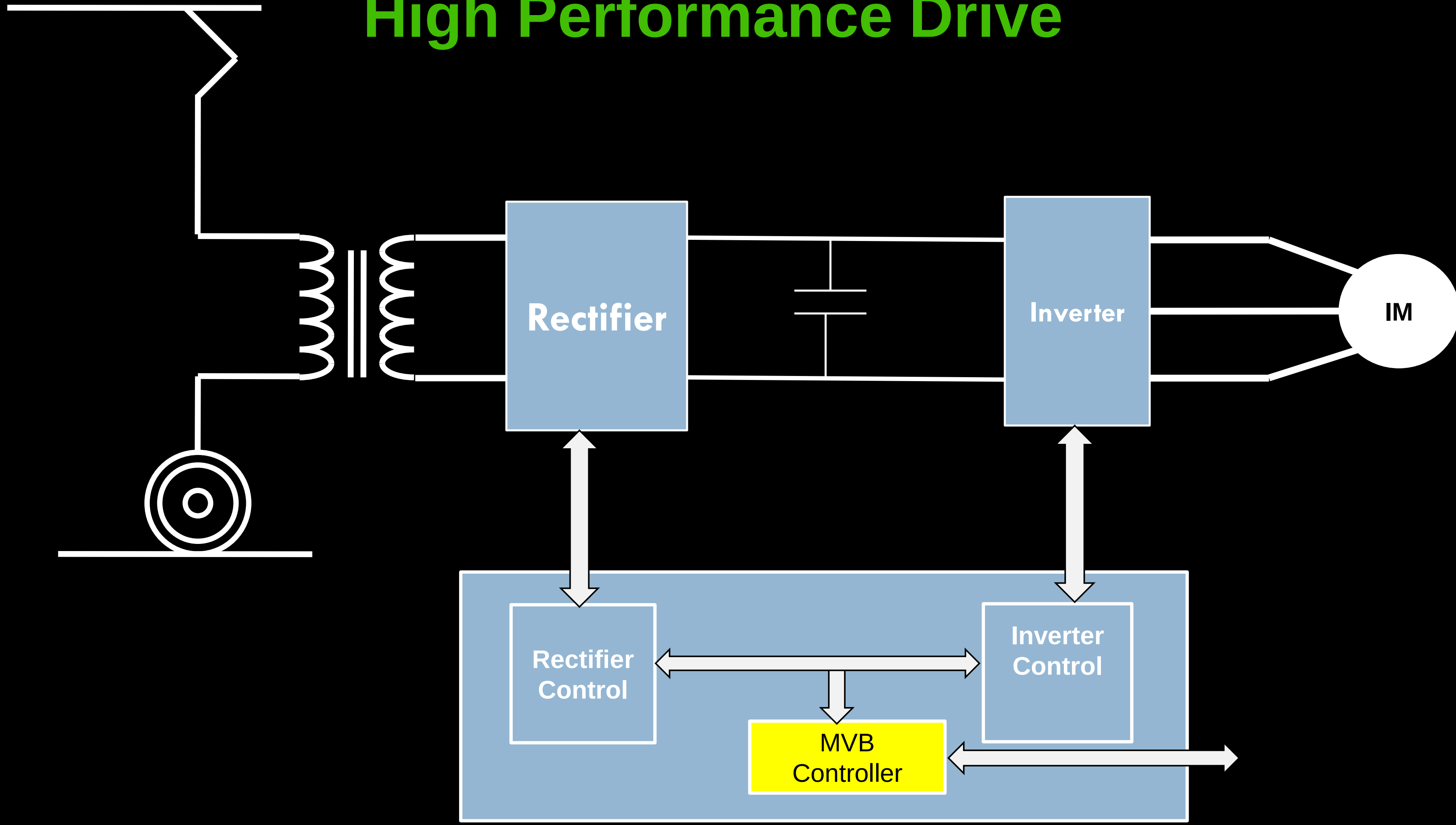
❑**Specifications**

➢Analog Input : +10V (ADC 13 bit 8 Channel, SPI interface)
➢Analog Output : +3.3V (DAC 12 bit 8 Channel, SPI interface)
➢Digital input : +5V
➢Digital Output : +5V
➢Supply voltage : 24 V
➢Stack Connector for SOPC_CONTROLLER Card

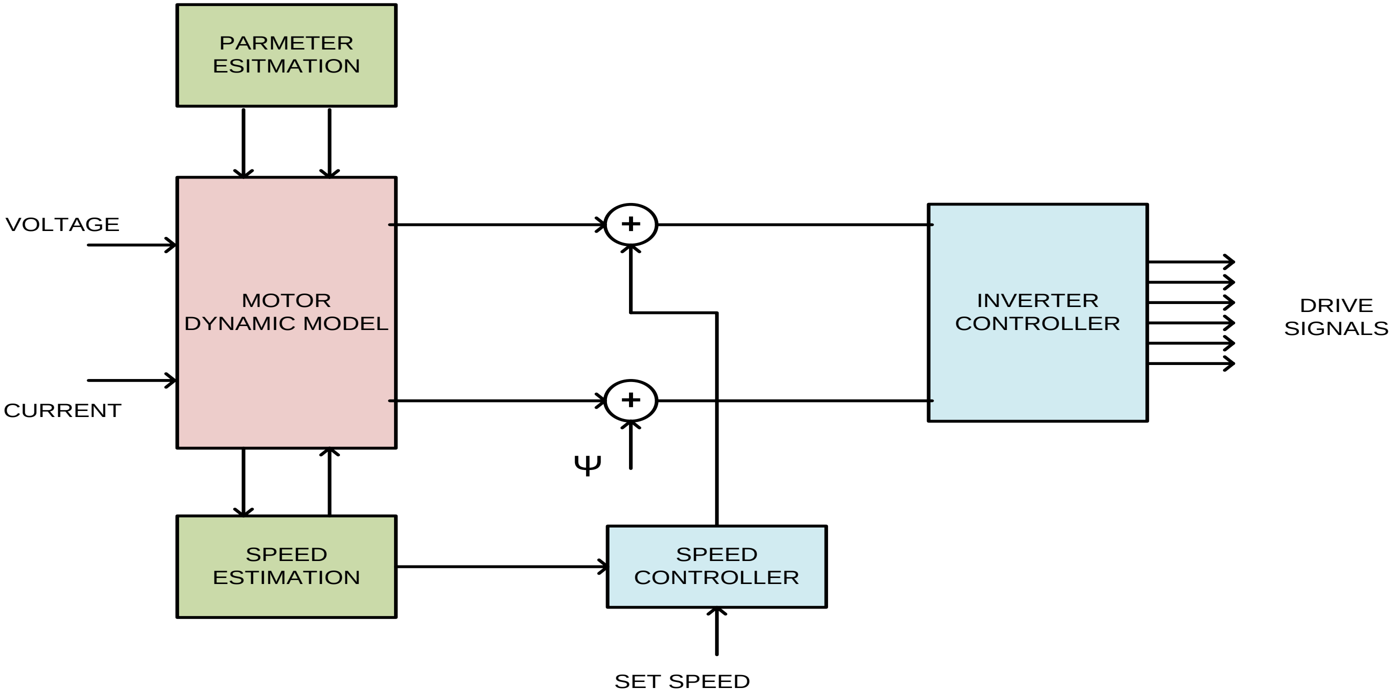
"CUSTOM DESIGN FOR SPECIFIC APPLICATION IS POSSIBLE"

SOPC Configuration

High Performance Drive

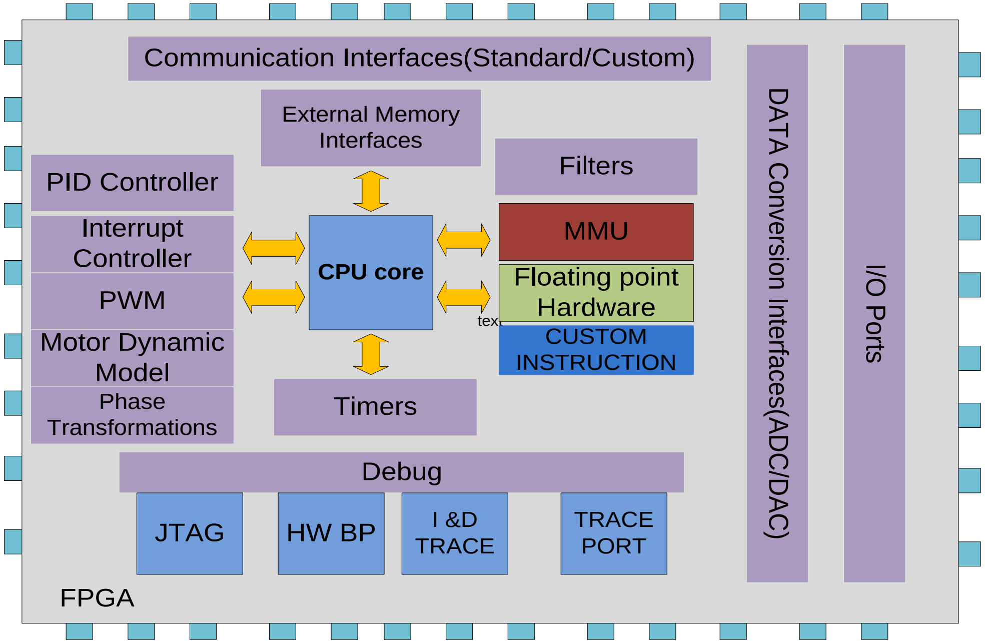


Inverter Control



- ❑ **Motor Model Estimates The speed from Voltage and currents**
- ❑ **DSP/Micro controller implementation, the motor model and other control blocks will be implemented as a software program.**
- ❑ **In SOPC architecture, control blocks are realized in Hardware using HDL called IPs**

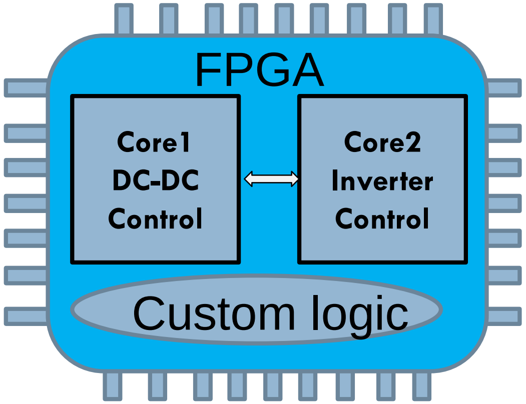
SOPC Configuration For Inverter Control



- ❑Control blocks now part of the CPU **– Hardware Modules**
- ❑Other similar PE applications with out redesign **-“Reuse”**
- ❑Can be accessed Using Custom Instructions/Functions

IPs Required for The application

SOPC Configuration for **DC-DC Control**	SOPC Configuration for **Inverter Control**
IP cores for Basic control	
Soft Processor core	Soft Processor core
PI Controller	PI Controller
PWM(2 Channel)	Motor Dynamic Model
Subtraction	PWM(6 Channel)
Limiter	Phase Transformers



Multi processor core Implementation in one FPGA Possible

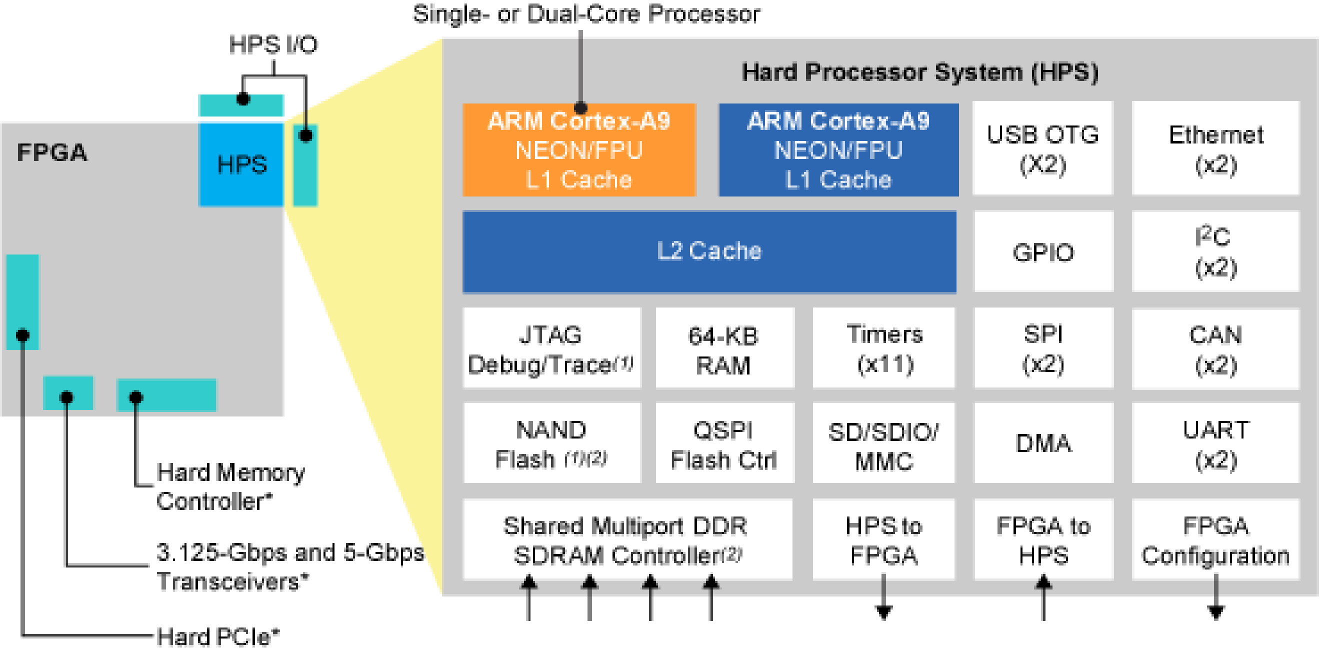
Processors in FPGAs

- Soft or Hard CPU cores

- Soft Processor
  - Processor implemented in VHDL, Verilog, etc., and downloaded onto FPGA hardware
    - Example: NIOS II, MicroBlaze, ARM Cortex-M1 etc
    - Highly Reconfigurable ,a schematic (or code like software).

"More than 20 soft core processors are available"
LEON 3, S1 core (64 Bit), ARM Cortex –M1,DSPUVA-16 etc

Hard processors on FPGA



*Optional Configuration

❑SoC FPGA

❑FPGA + Hard Processor System (HPS)

❑Can’t alter Placement , Routing, and Area

❑Cyclone V SoC from Altera

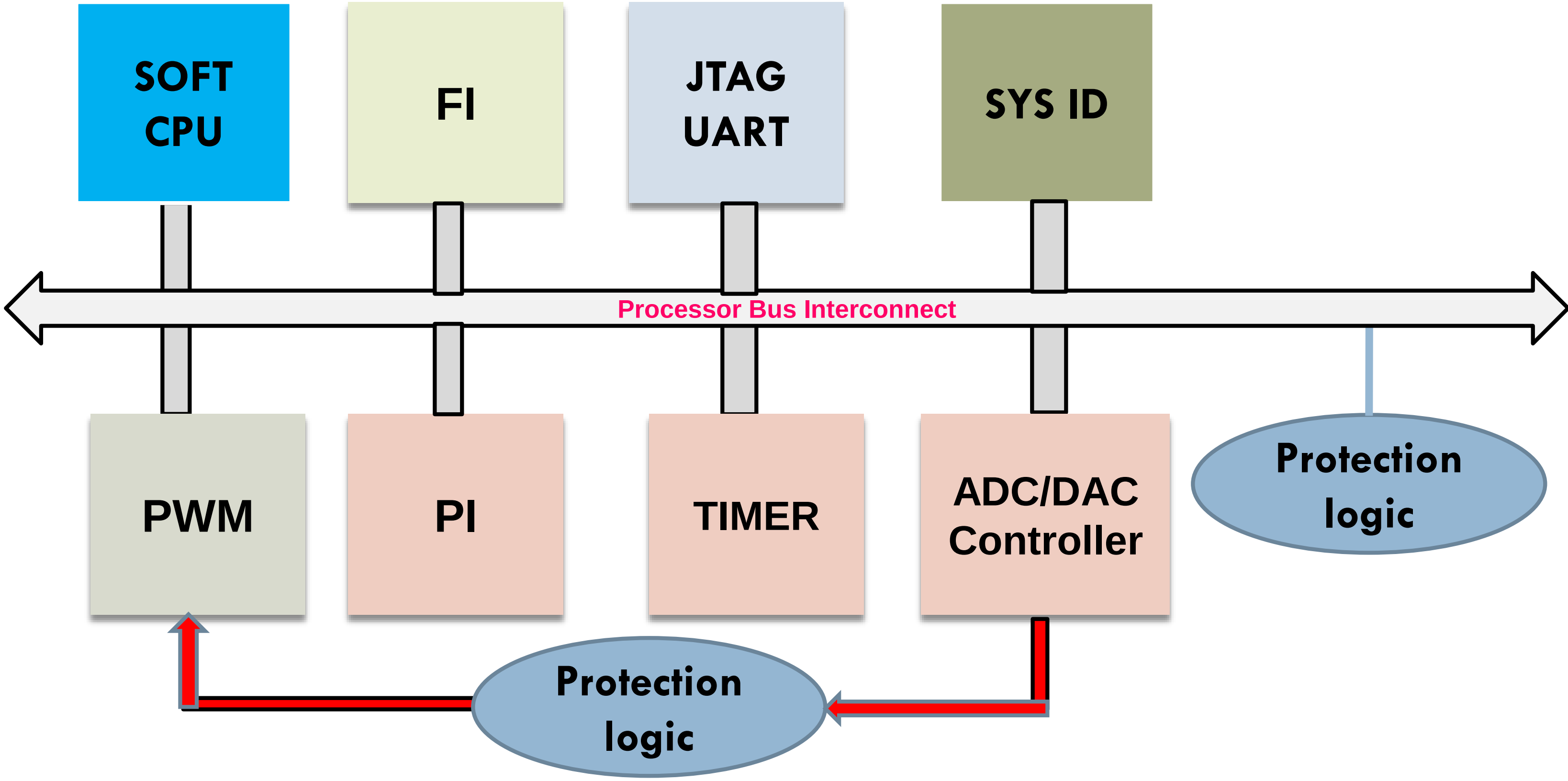
❑Optimized for Higher speed of operation 925 MHz

❑Costly

ALTERA NIOS II CPU

- Full 32-bit instruction set, data path, and address space
- 32 external interrupt sources
- Single-instruction 32 x32 multiply and divide producing a 32-bit result.
- Access to a variety of on-chip peripherals, and interfaces to off-chip memories and peripherals
- Speed : 20 MHz to 430 MHz

Power Electronics Processor system



❑The top level entity of the IPs should meet AVALON Specification for NIOS

❑For ARM, AMBA Bus specification

Development Of IPs- SOPC IP Developer

- **The Control blocks should be designed in HDL**
- **Functionality and timing are properly monitored using test bench or IP Verification methods.**
- ***Time consuming process but one time process***

Technology Developed

- **PE IP Library**
  - ➢PE Specific IPs
- Design procedure for the IP Developer
- Drivers for PE IP library
- Design procedure for SOPC user
- SOPC Controller Card
- SOPC Interface Card
- PE Application Evaluation Report

Application Development in ALTERA FPGAs

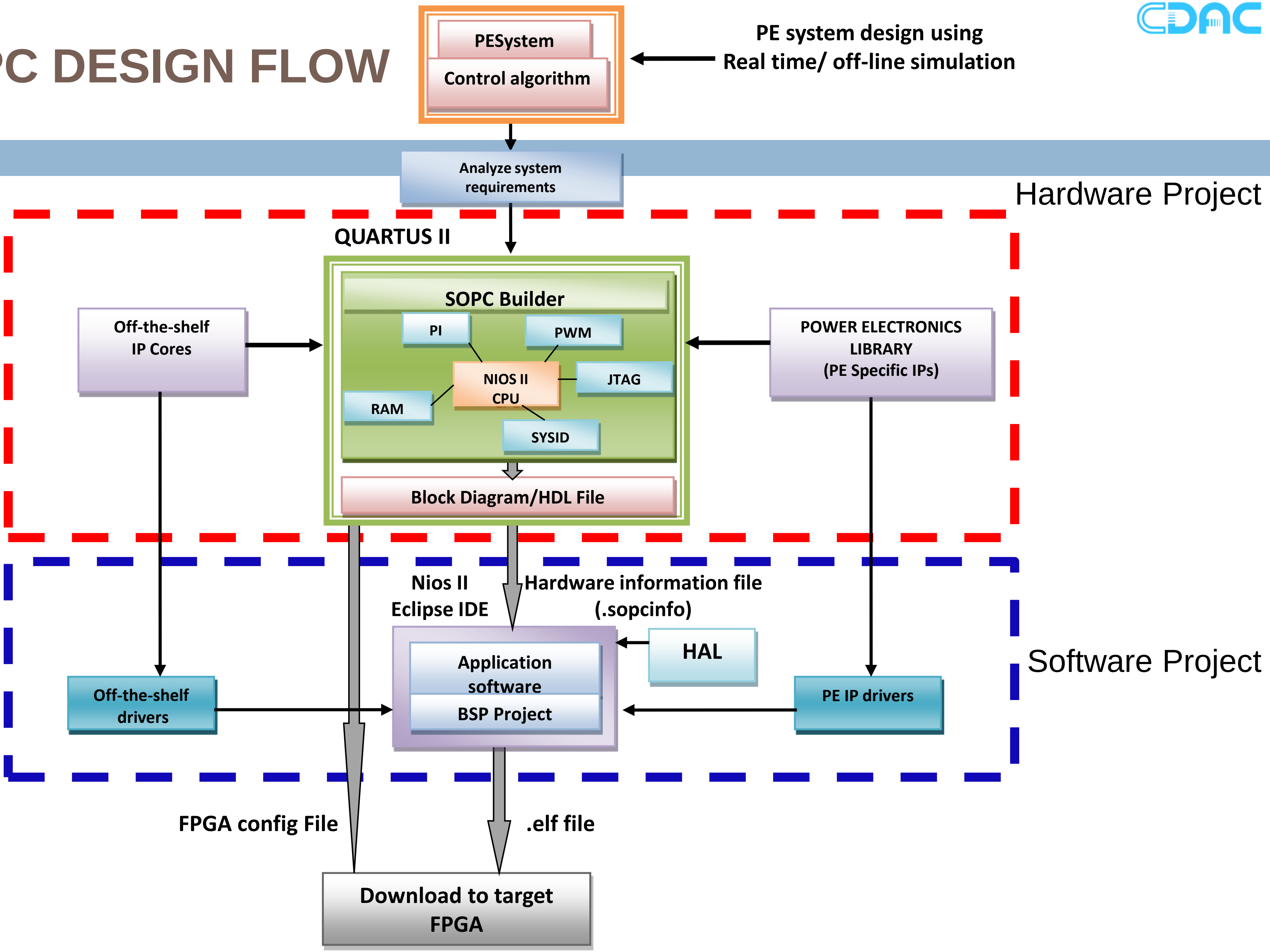
"Faster Concept to System"

- Tools
  - Altera QUARTUS II V9.1 or higher with SOPC Builder/Qsys
  - NIOS II Eclipse IDE for software development
  - Model sim for Simulation
- Hardware

  SOPC_PE CONTROLLER_V1 card

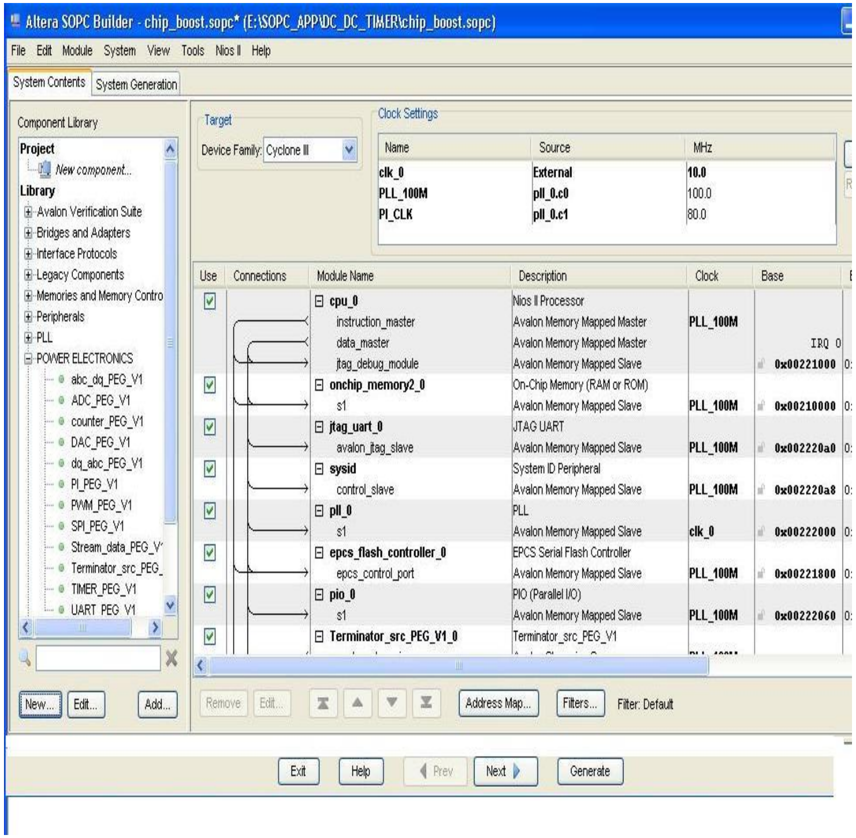
  SOPC Peripheral Board

SOPC DESIGN FLOW



SOPC Hardware Project

- ❑ Drag and drop approach



Target

Device Family: Cyclone III

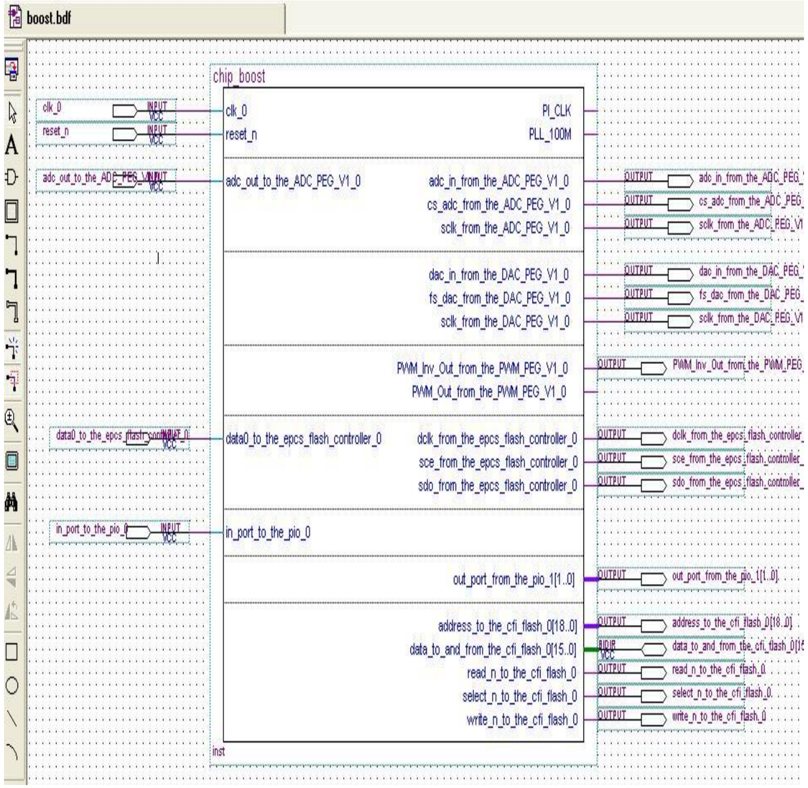
Clock Settings

Name	Source	MHz
clk_0	**External**	**10.0**
PLL_100M	**pll_0.c0**	100.0
PI_CLK	**pll_0.c1**	80.0

Use	Connections	Module Name	Description	Clock	Base
☑		**cpu_0**	Nios II Processor		
		instruction_master	Avalon Memory Mapped Master	**PLL_100M**	
		data_master	Avalon Memory Mapped Master		IRQ 0
		jtag_debug_module	Avalon Memory Mapped Slave		**0x00221000**
☑		**onchip_memory2_0**	On-Chip Memory (RAM or ROM)		
		s1	Avalon Memory Mapped Slave	**PLL_100M**	**0x00210000**
☑		**jtag_uart_0**	JTAG UART		
		avalon_jtag_slave	Avalon Memory Mapped Slave	**PLL_100M**	**0x002220a0**
☑		**sysid**	System ID Peripheral		
		control_slave	Avalon Memory Mapped Slave	**PLL_100M**	**0x002220a8**
☑		**pll_0**	PLL		
		s1	Avalon Memory Mapped Slave	**clk_0**	**0x00222000**
☑		**epcs_flash_controller_0**	EPCS Serial Flash Controller		
		epcs_control_port	Avalon Memory Mapped Slave	**PLL_100M**	**0x00221800**
☑		**pio_0**	PIO (Parallel I/O)		
		s1	Avalon Memory Mapped Slave	**PLL_100M**	**0x00222060**
☑		**Terminator_src_PEG_V1_0**	Terminator_src_PEG_V1		

Filter: Default

ALTERA SOPC Builder /Qsys


boost.bdf
chip_boost
clk_0
reset_n
adc_out_to_the_ADC_PEG_V1_0
data0_to_the_epcs_flash_controller_0
in_port_to_the_pio_0
INPUT
VCC
PI_CLK
PLL_100M
adc_in_from_the_ADC_PEG_V1_0
cs_adc_from_the_ADC_PEG_V1_0
sclk_from_the_ADC_PEG_V1_0
dac_in_from_the_DAC_PEG_V1_0
fs_dac_from_the_DAC_PEG_V1_0
sclk_from_the_DAC_PEG_V1_0
PWM_Inv_Out_from_the_PWM_PEG_V1_0
PWM_Out_from_the_PWM_PEG_V1_0
dclk_from_the_epcs_flash_controller_0
sce_from_the_epcs_flash_controller_0
sdo_from_the_epcs_flash_controller_0
out_port_from_the_pio_1[1..0]
address_to_the_cfi_flash_0[18..0]
data_to_and_from_the_cfi_flash_0[15..0]
read_n_to_the_cfi_flash_0
select_n_to_the_cfi_flash_0
write_n_to_the_cfi_flash_0
OUTPUT
BIDIR
inst

Processor (Block view)

For Xilinx - Vivado

Application Software Design

- NIOS II Eclipse IDE – NIOS II EDS
- Supports Assembly, C/C++
    - HAL/MicroC/OS II
        - HAL
            - Confirming POSIX Standards
            - Vendor - Altera
- BSP is generated from ‘.sopcinfo’ file and loads the required drivers from library

Custom instructions/Custom functions

- Control blocks are accessed through custom instructions/custom functions
  - Fixed Instruction
  - Simple and easily understandable by the user

For the PI Controller

- initialize PI module with Kp and Ki Registers

```
hw_init_pi(PI_PEG_V1_0_BASE, Kp_val ,Ki_val);
```

- enable/disable PI Module

```
hw_enable_pi(PI_PEG_V1_0_BASE,PI_Cntrl_Reg);
```

- input to the PI module

```
hw_pi_in(PI_PEG_V1_0_BASE,error);
```

- out from the PI module

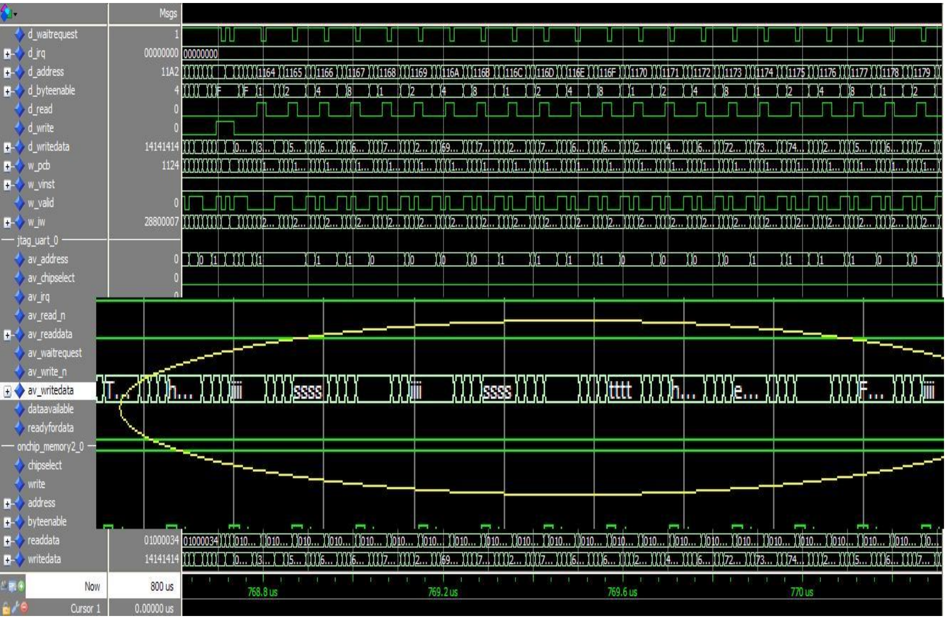
```
pwm_input= hw_pi_read_output(PI_PEG_V1_0_BASE);
```

Custom instructions/Custom functions

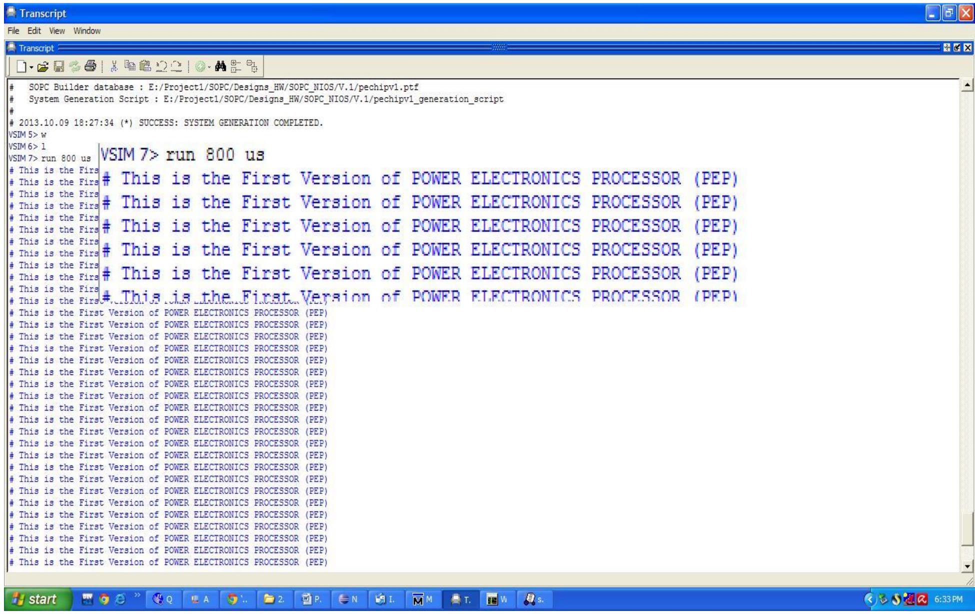
- Good Readability
- Number of lines are less
- Faster execution

SOPC Simulation

- Using ModelSim
  - Can verify SOPC Hardware and application software together - *Hardware Software – Co verification*



Hardware Signal Flow



```
#   SOPC Builder database : E:/Project1/SOPC/Designs_HW/SOPC_NIOS/V.1/pechipv1.ptf
#   System Generation Script : E:/Project1/SOPC/Designs_HW/SOPC_NIOS/V.1/pechipv1_generation_script
#
# 2013.10.09 18:27:34 (*) SUCCESS: SYSTEM GENERATION COMPLETED.
VSIM 5> w
VSIM 6> 1
VSIM 7> run 800 us
# This is the Firs
# This is the Firs
# This is the Firs
# This is the Firs
# This is the Firs
# This is the Firs
# This is the Firs
# This is the Firs
# This is the Firs
# This is the Firs
# This is the Firs
# This is the Firs
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
```

```
VSIM 7> run 800 us
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
# This is the First Version of POWER ELECTRONICS PROCESSOR (PEP)
```

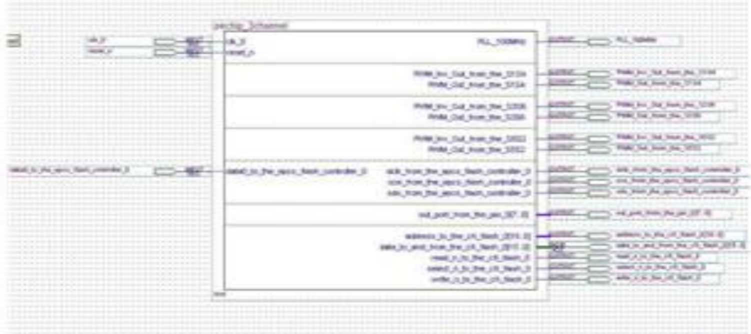
JTAG terminal

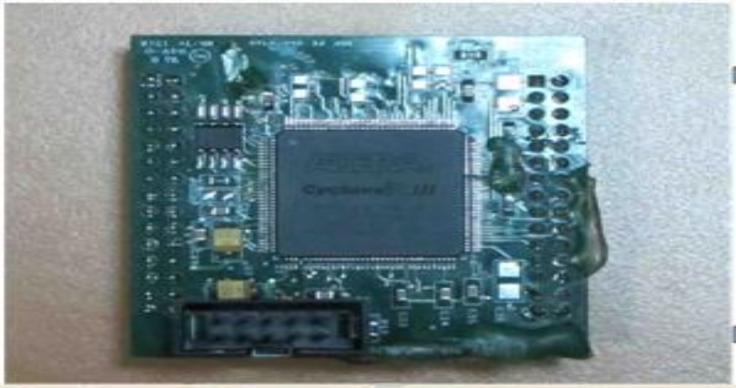
Technology Deliverables

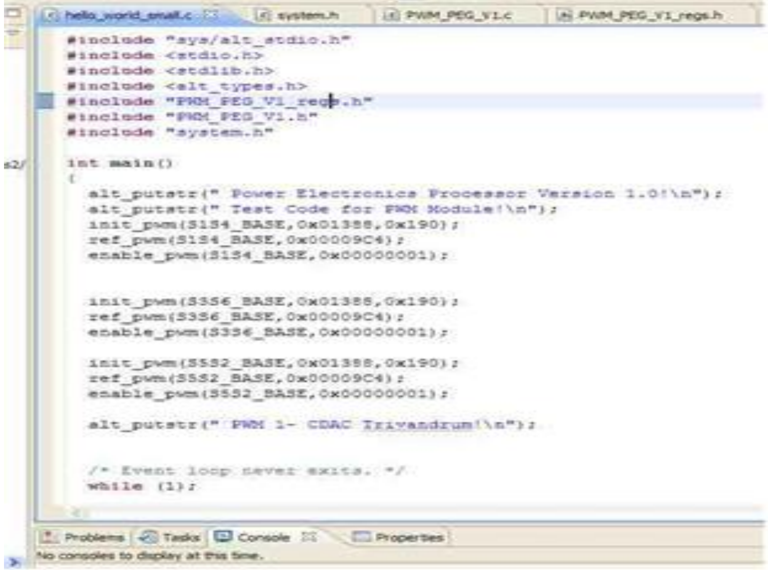
SOPC PE Controller

- Soft processor Integrated **controller design procedure** and user manual
- **HDL control library elements** for hardware accelerators - (PI controller, Filters, PLL,PWM, Phase Transformations etc)
  - ***19 PE Specific IPs***
- **Application evaluation report using** SOPC standard (*v1.0*) controller
  - Voltage Mode control of DC-DC Boost converter
  - AC drive –VVVF AC Induction Motor – Single core and Multi core
  - FOC of Induction Motor (As HiL in FSS miniature)

Development Activities

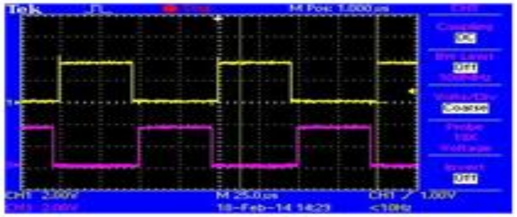


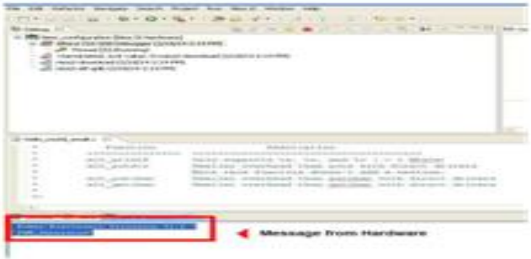




```c
#include "sys/alt_stdio.h"
#include <stdio.h>
#include <stdlib.h>
#include <alt_types.h>
#include "PWM_PEG_V1_regs.h"
#include "PWM_PEG_V1.h"
#include "system.h"

int main()
{
  alt_putstr(" Power Electronics Processor Version 1.0!\n");
  alt_putstr(" Test Code for PWM Module!\n");
  init_pwm(S1S4_BASE,0x01388,0x190);
  ref_pwm(S1S4_BASE,0x000009C4);
  enable_pwm(S1S4_BASE,0x00000001);


  init_pwm(S3S6_BASE,0x01388,0x190);
  ref_pwm(S3S6_BASE,0x000009C4);
  enable_pwm(S3S6_BASE,0x00000001);

  init_pwm(S5S2_BASE,0x01388,0x190);
  ref_pwm(S5S2_BASE,0x000009C4);
  enable_pwm(S5S2_BASE,0x00000001);

  alt_putstr(" PWM 1- CDAC Trivandrum!\n");


  /* Event loop never exits. */
  while (1);
```

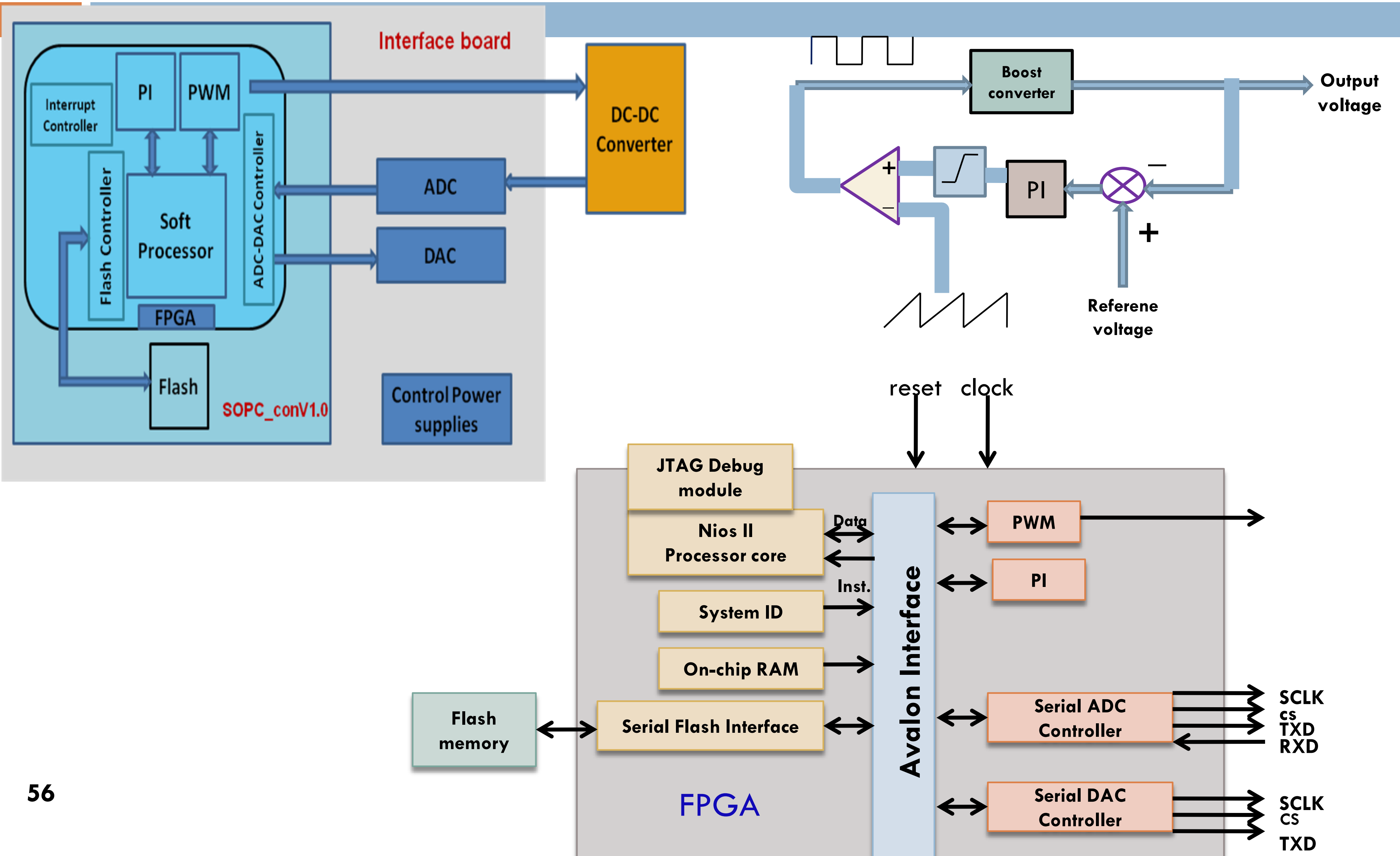
No consoles to display at this time.



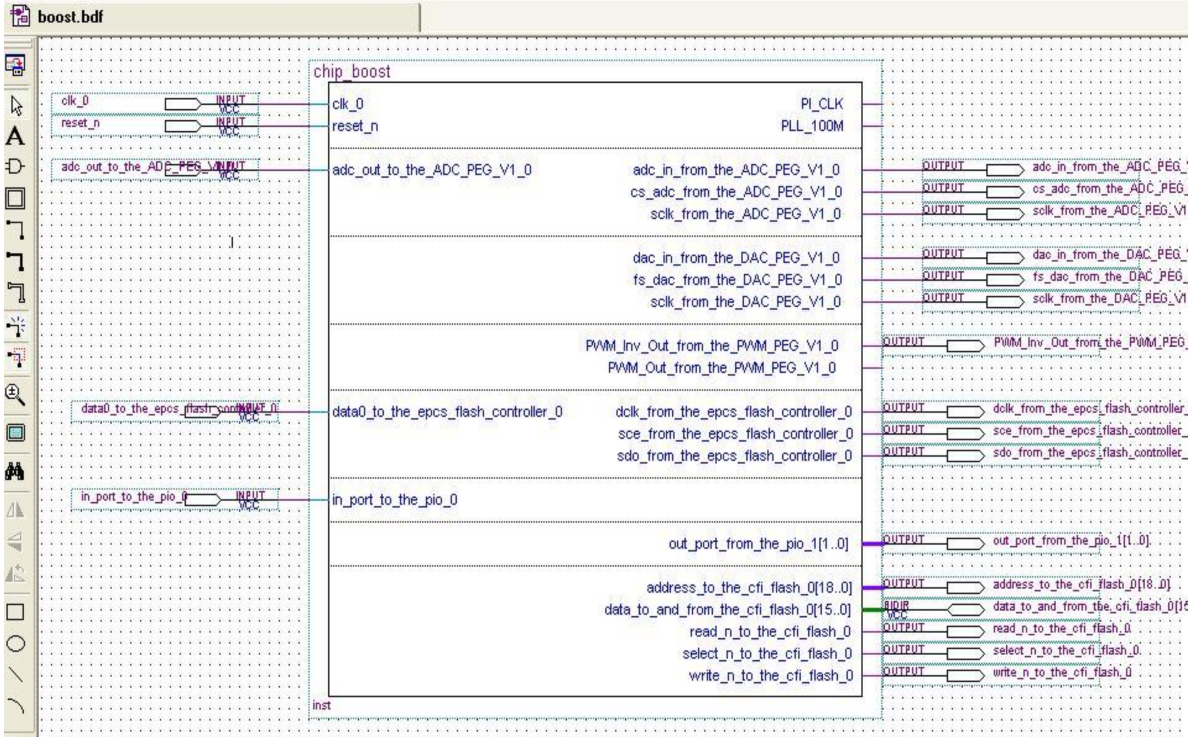


Message from Hardware

DC-DC converter control using SOPC controller



SOPC Configured for Dc-Dc Control


boost.bdf
chip_boost
clk_0
reset_n
adc_out_to_the_ADC_PEG_V1_0
data0_to_the_epcs_flash_controller_0
in_port_to_the_pio_0
INPUT
VCC
PI_CLK
PLL_100M
adc_in_from_the_ADC_PEG_V1_0
cs_adc_from_the_ADC_PEG_V1_0
sclk_from_the_ADC_PEG_V1_0
dac_in_from_the_DAC_PEG_V1_0
fs_dac_from_the_DAC_PEG_V1_0
sclk_from_the_DAC_PEG_V1_0
PWM_Inv_Out_from_the_PWM_PEG_V1_0
PWM_Out_from_the_PWM_PEG_V1_0
dclk_from_the_epcs_flash_controller_0
sce_from_the_epcs_flash_controller_0
sdo_from_the_epcs_flash_controller_0
out_port_from_the_pio_1[1..0]
address_to_the_cfi_flash_0[18..0]
data_to_and_from_the_cfi_flash_0[15..0]
read_n_to_the_cfi_flash_0
select_n_to_the_cfi_flash_0
write_n_to_the_cfi_flash_0
OUTPUT
BIDIR
inst

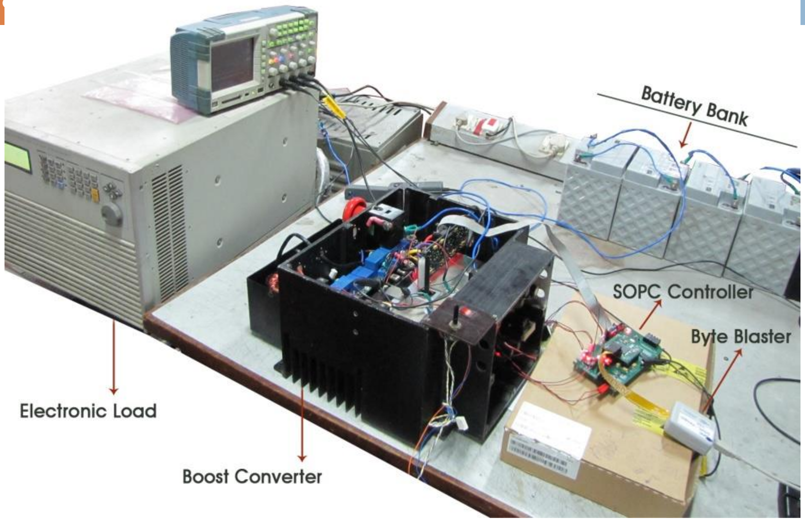
❖**PI controller is a Hardware module to the CPU**

❖**Accessed by custom functions**

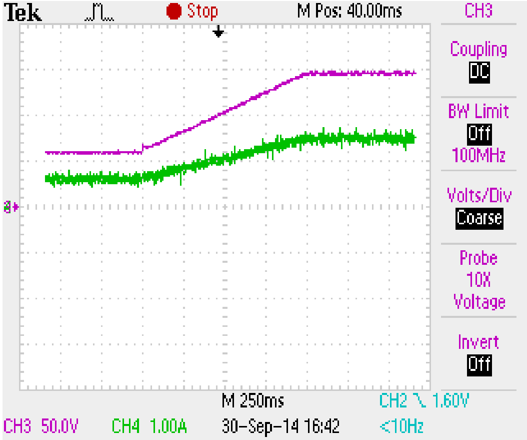
eg:- pi_in(PI_PEG_V1_0_BASE,error);

pwm_input = pi_read_output(PI_PEG_V1_0_BASE);

DC-DC Converter control



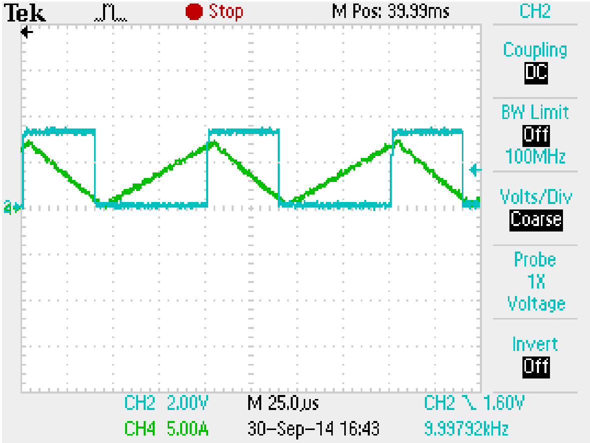
Test setup of DC-DC Converter



Output voltage(pink) and load current(green) during softstart

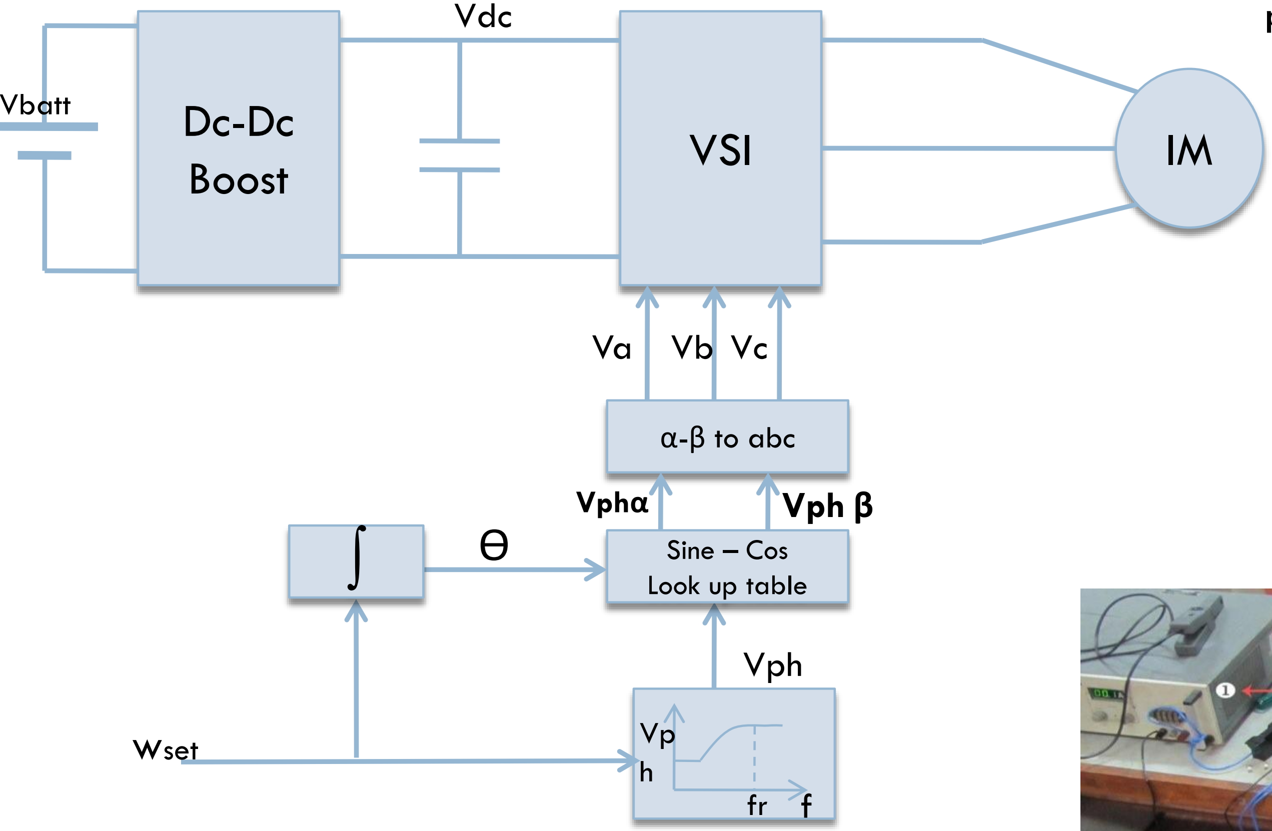
Bench Marking

Application	TMS320F2812	SOPC
Boost Converter	6 µs(CPU clock-150 MHz)	800 ns(CPU clock-100 MHz)



Inductor current (green) and PWM Pulse (blue) in steady state

AC Drive control



Vbatt

Dc-Dc Boost

Vdc

VSI

IM

Va Vb Vc

α-β to abc

Vphα

Vph β

∫

Ɵ

Sine – Cos Look up table

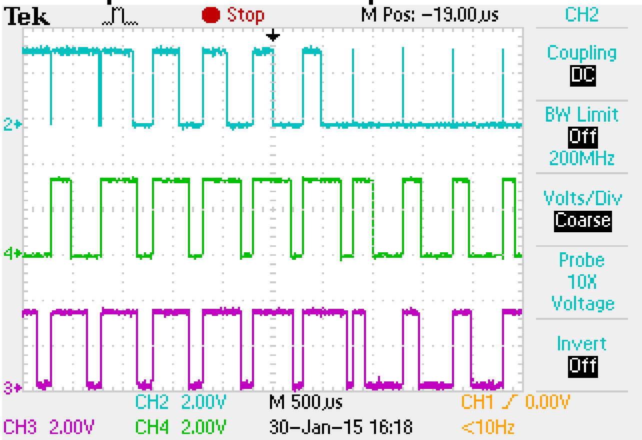
Vph

Wset

Vp h

fr f

pwms top switches of the 3phase inverter



Tek Stop M Pos: –19.00µs

CH2

Coupling DC

BW Limit Off 200MHz

Volts/Div Coarse

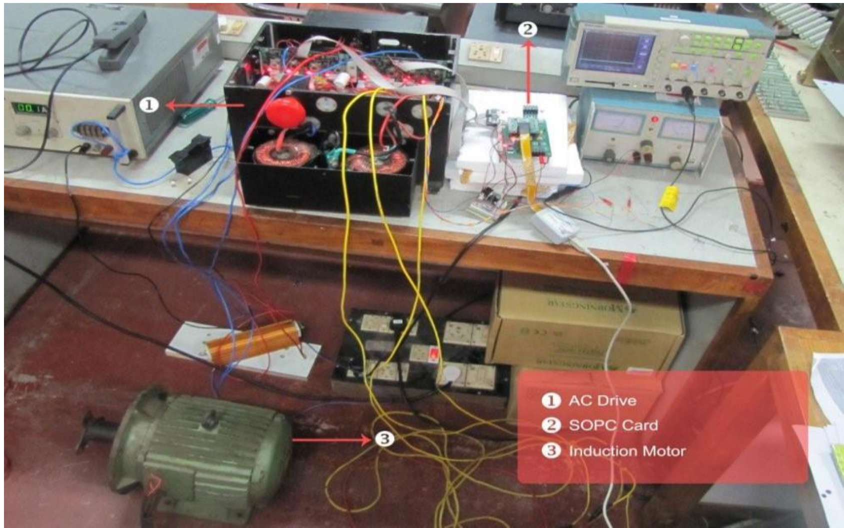
Probe 10X Voltage

Invert Off

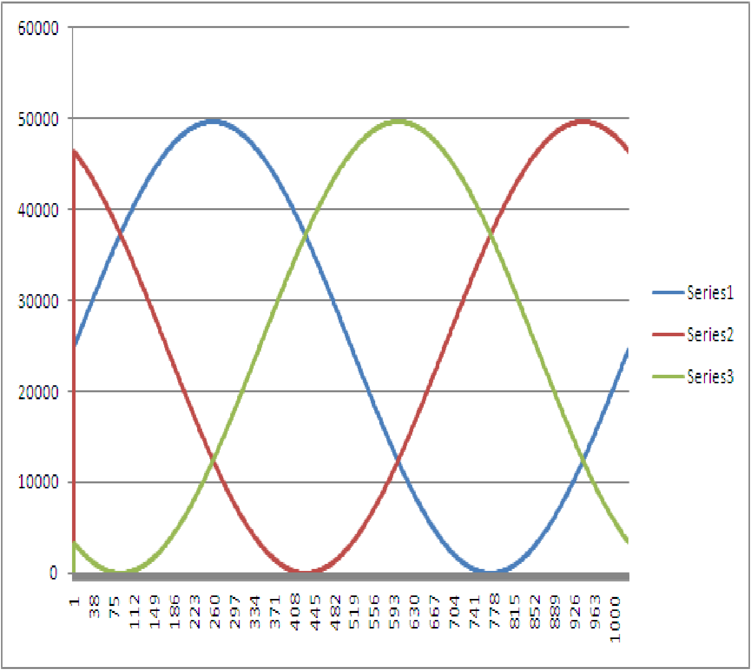
CH2 2.00V M 500µs CH1 ╱ 0.00V

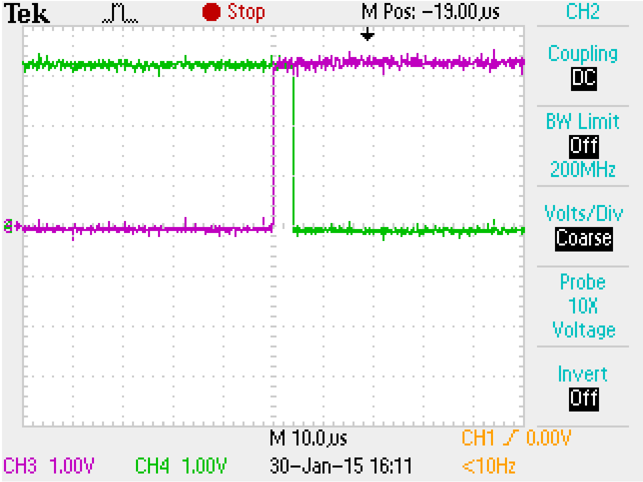
CH3 2.00V CH4 2.00V 30–Jan–15 16:18 <10Hz

AC Drive assembled in CDAC



AC Drive control





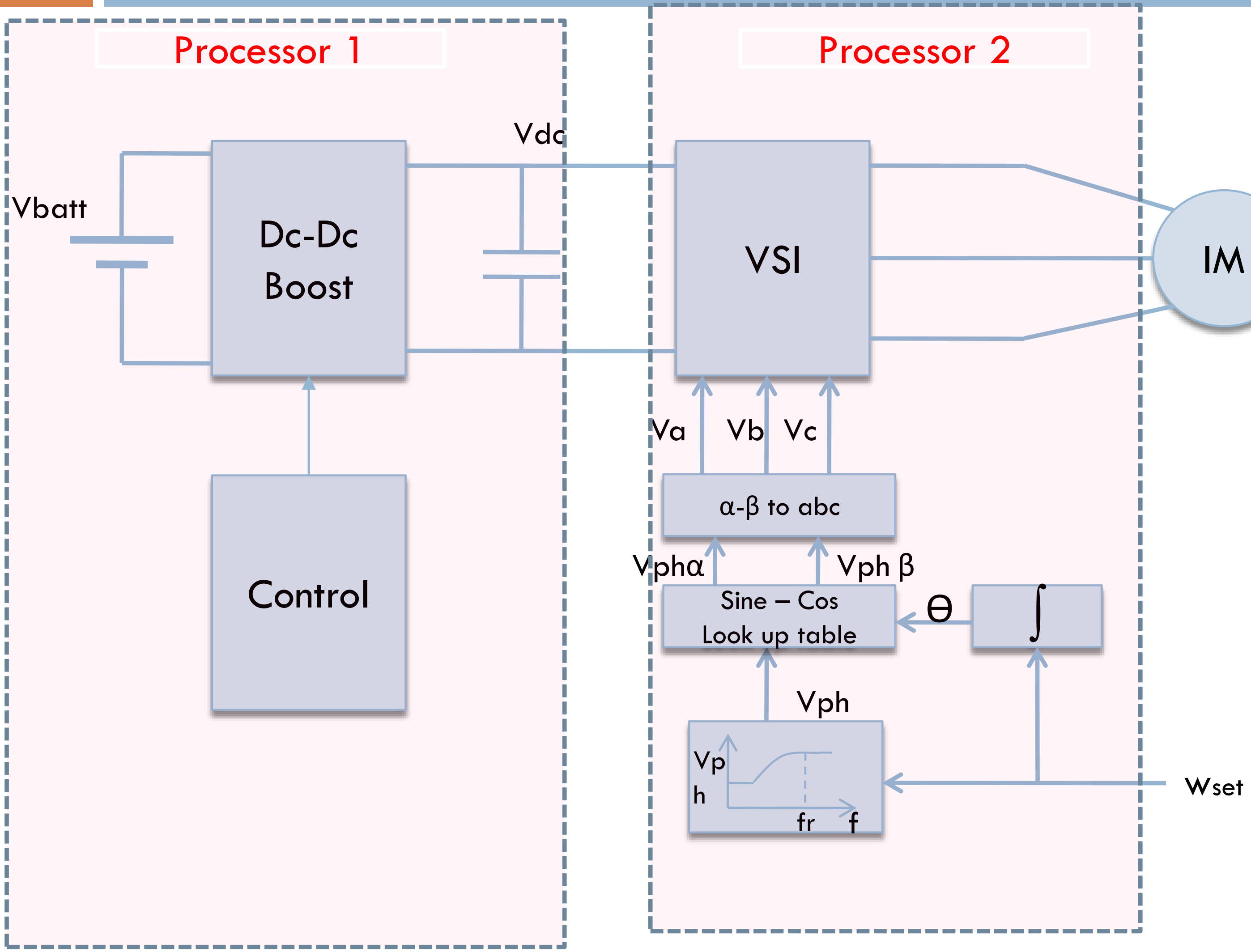
pwm showing dead time of 4us('a' phase PWMs)

Plot of Va, Vb and Vc values from NiosII through alpha_beta_abc IP

α-β to abc, Integrator are in Hardware
The C code implementation of α-β to abc, took - 600ns
alpha_beta _to_abc IP took only 300ns to perform the conversion

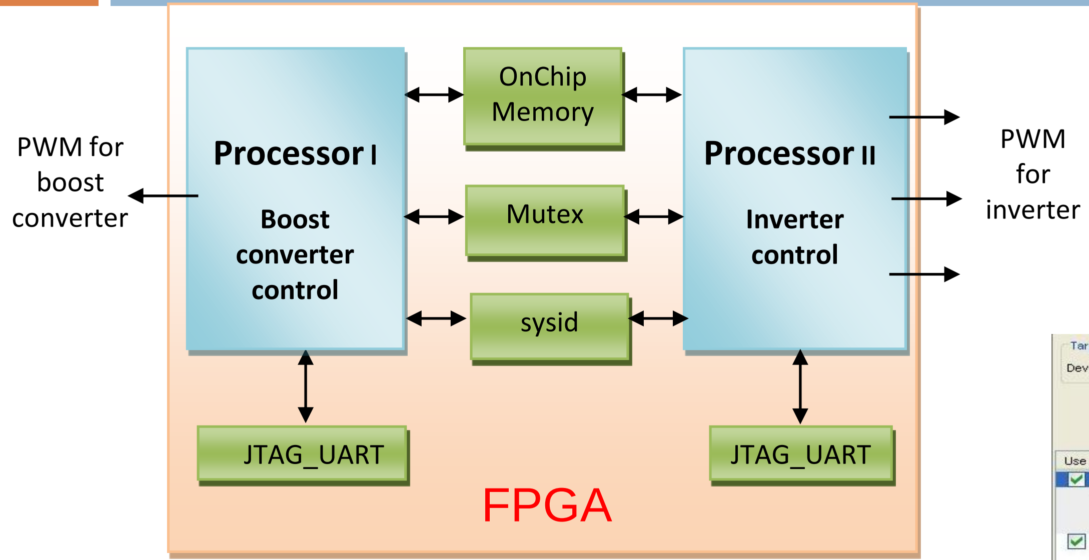
Main loop execution time = 2 us

AC Drive application in Multi-core



Boost converter control in one CPU Core and Inverter control in other core

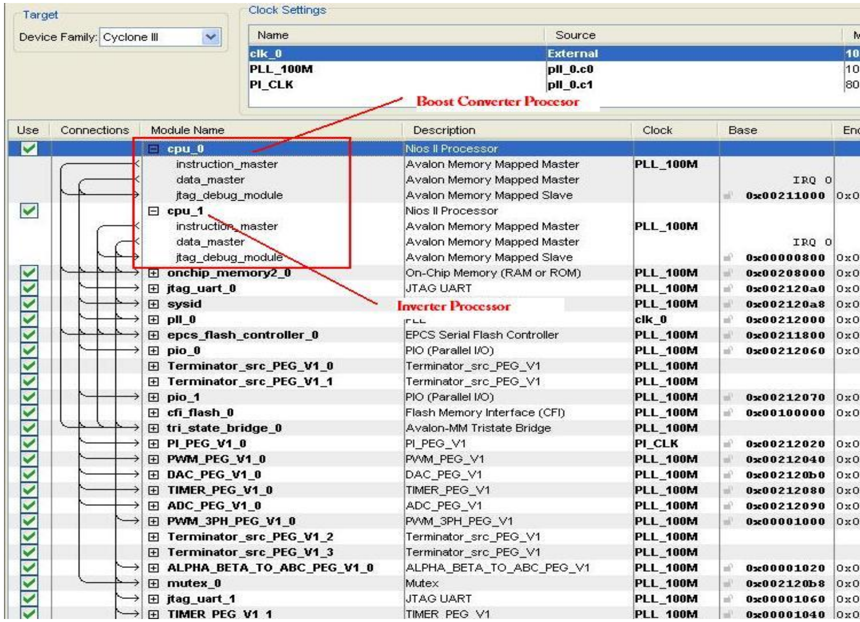
AC Drive application in Multi-core



Processor configuration in Multi core

Main loop execution time is 1.5µs

SOPC View in Multi core



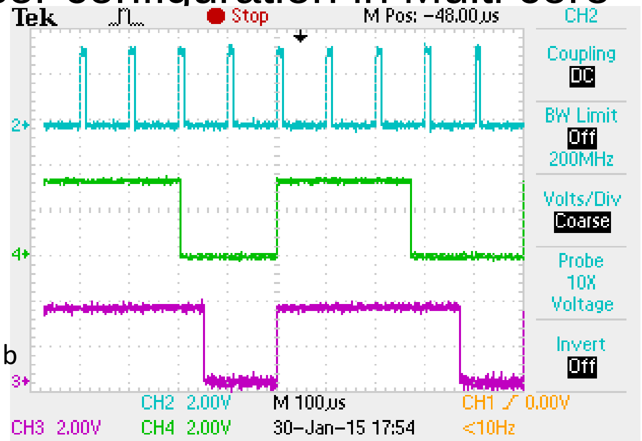
Device Family: Cyclone III

Name	Source	M
clk_0	External	10
PLL_100M	pll_0.c0	10
PI_CLK	pll_0.c1	80

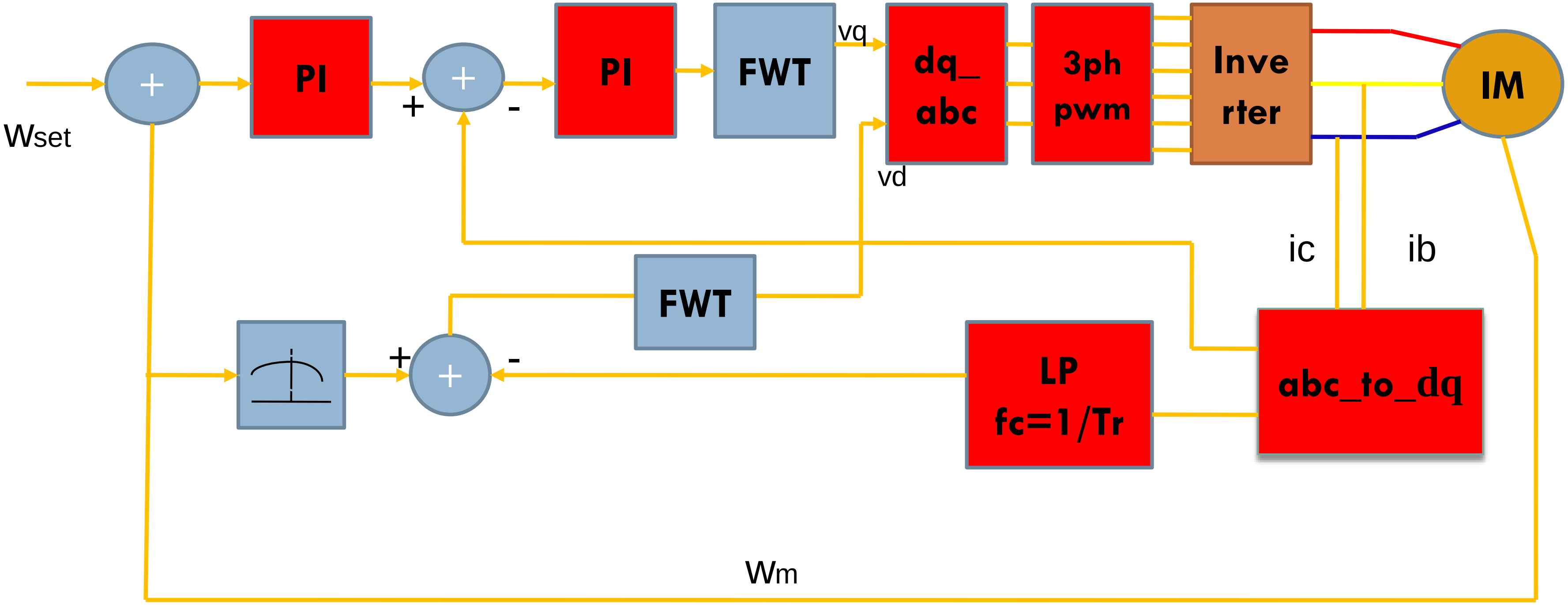
Use	Connections	Module Name	Description	Clock	Base	End
☑		⊟ cpu_0	Nios II Processor			
		instruction_master	Avalon Memory Mapped Master	PLL_100M		
		data_master	Avalon Memory Mapped Master		IRQ 0	
		jtag_debug_module	Avalon Memory Mapped Slave		0x00211000	0x0
☑		⊟ cpu_1	Nios II Processor			
		instruction_master	Avalon Memory Mapped Master	PLL_100M		
		data_master	Avalon Memory Mapped Master		IRQ 0	
		jtag_debug_module	Avalon Memory Mapped Slave		0x00000800	0x0
☑		⊞ onchip_memory2_0	On-Chip Memory (RAM or ROM)	PLL_100M	0x00208000	0x0
☑		⊞ jtag_uart_0	JTAG UART	PLL_100M	0x002120a0	0x0
☑		⊞ sysid		PLL_100M	0x002120a8	0x0
☑		⊞ pll_0	PLL	clk_0	0x00212000	0x0
☑		⊞ epcs_flash_controller_0	EPCS Serial Flash Controller	PLL_100M	0x00211800	0x0
☑		⊞ pio_0	PIO (Parallel I/O)	PLL_100M	0x00212060	0x0
☑		⊞ Terminator_src_PEG_V1_0	Terminator_src_PEG_V1	PLL_100M		
☑		⊞ Terminator_src_PEG_V1_1	Terminator_src_PEG_V1	PLL_100M		
☑		⊞ pio_1	PIO (Parallel I/O)	PLL_100M	0x00212070	0x0
☑		⊞ cfi_flash_0	Flash Memory Interface (CFI)	PLL_100M	0x00100000	0x0
☑		⊞ tri_state_bridge_0	Avalon-MM Tristate Bridge	PLL_100M		
☑		⊞ PI_PEG_V1_0	PI_PEG_V1	PI_CLK	0x00212020	0x0
☑		⊞ PWM_PEG_V1_0	PWM_PEG_V1	PLL_100M	0x00212040	0x0
☑		⊞ DAC_PEG_V1_0	DAC_PEG_V1	PLL_100M	0x002120b0	0x0
☑		⊞ TIMER_PEG_V1_0	TIMER_PEG_V1	PLL_100M	0x00212080	0x0
☑		⊞ ADC_PEG_V1_0	ADC_PEG_V1	PLL_100M	0x00212090	0x0
☑		⊞ PWM_3PH_PEG_V1_0	PWM_3PH_PEG_V1	PLL_100M	0x00001000	0x0
☑		⊞ Terminator_src_PEG_V1_2	Terminator_src_PEG_V1	PLL_100M		
☑		⊞ Terminator_src_PEG_V1_3	Terminator_src_PEG_V1	PLL_100M		
☑		⊞ ALPHA_BETA_TO_ABC_PEG_V1_0	ALPHA_BETA_TO_ABC_PEG_V1	PLL_100M	0x00001020	0x0
☑		⊞ mutex_0	Mutex	PLL_100M	0x002120b8	0x0
☑		⊞ jtag_uart_1	JTAG UART	PLL_100M	0x00001060	0x0
☑		⊞ TIMER_PEG_V1_1	TIMER_PEG_V1	PLL_100M	0x00001040	0x0

PWM for Boost Converter from Processor 1

PWM for Inverter from Processor 2 (a phase and b phase - Top switches)



FOC on SOPC



Red blocks – are hardware IPs

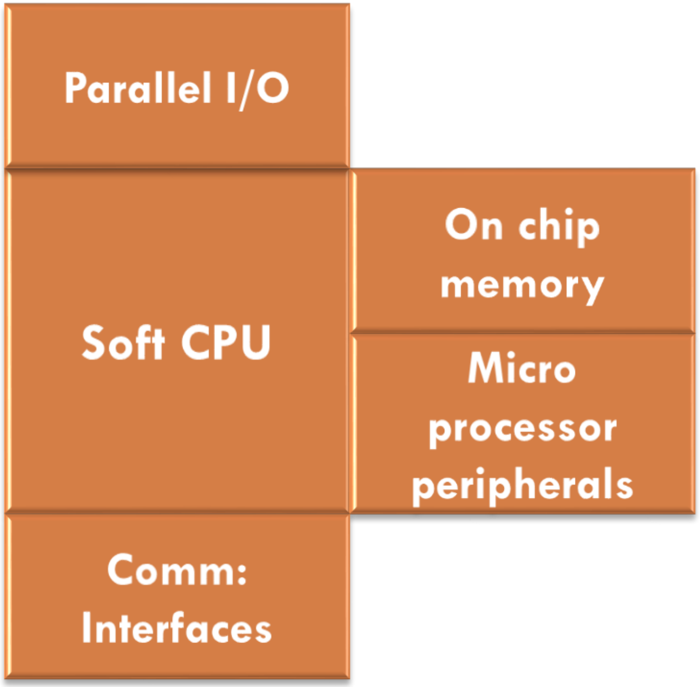
Main loop execution = 10 us in 100MHz
For TMS 320F2814 = 22 us in 150MHz

Design flow

Soft CPU

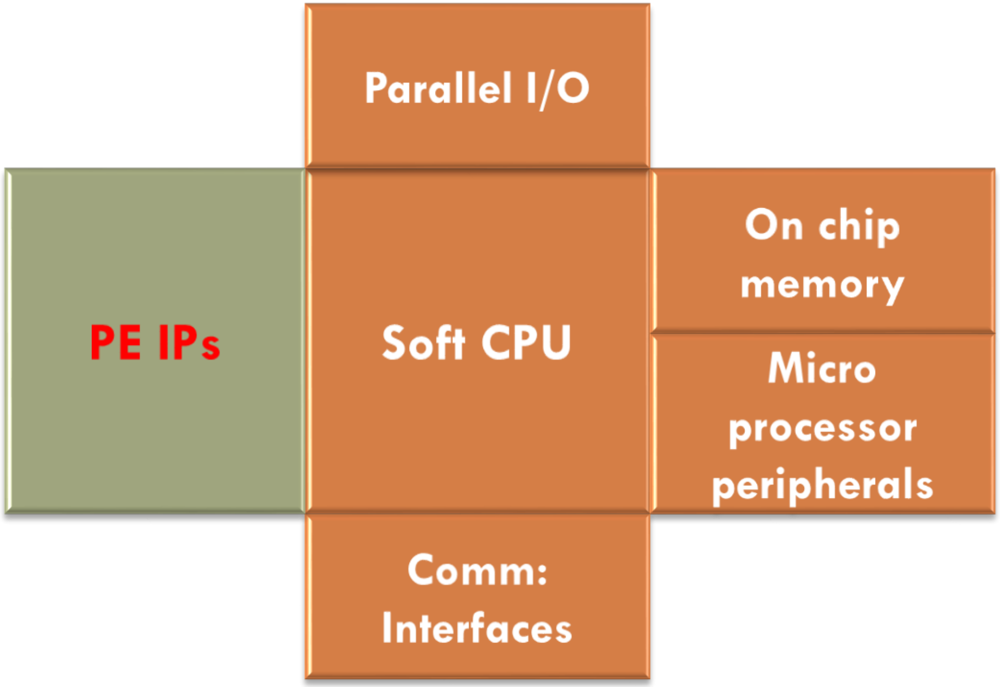
Off the Shelf CPU Cores -NIOS II, Mico Blaze,Leon3 etc.

Design flow



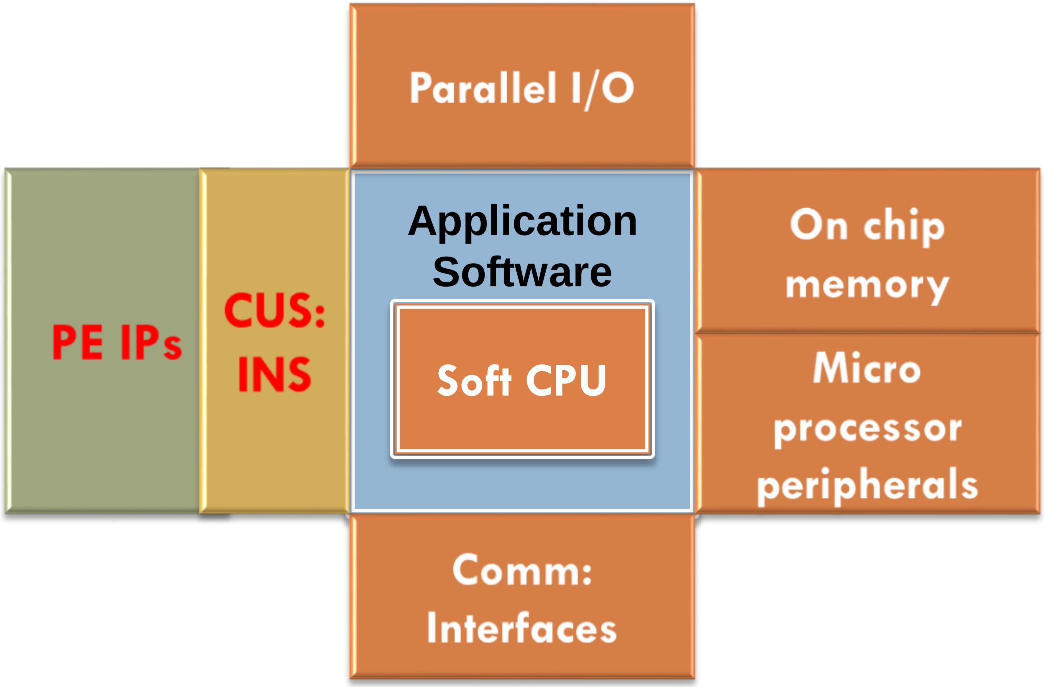
Micro processor peripherals :- JTAG, Hardware Multiplier, memory interfaces, sys_id etc.
Communication Interfaces :-   CAN,USB etc.

Design flow



PE IPs :PWM ,PI,abc_to_dq,dq_abc, ADC/DAC controllers,Integrators etc.

Design flow



For Multiplication - MUL A,B Similarly For PI Controller IP
hw_pi_in(PI_ID,error);
out = hw_pi_read(PI_ID);

Number of lines in the application code is less

Soft IP core: Design Advantages

- Higher level of design reuse
- Re-configurability
- Reduced obsolescence risk
- Simplified design update or change

  Lesser over head in control software design

- Parallel Processing
- Improved Time to market
- Single to multi-core flexibility

THANK YOU

ajeesh@cdac.in