Uso del Lenguaje VHDL-AMS en la Modelacion, Simulation, e implementacion de SoC para Aplicaciones Mecatronicas

Dr. Juan R. Pimentel

Professor of Computer Engineering

Kettering University

Flint, Michigan, USA

Presentation Outline

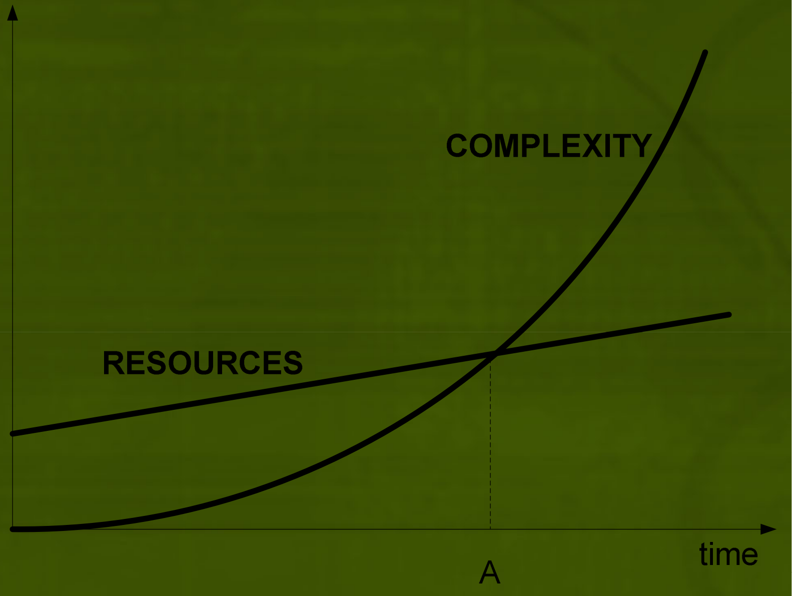
- Introduction and motivation
- Example of Complexity: Powertrain of HEVs
- Proposed Approach
- The VHDL-AMS language
- Case study
  - FPGA implementation of an AC motor controller
  - Overall architecture
  - Modeling and simulation
- Conclusions

Introduction

Despite great advances in:

- SLD (System Level Design)
- SoC (System on Chip) and
- EDA (Electronic Design Automation:

Designing complex systems with high productivity gains is still a challenge.



High level of expertise is required

Main problems:
- COMPLEXITY
- Management of complexity

Introduction: Automotive Complexity

There is a need for alternative automotive energy sources

- Alternative fuel
- Hybrid-Electric Vehicles (HEVs) and EVs
- Fuel-Cell based EVs

Hybrid-Electric vehicles (HEV) and EVs are practical today

Electronics and Software within HEVs are extremely complex

Main Issue: Cost and superior capabilities of all components including battery technology

Modeling and Simulation play an increasingly important role in the analysis and design of these types of systems

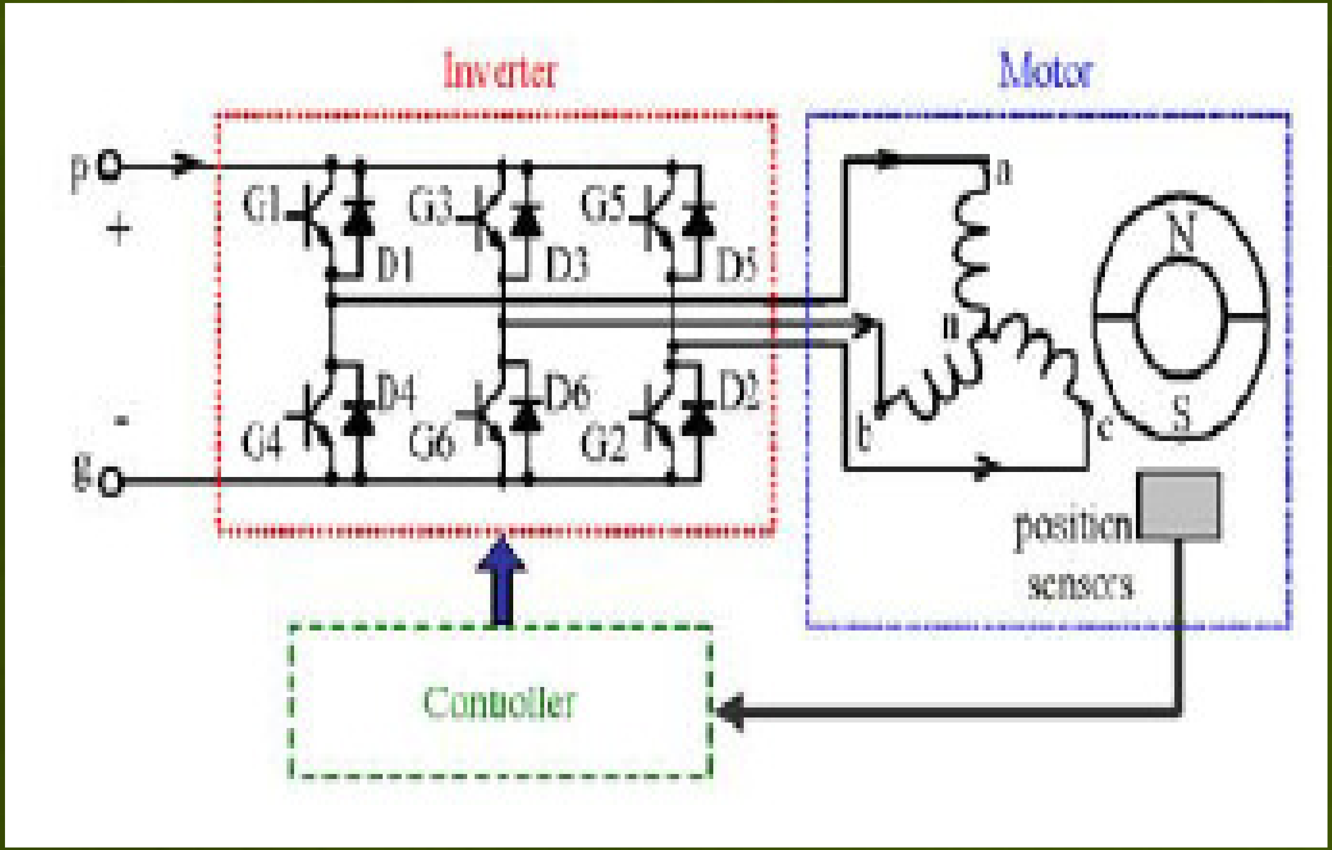
Introduction: Complexity

Kettering University has 2 Ford Escape Hybrid-Electric Vehicles

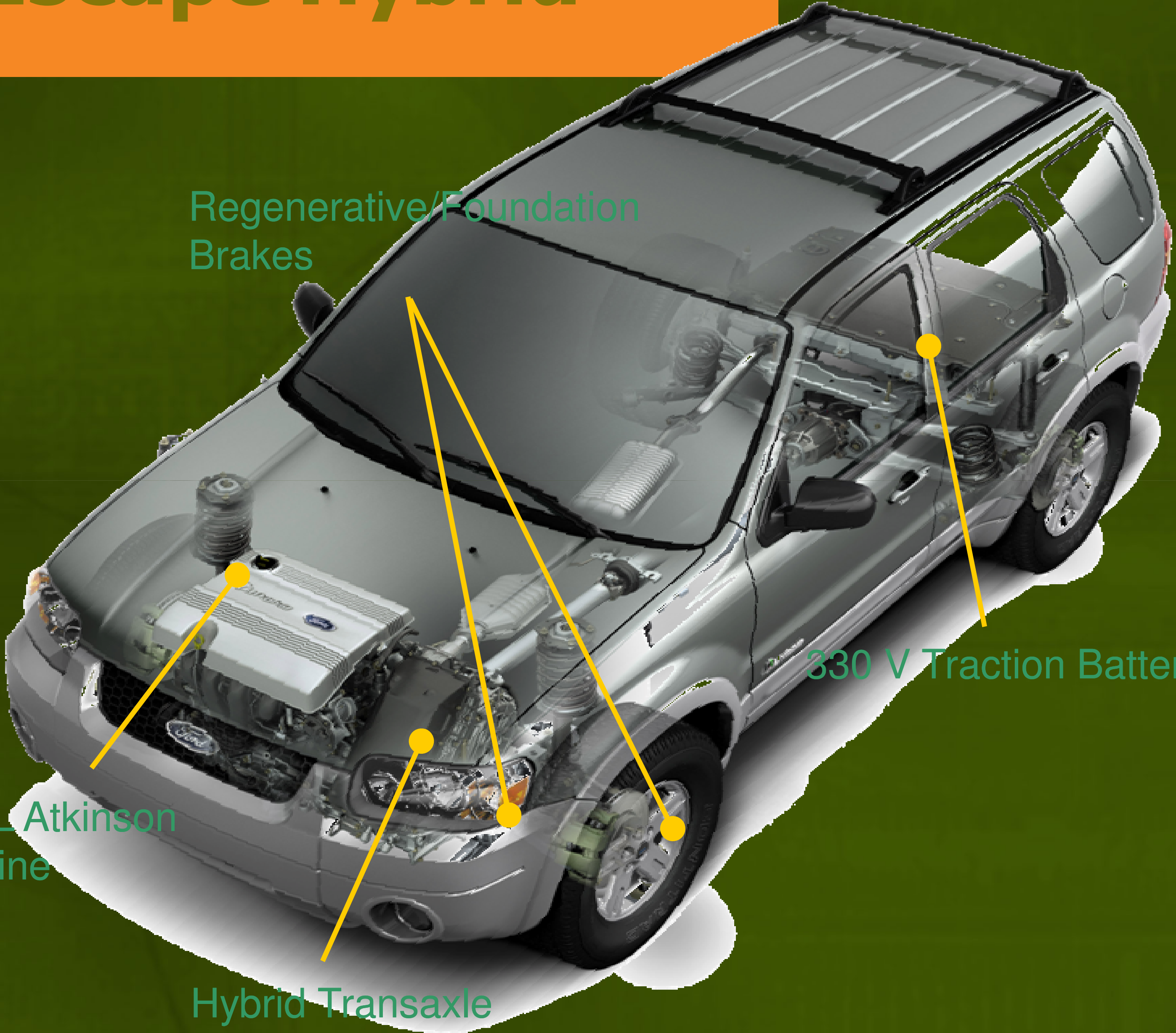
- One was totally diss-assembled
- The second is being used to test new designs

Introduction: Complexity

Mentor Graphics - Kettering University Project Focus of VHDL-AMS Modeling and Simulation

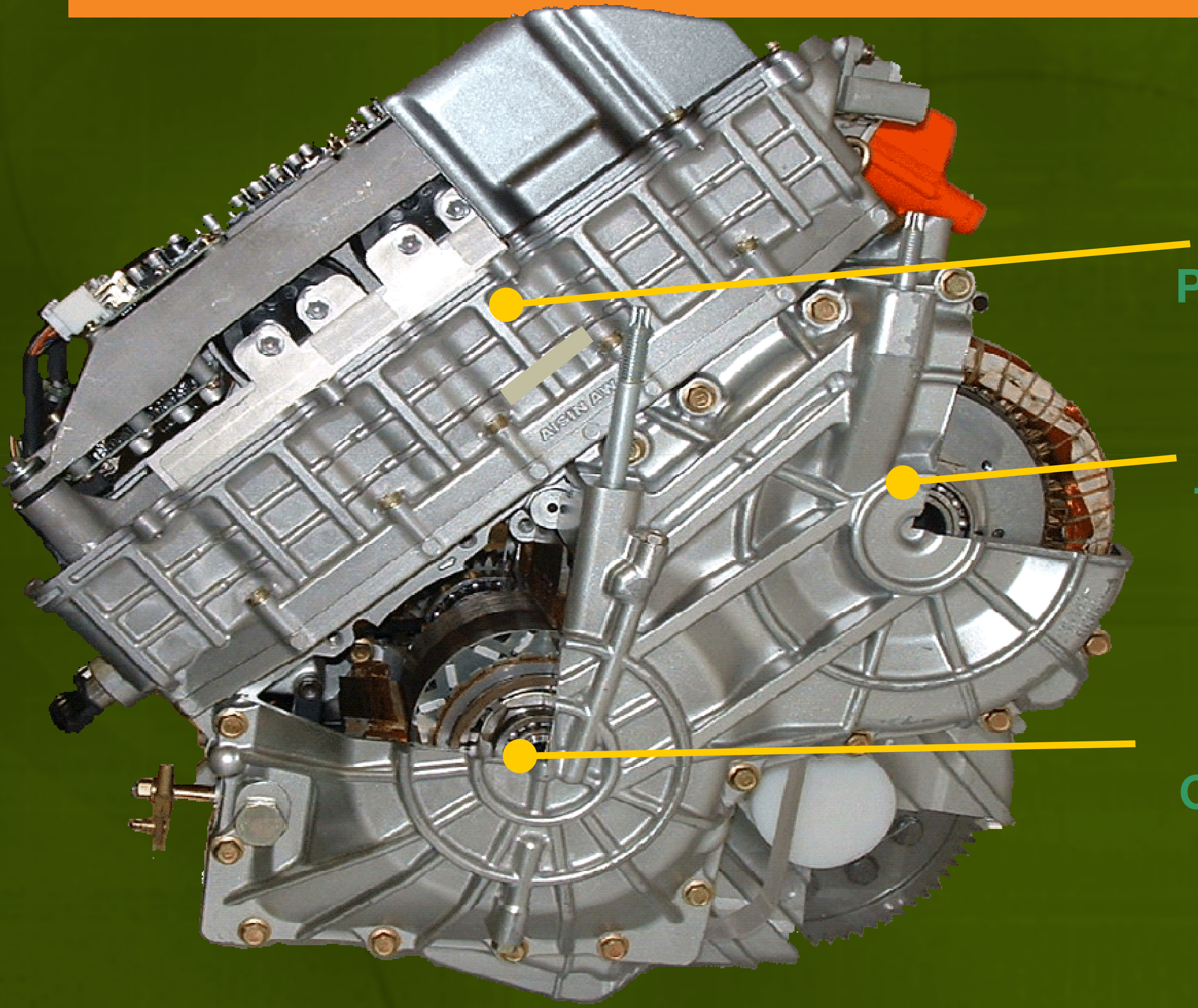


Ford Escape Hybrid

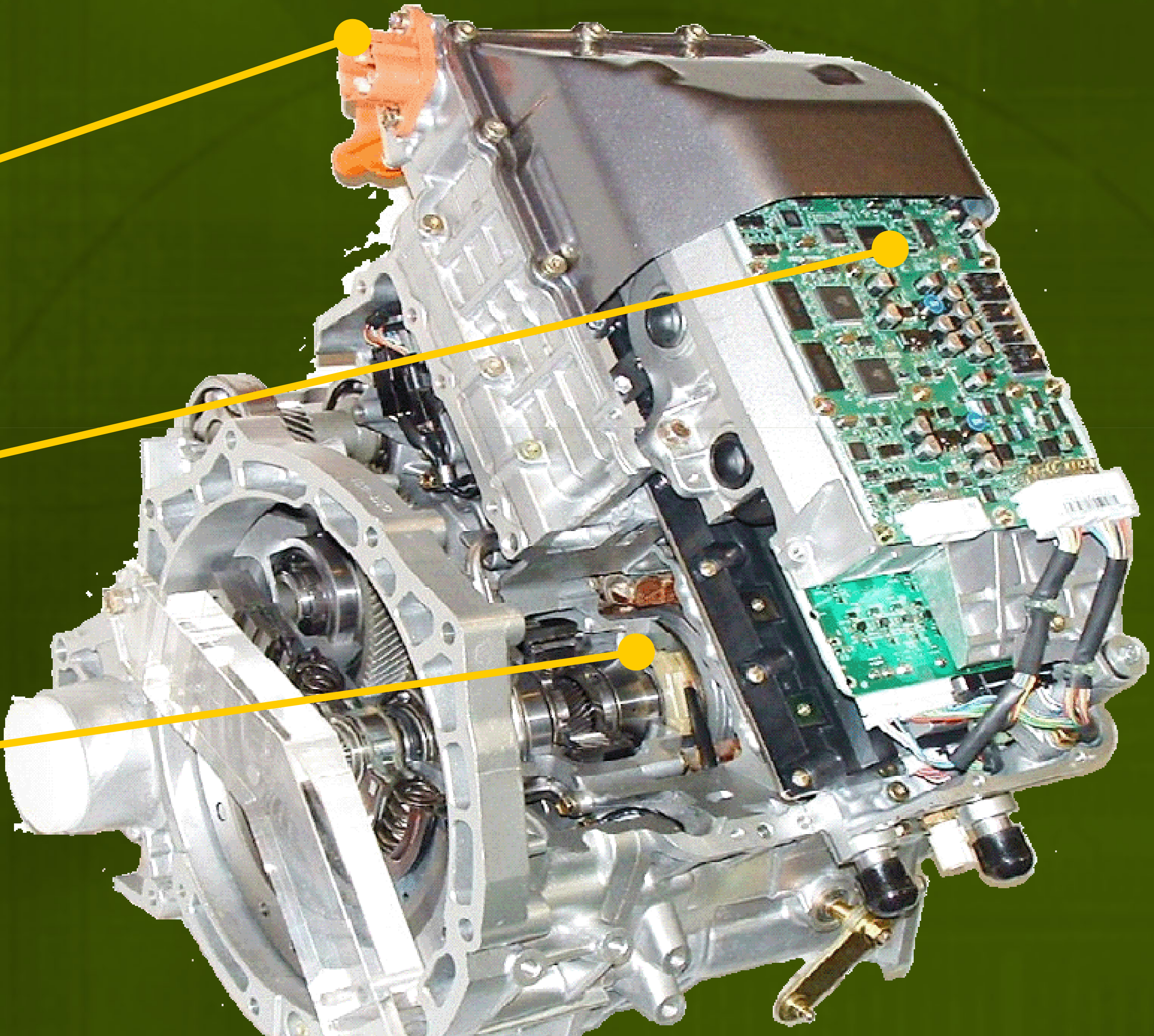


Ford Escape Hybrid

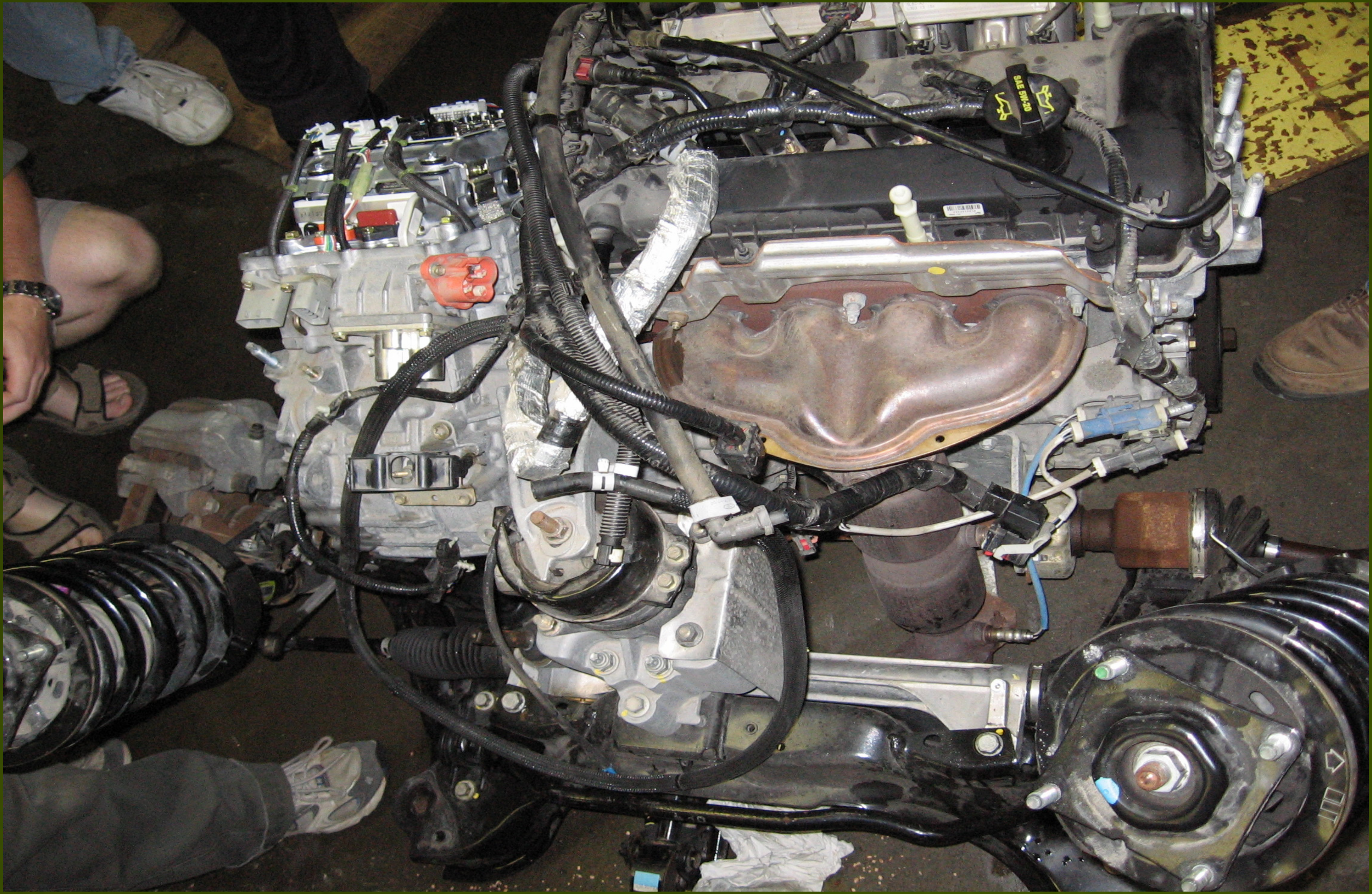
Escape Hybrid Transaxle



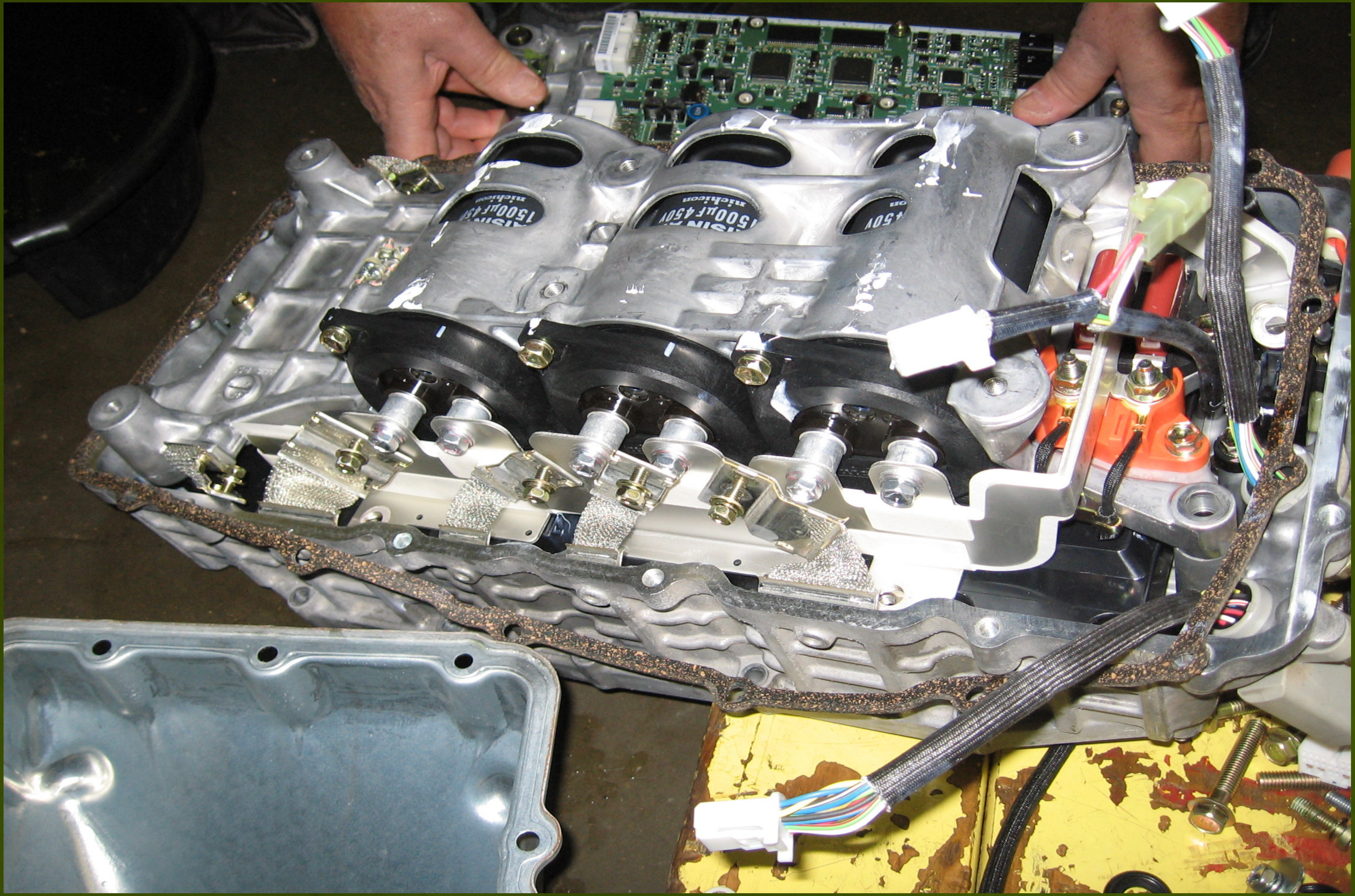
Escape Hybrid Transaxle Cut-Away



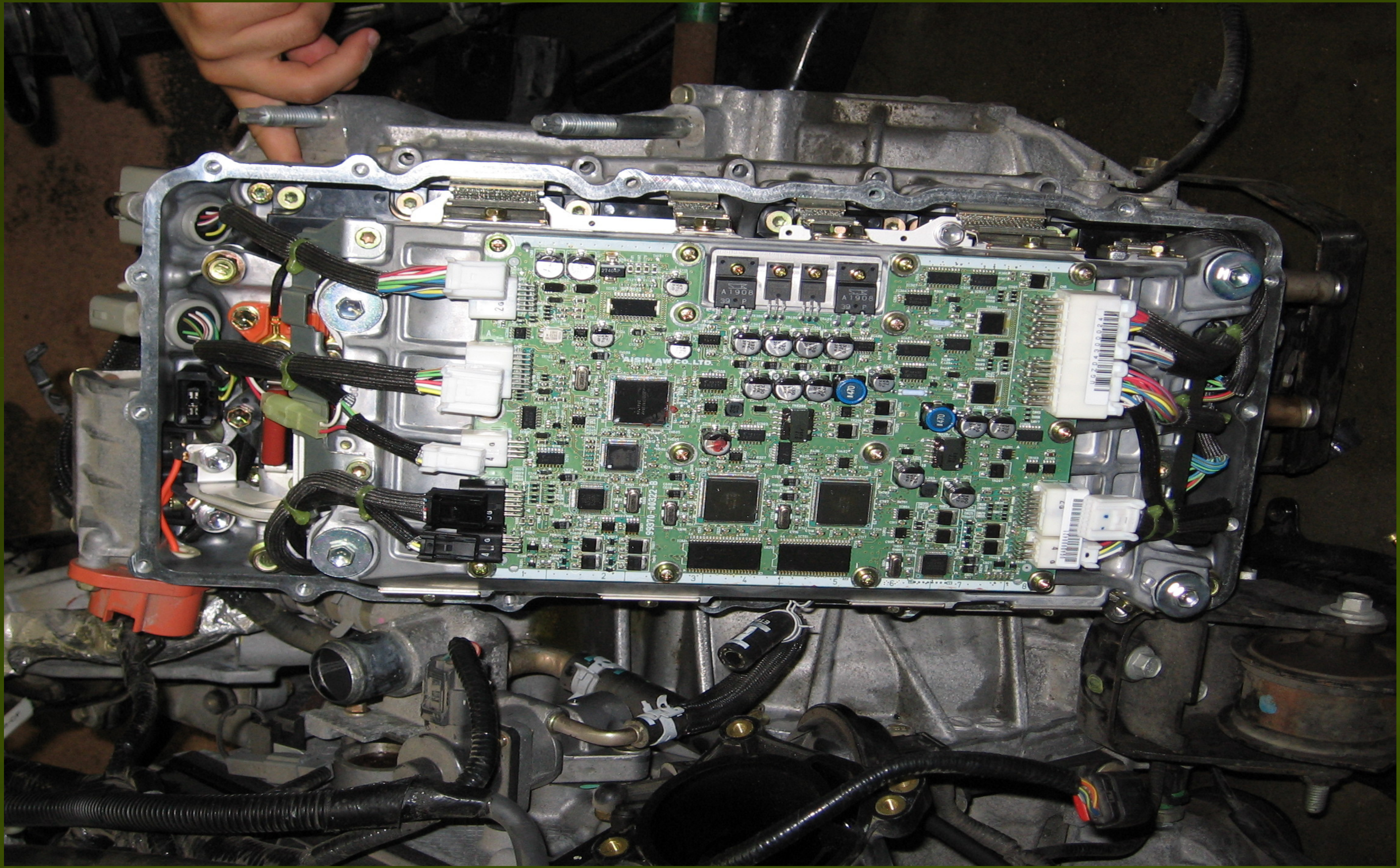
Escape CVT & Power Electronics



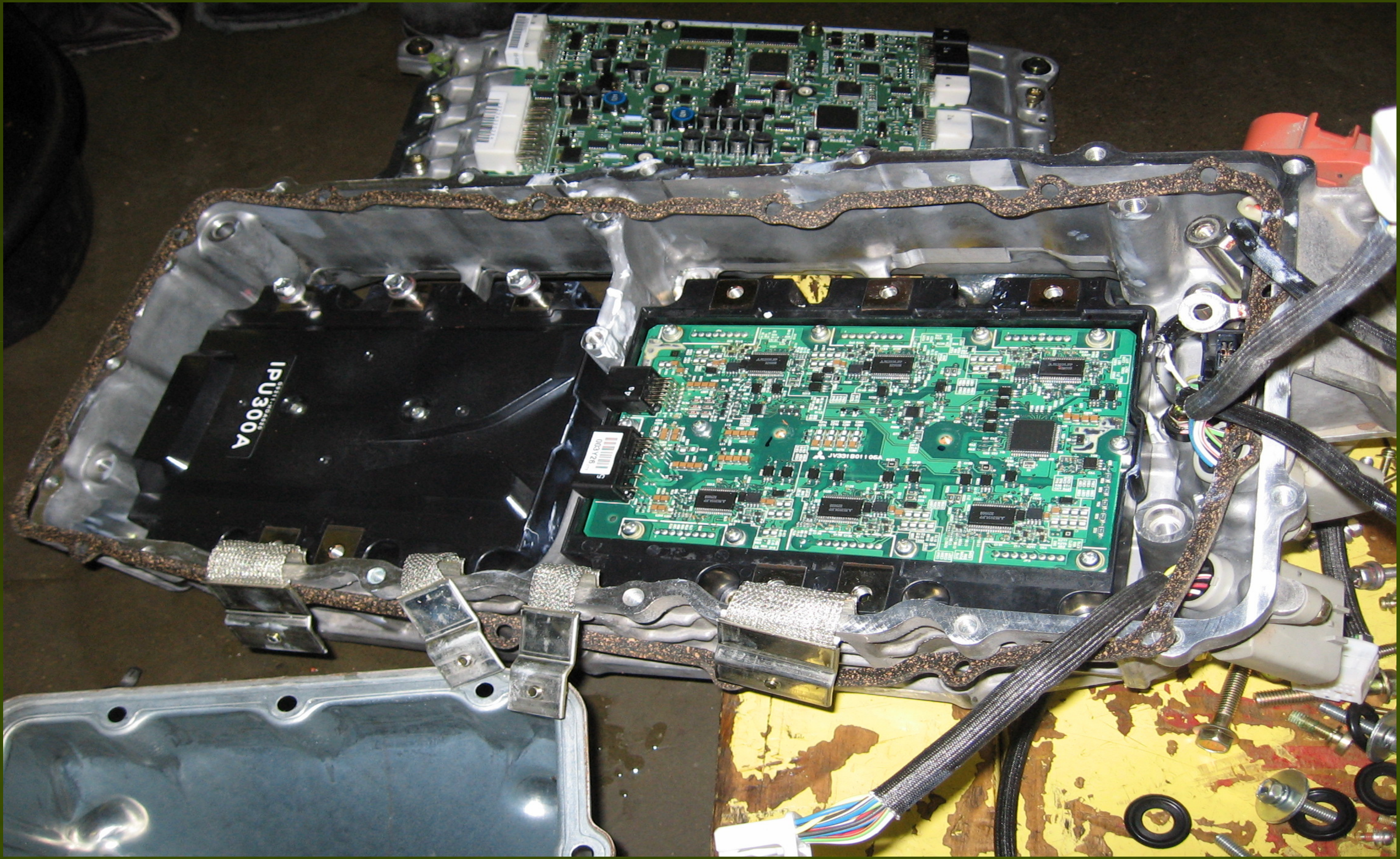
Capacitors Across Escape Battery



Dual Inverters and Control Board



Inverter Control Board: Ford Escape



Introduction

From time to time, certain paradigms emerge or are used to reduce complexity

MAIN PARADIGMS

Language based:

C, SystemC, VHDL, VHDL-AMS, Verilog

Tool based:

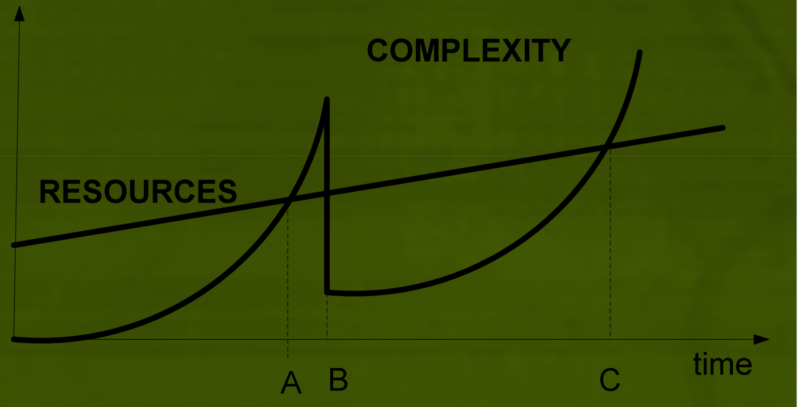
Malab/Simulink, SystemVision, EDA (capture, analysis, synthesis, test, simulation, etc.)

Methodology or Process based:

Top-down, V-cycle, AUTOSAR

Standards based:

UML, VHDL-AMS



Proposed Approach

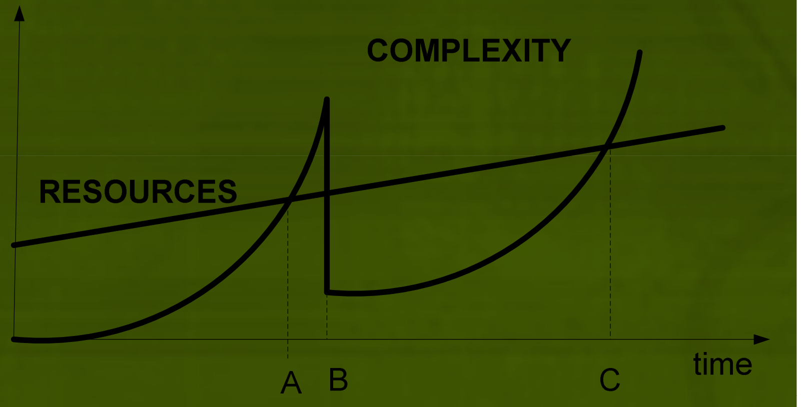
QUESTION

What is the best or most appropriate paradigm to reduce complexity?

There is no agreement!

ELEMENTS OF THE SOLUTION

- System view
- Modeling and Architecture View
- Language based
- Methodology based
- Human element

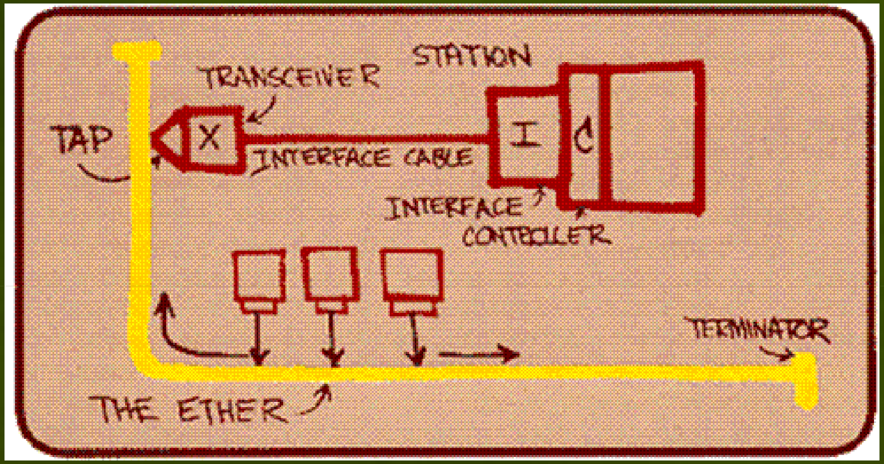


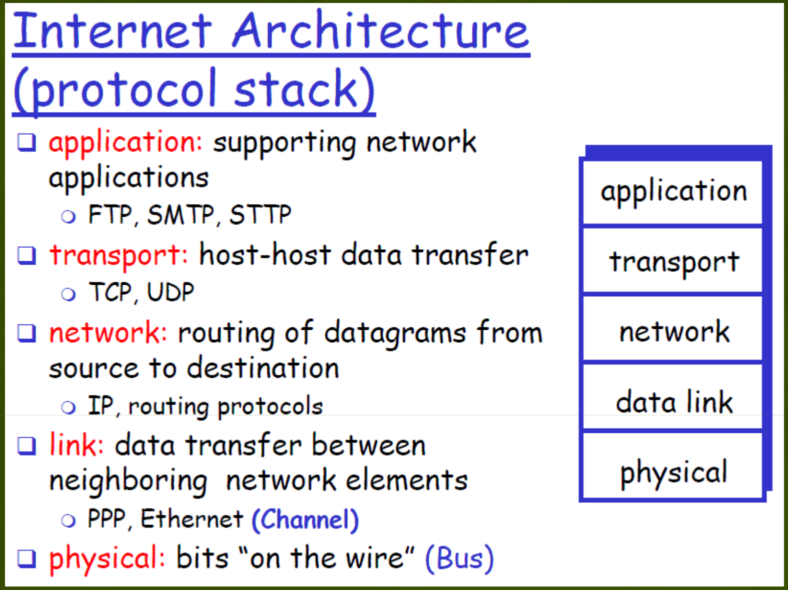
FURTHER QUESTIONS

- What is a System?
- What is an Architecture?

Proposed Approach

QUESTION
What do these have in common?

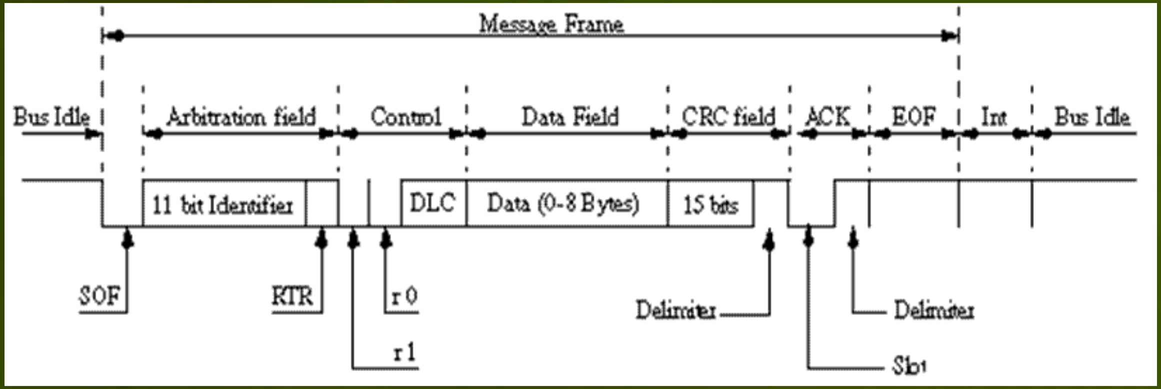




Internet Architecture (protocol stack)

- application: supporting network applications
  - FTP, SMTP, STTP
- transport: host-host data transfer
  - TCP, UDP
- network: routing of datagrams from source to destination
  - IP, routing protocols
- link: data transfer between neighboring network elements
  - PPP, Ethernet (Channel)
- physical: bits "on the wire" (Bus)

application
transport
network
data link
physical



Proposed Approach

The three figures:

- Ethernet protocol idea
- CAN protocol format
- Internet network architecture

COMMON THINGS

- Conceived by humans not computers, or tools
- Architecture or constituent elements of an architecture
- Simple
- Hugely successful

Proposed Approach - Generic

- Start at the highest level using graphical means
- View a product & components as architectures at various levels
- Define : architectures, constituent elements, and interfaces
- Keep it simple
- Validate the architecture and its constituent elements using models, analysis, simulation, and prototypes
- Iterate the above steps for each level and element
- Use languages, tools, processes, and standards appropriately and sparingly.

Specific Phased Approach: SLD or SoC

- Phase I: Behavioral VHDL-AMS modeling and simulation
- Phase II: Behavioral VHDL-AMS architectural modeling and simulation
- Phase III: Digital properties modeling and simulation
- Phase IV: Synthesizable modeling and simulation
- Phase V: Logic Synthesis
- Phase VI: Overall final test, verification, and validation

Final Output: FPGA/SoC Design of the AC motor controller

VHDL-AMS Language

Entornos de Simulation

- FORTRAN
- CSMP
- C, C++, JAVA
- SPICE
- VHDL
- MATHCAD
- MATLAB/SIMULINK
- VHDL-AMS

VHDL:

Very High Speed Hardware Description Language

AMS: Analog Mix-Signal

Can be used for modeling, simulation, specification, and synthesis of phases I through IV of the methodology

VHDL-AMS

Estandard del IEEE

El mas sofisticado y moderno de los lenguajes de programacion

Cubre muchas areas:

- Digital (superset del VHDL)
- analog
- mixed-signal
- thermal
- mechanical
- fluidic

Disenado por Ingenieros para Ingenieros

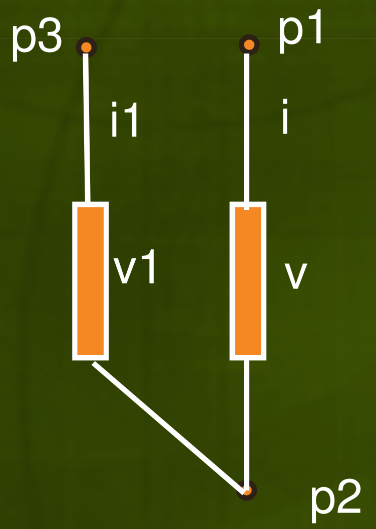
- no se basa en un modelo de computacion
- se basa en modelar dispositivos fisicos
- tiene nocion del tiempo, de unidades

VHDL-AMS

- Soluciona ecuaciones en forma automatica
- (obedece a leyes de conservacion de energia)
- El ciclo de simulacion esta bien definido
- Tiene nocion del tiempo y de unidades
- Incorpora los siguientes objetos (ademas de los disponibles en el VHDL):
    - quantity
    - terminal
- Modela comportamiento
    - continuo
    - discontinuo
- Basado en VHDL y ADA
- El mas aceptado a nivel mundial

VHDL-AMS: Sistemas conservativos

Quantity v across i through p1 to p2;
Quantity v1 across i1 through p3 to p2;



VHDL-AMS: Modelo de un Inductor

v = L di/dt

En formato VHDL-AMS

v == L*i’dot;

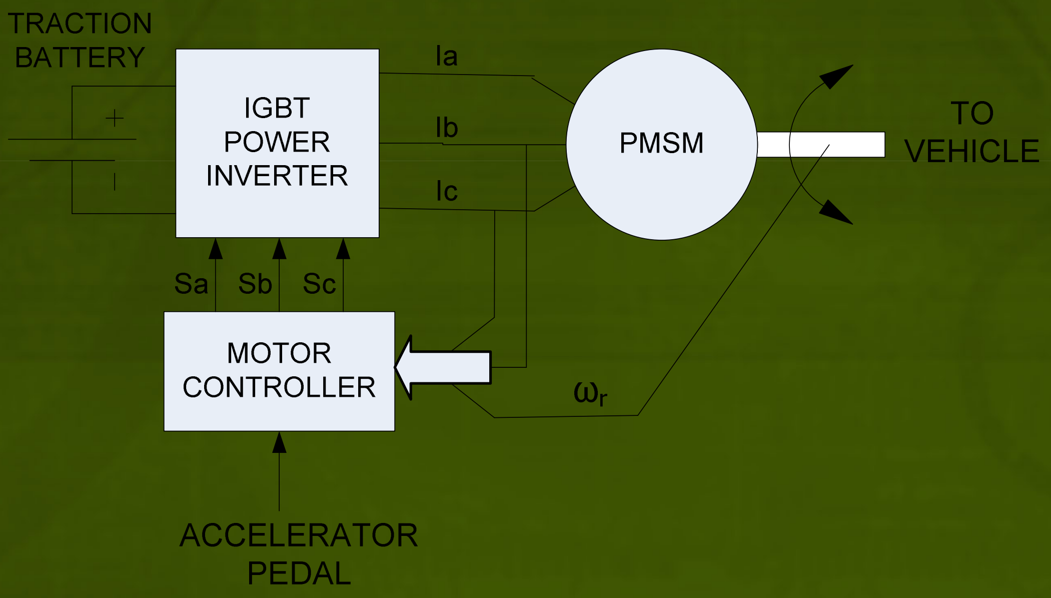
```
library ieee_proposed; use ieee_proposed.electrical_systems.all;

entity inductor is
  port(terminal n1,n2: electrical);
end entity inductor;

architecture ideal of inductor is
   constant L: inductance := 0.5;
   quantity branch_voltage across branch_current through n1 to n2;
begin
   branch_voltage == L*branch_current’dot;
end architecture ideal;
```

Case Study
Design and implementation of the powertrain of HEVs, EVs, PEVs

- It is a complex system
  - 3-phase AC motors
  - Power electronics
  - Non-linear controls
  - Hardware synthesis



- Challenge: Produce designs with:
  - High productivity gains
  - Reduced complexity
  - Excellent performance
  - Low cost

- Modeling and Simulation based: SystemVision (Mentor Graphics) – Implements VHDL-AMS

Specific Phased Approach: SLD or SoC

- Phase I: Behavioral VHDL-AMS modeling and simulation

- Phase II: Behavioral VHDL-AMS architectural modeling and simulation

- Phase III: Digital properties modeling and simulation

- Phase IV: Synthesizable modeling and simulation

- Phase V: Logic Synthesis

- Phase VI: Overall final test, verification, and validation

Final Output: FPGA/SoC Design of the AC motor controller

Specific Phased Approach: SLD or SoC

Phase I: Behavioral VHDL-AMS modeling and simulation

- Simplified models are used
  E.g., model motor in synchronous coordinates
- Emphasis: intuitive understanding of component behavior
- Useful for studying control system algorithms
- Independent of target implementation
  - Microcontroller
  - FPGA
- Independent of implementation architecture
- Not directly related to synthesis

Case Study: Permanent Magnet Synchronous Motor (PMSM) Model

Motor equations

$v_d = Ri_d + d\,\Phi_d/dt - \omega\Phi_q$

$v_q = Ri_q + d\,\Phi_q/dt + \omega\Phi_d$

$\Phi_d = L_d i_d + \Phi_m$

$\Phi_q = L_q i_q$

$T_e = (3p/2)[\Phi_m i_q + (L_d - L_q)\, i_d i_q]$

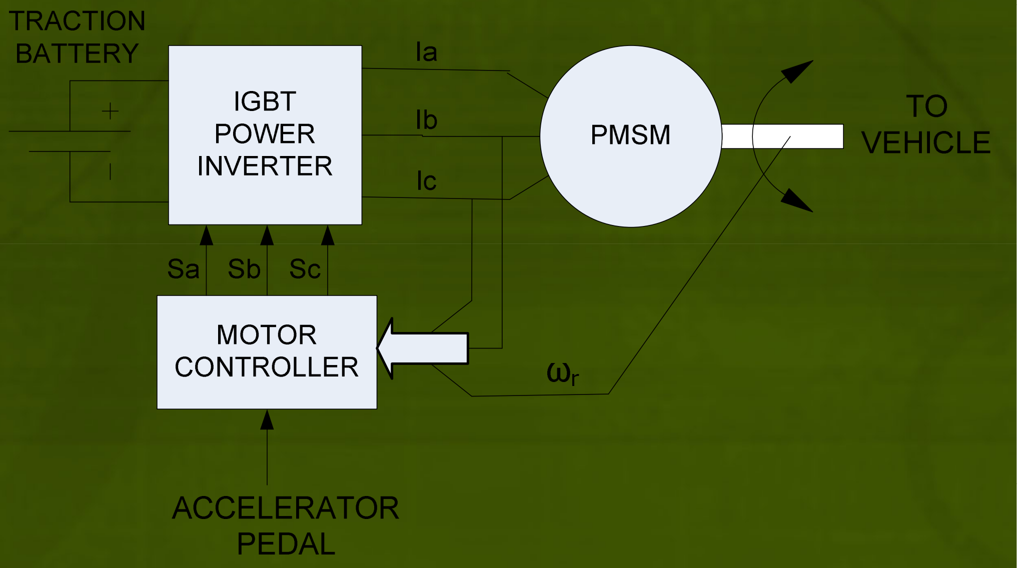
Motor parameters

v_d, v_q	stator voltages
i_d, i_q	stator currents
Φ_d, Φ_q	stator flux linkages
Φ_m	flux created by rotor
R	stator resistance
L_d, L_q	stator inductances
ω	synchronous freq.

Case Study: Phase I
AC Motor Control Algorithms

Traditional methods
- FOC:Field Oriented Control
- DTC: Direct Torque Control

Non-Linear methods
- Liapunov
- I/O Linearization
- Sliding Mode Control
- Robust Torque Control



Non-Lineal Control: Lyapunov Method …

Control Equations

$$v_d = K_d L_d e_d + K_1 L_d \theta_d + R i_d - \omega L_q i_q$$

$$v_q = K_q L_q e_q + K_2 L_q \theta_q + R i_q + \omega L_d i_d + \omega \Phi_m$$

Where

$e_d = (i^*_d - i_d)$; $\theta_d = \int e_d dt$

$e_q = (i^*_q - i_q)$; $\theta_q = \int e_q dt$

M. Quassaid, M. Cherkaoui, A. Nejmi, *and M. Maaroufi*, Nonlinear Torque Control for PMSM: A Lyapunov Technique Approach, Trans. on Engineering, Computing, and Technology, June, 2005.

Case Study: Phase I
Behavioral Model (Induction Motor)

```
library IEEE; use IEEE.std_logic_1164.all; use IEEE.math_real.all;
library IEEE; use IEEE.std_logic_1164.all;
library IEEE;
use IEEE.electrical_systems.all;
use IEEE.mechanical_systems.all;

entity IM_theta_3ph is
port
(
    terminal mot_out : ROTATIONAL_VELOCITY ;
    terminal theta_cmd : rotational ;
    terminal ia, ib, ic : electrical
);
```

Case Study: Phase I
Behavioral Model (Induction Motor)

```
begin
        wslip == theta'dot - wr;

        -- 3 to 2 phase conversion
        isqs == ia_int;
        isds == ia_int*(-1.0/sqrt(3.0)) - ib_int*(2.0/sqrt(3.0));

        -- Stationary to synchronous conversion
        ieqs == isqs*cos(theta) - isds*sin(theta);
        ieds == isqs*sin(theta) + isds*cos(theta);

        -Rr*ieqr == lmb_eqr'dot + (wslip)*lmb_edr;
        -Rr*iedr == lmb_edr'dot - (wslip)*lmb_eqr;

        lmb_eqr == Lm*ieqs + Lr*ieqr;
        lmb_edr == Lm*ieds + Lr*iedr;

        Tout == -K*(lmb_edr*ieqs - lmb_eqr*ieds);
end architecture behavioral;
```

Case Study: Phase I Transformations: Entity

```
library IEEE; use IEEE.std_logic_1164.all; use IEEE.math_real.all;
library IEEE; use IEEE.electrical_systems.all;
use IEEE.mechanical_systems.all;

entity two2three_xform is
port
(
   terminal veq_in, ved_in : electrical ;
   terminal theta_in : rotational ;
   terminal va_out, vb_out, vc_out : electrical );

end two2three_xform;
```

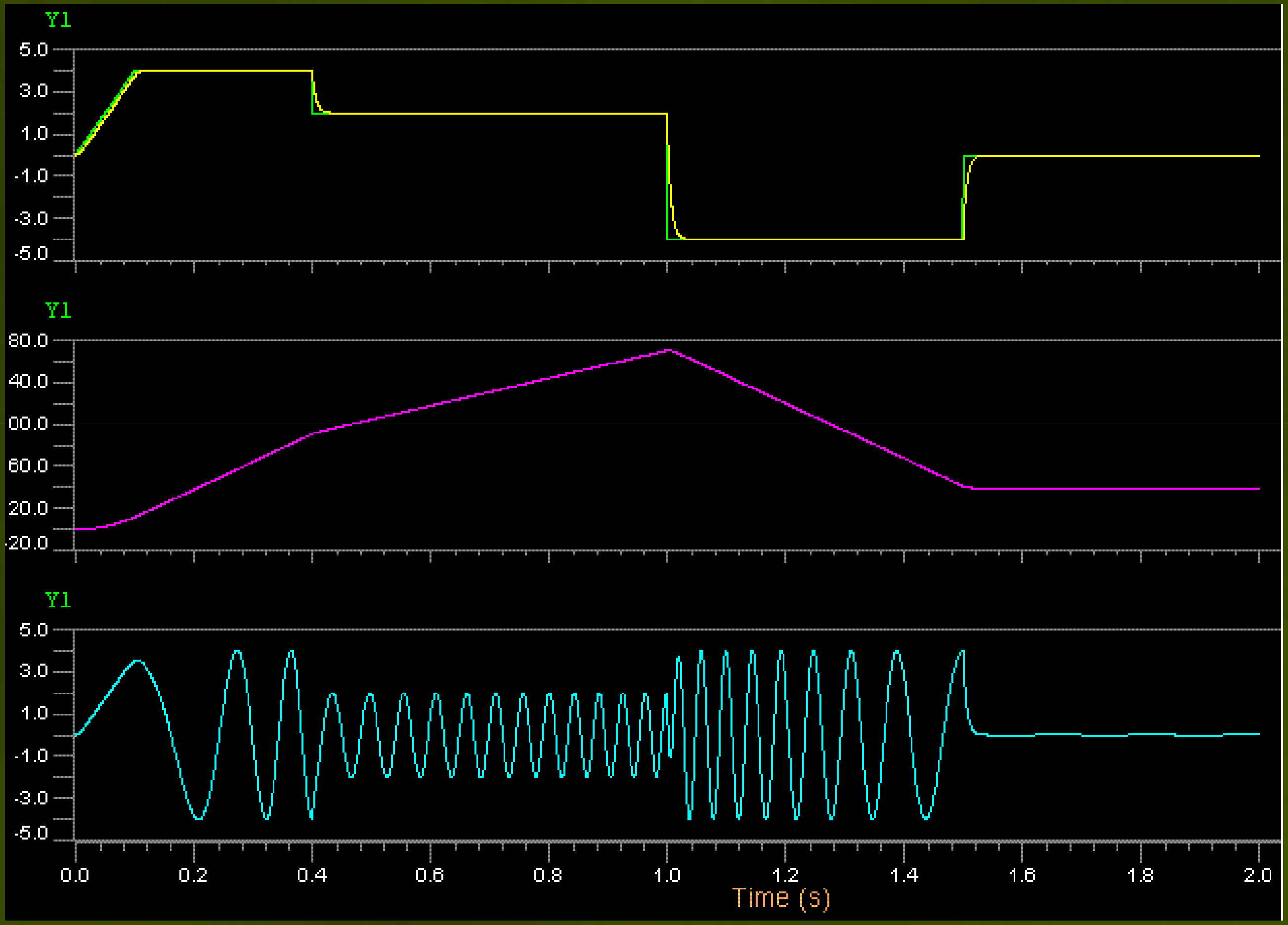
Case Study: Phase I Transformations: Architecture

```
architecture behavioral of two2three_xform is

        quantity veq across veq_in to electrical_ref;
        quantity ved across ved_in to electrical_ref;
        quantity theta across theta_in to rotational_ref;
        quantity vsa across isa through va_out to electrical_ref;
        quantity vsb across isb through vb_out to electrical_ref;
        quantity vsc across isc through vc_out to electrical_ref;
        quantity vsq : voltage ;      -- Synchronous q-axis quantity
        quantity vsd : voltage ;      -- Synchronous d-axis quantity
begin
        vsq == veq*cos(theta) + ved*sin(theta);
        vsd == -veq*sin(theta) + ved*cos(theta);
        vsa == vsq;
        vsb == -0.5*vsq - (sqrt(3.0)/2.0)*vsd;
        vsc == -vsa - vsb;

end architecture behavioral;
```

Case Study: Lyapunov Motor Controller



Commanded and actual torque in N-m

motor speed in rad/sec

three-phase currents in Amps

Specific Phased Approach: SLD or SoC

- Phase I: Behavioral VHDL-AMS modeling and simulation
- Phase II: Behavioral VHDL-AMS architectural modeling and simulation
- Phase III: Digital properties modeling and simulation
- Phase IV: Synthesizable modeling and simulation
- Phase V: Logic Synthesis
- Phase VI: Overall final test, verification, and validation

Final Output: FPGA/SoC Design of the AC motor controller

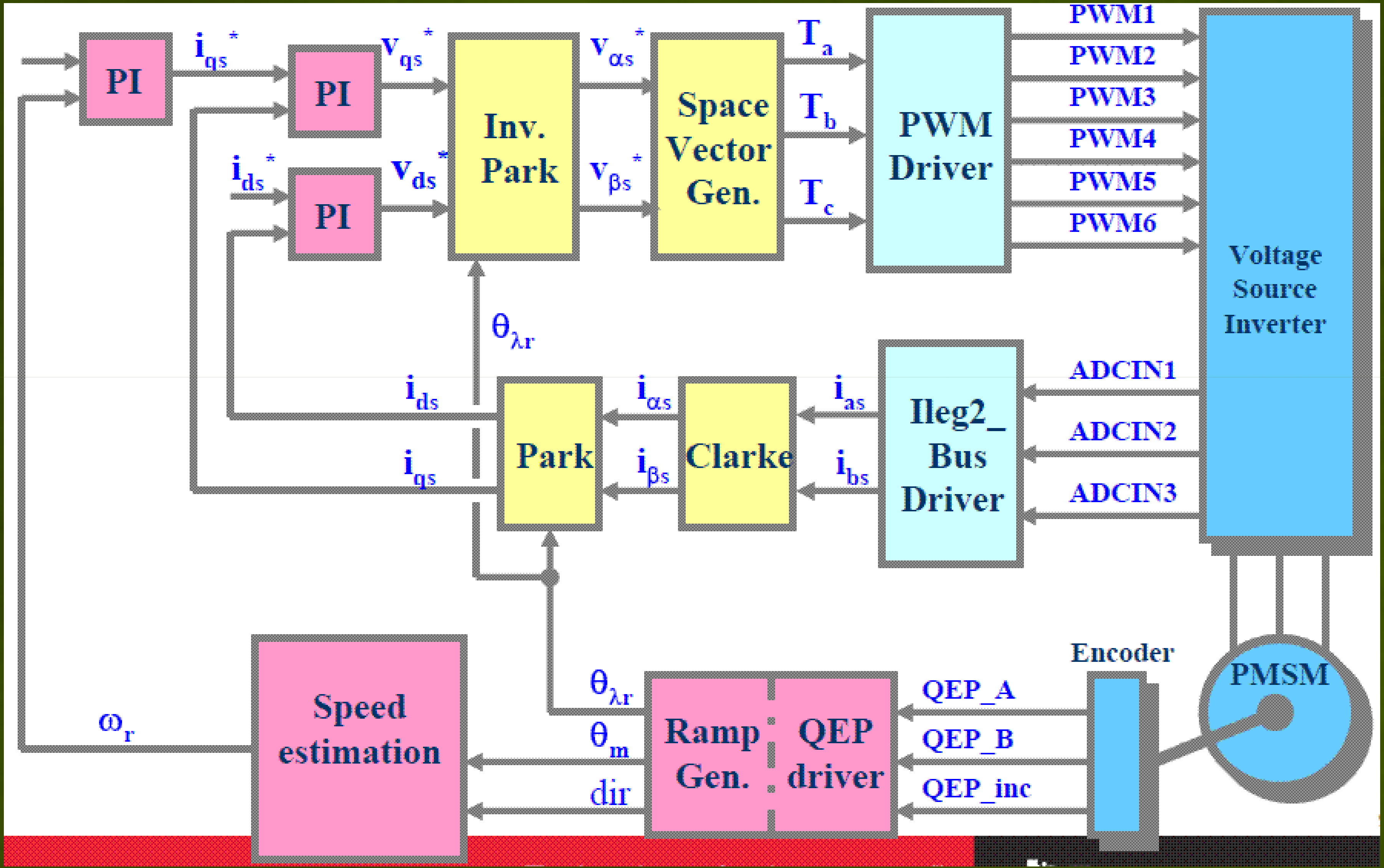
Specific Phased Approach: SLD or SoC

Phase II: Behavioral VHDL-AMS architectural modeling and simulation

- The overall system is decomposed into constituent architectural components related to final implementation
- The component interfaces are carefully defined
- The model is still independent from target hardware:
  - Microcontroller based
  - FPGA based
  - FPGA vendor

- No synthesis considerations are addressed

Classical Technique: Field Oriented Control



Case Study: Phase II

Inverter Firing Controller

Pulse Control Block

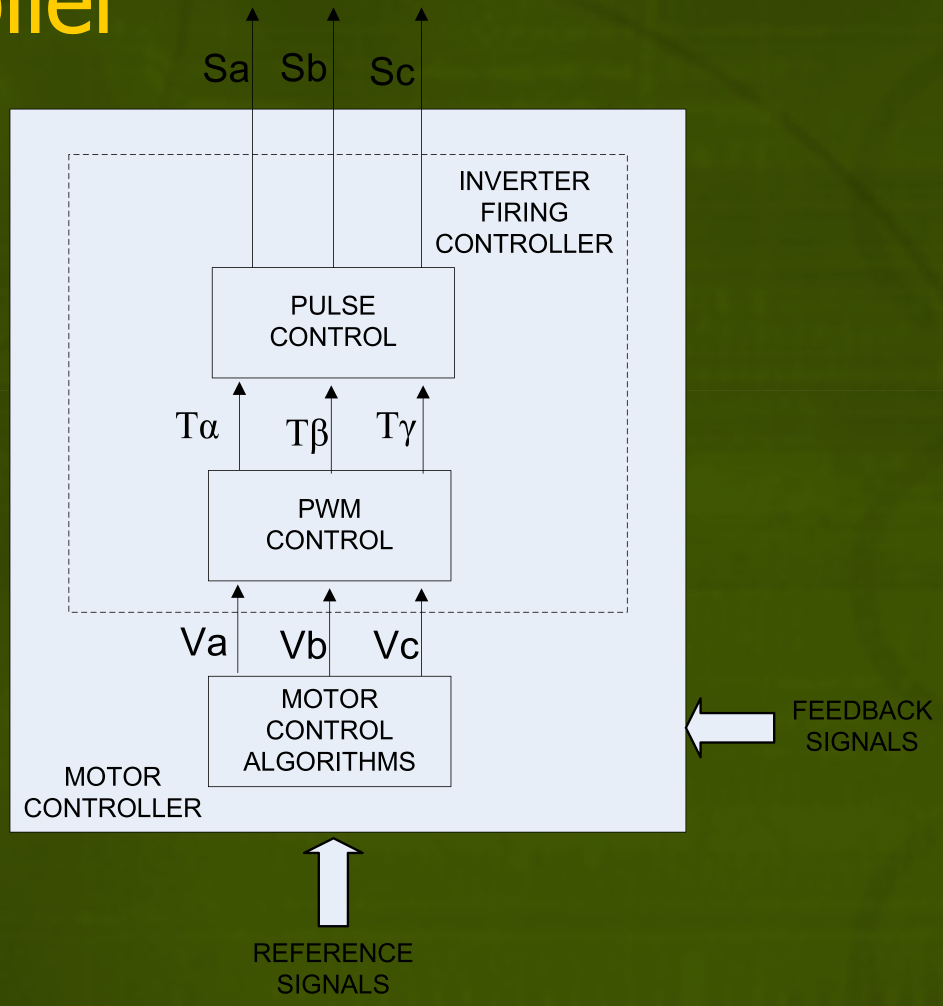
- Generate 3 digital control signal for the power inverter

PWM Control Block

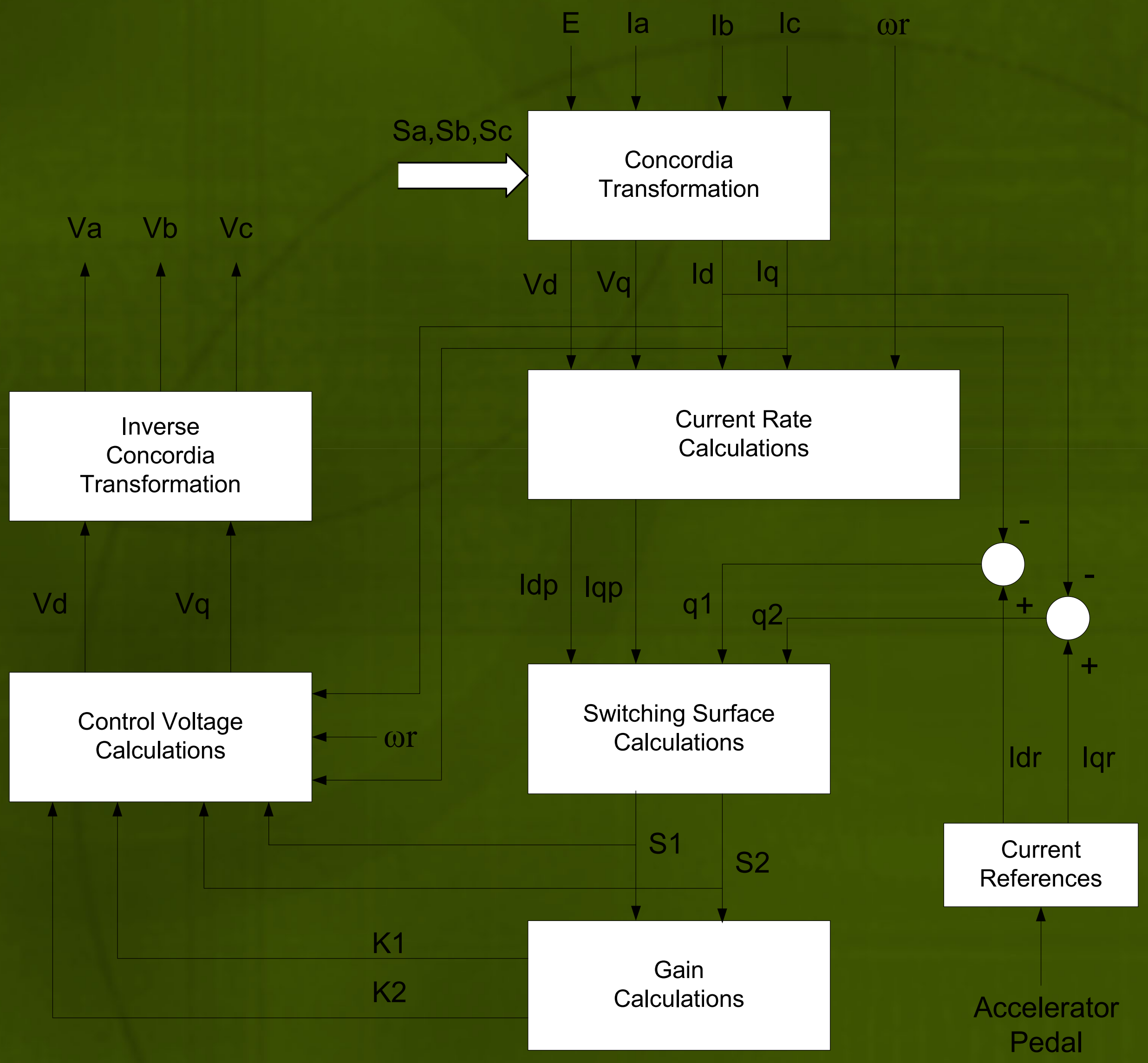
- Generate duty cycles for the three phase digital control signals

Motor Control Algorithms

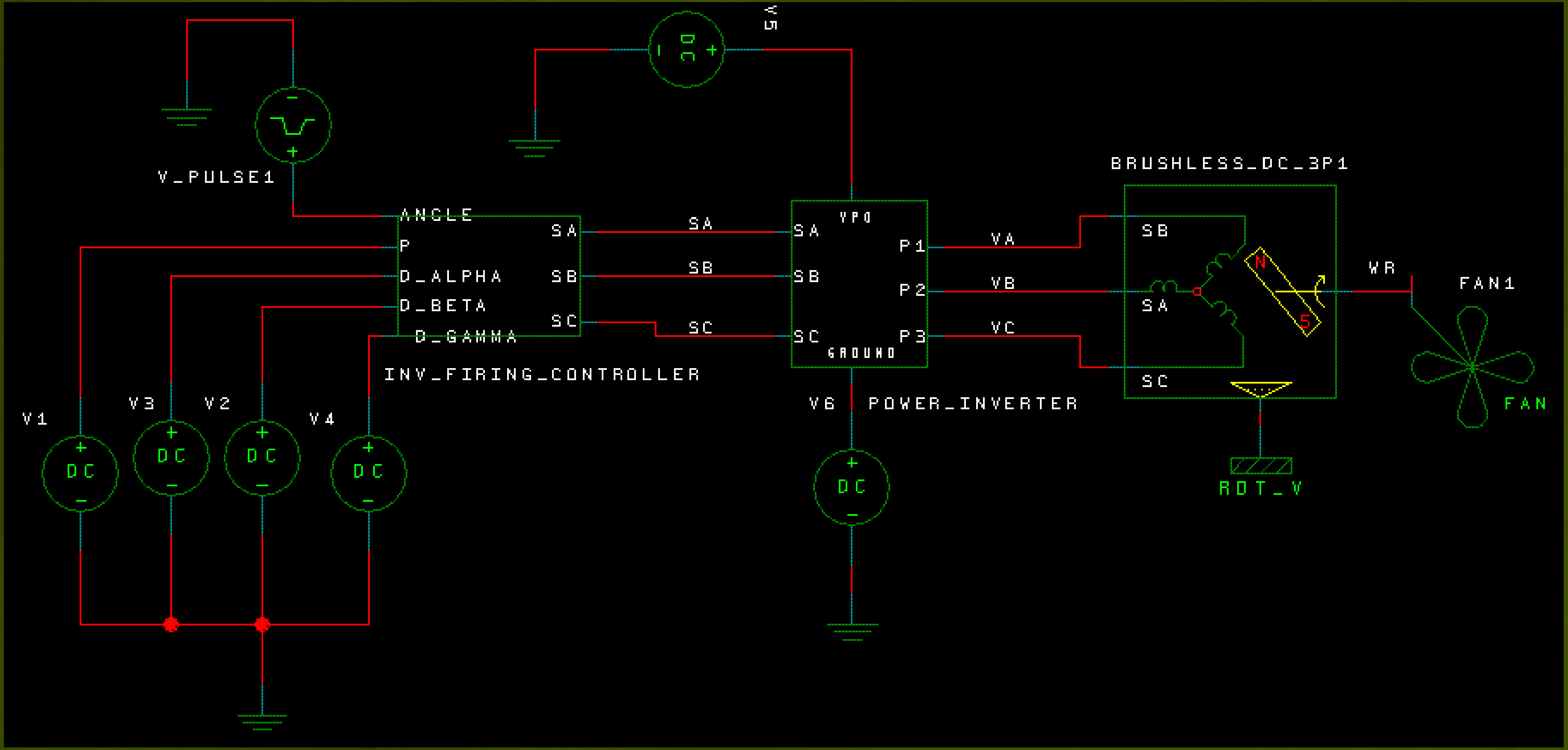
- Torque Control



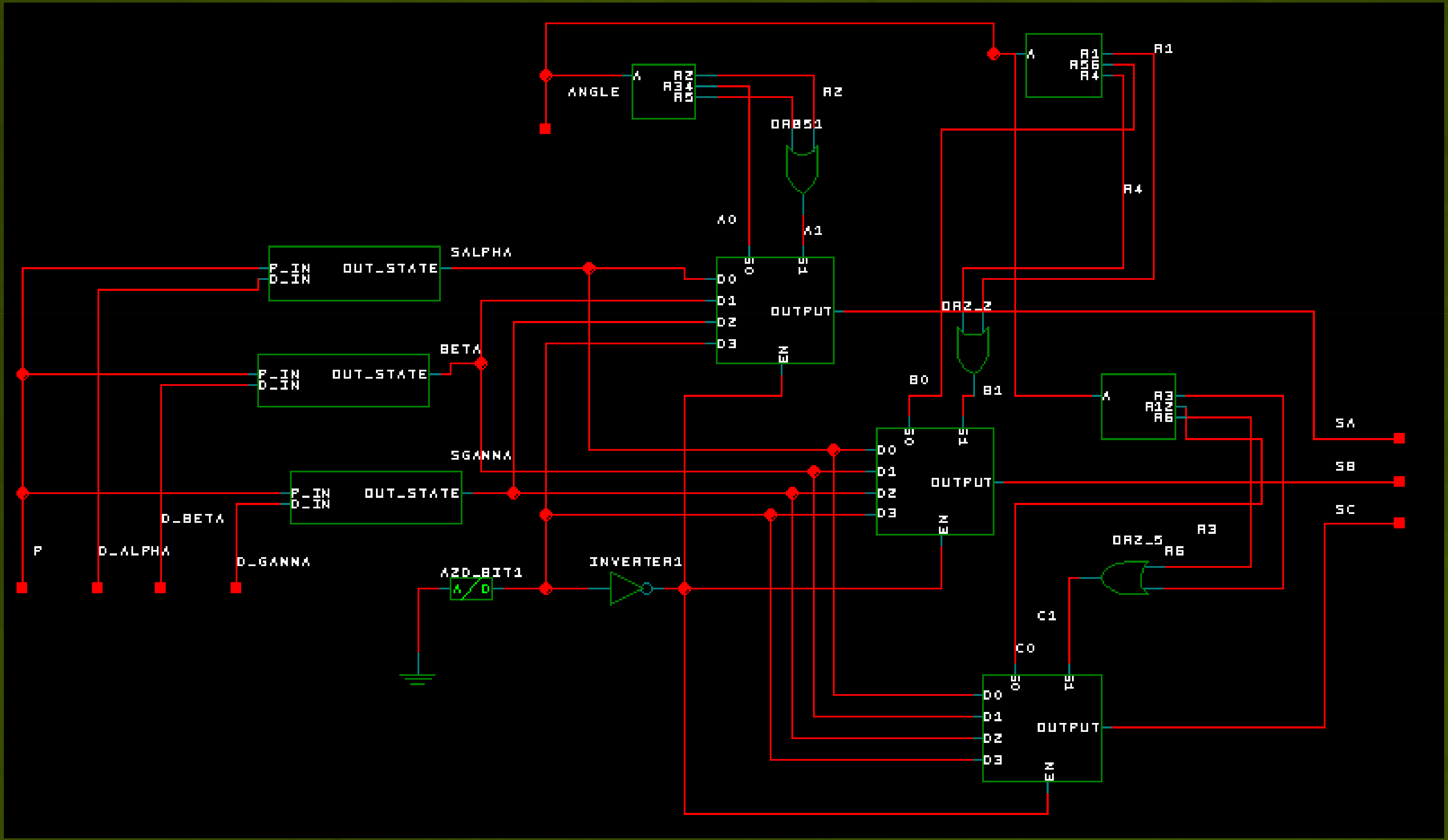
Case Study: Robust, Sliding Mode Control



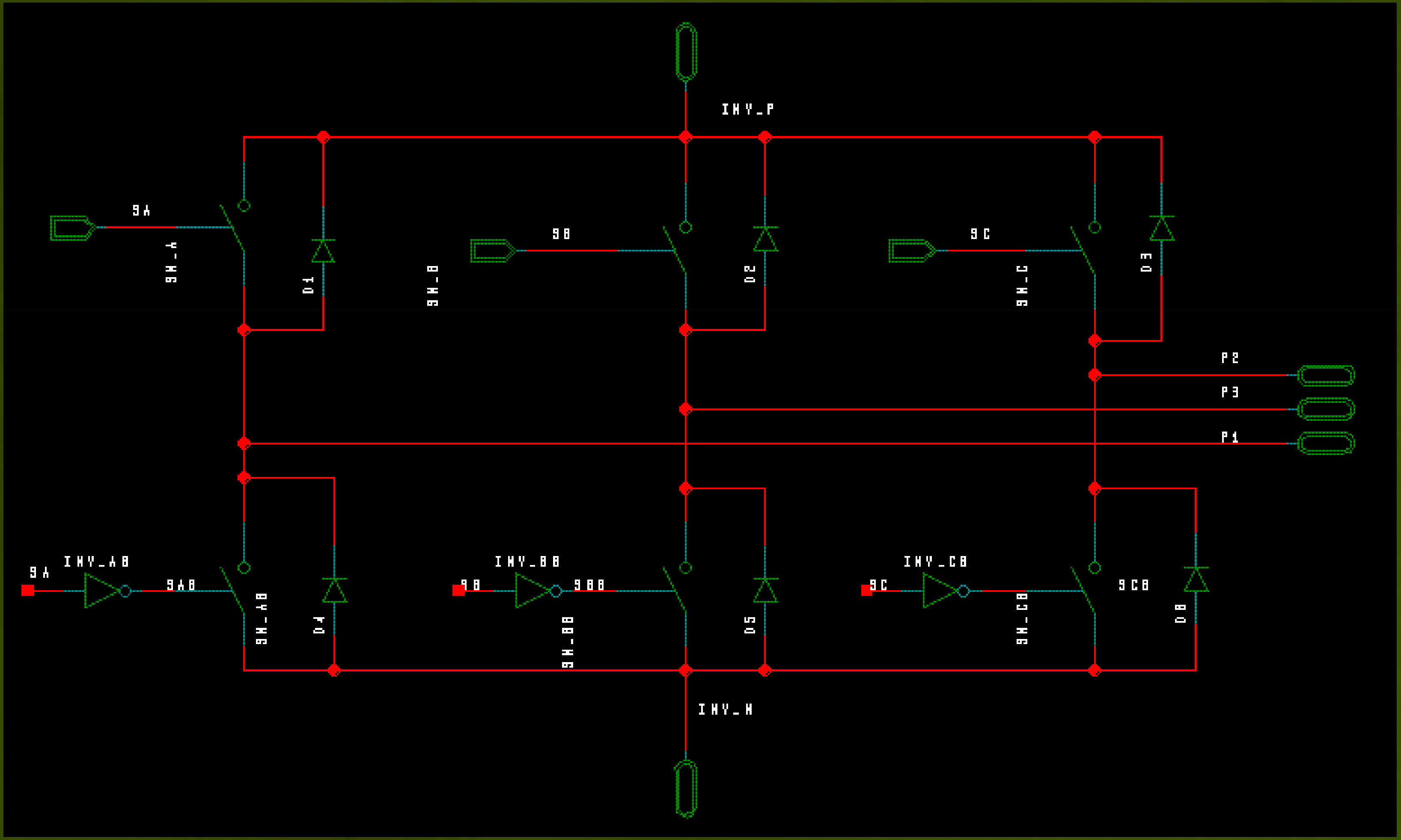
Case Study
Testing Firing Controller, Inverter, & Motor



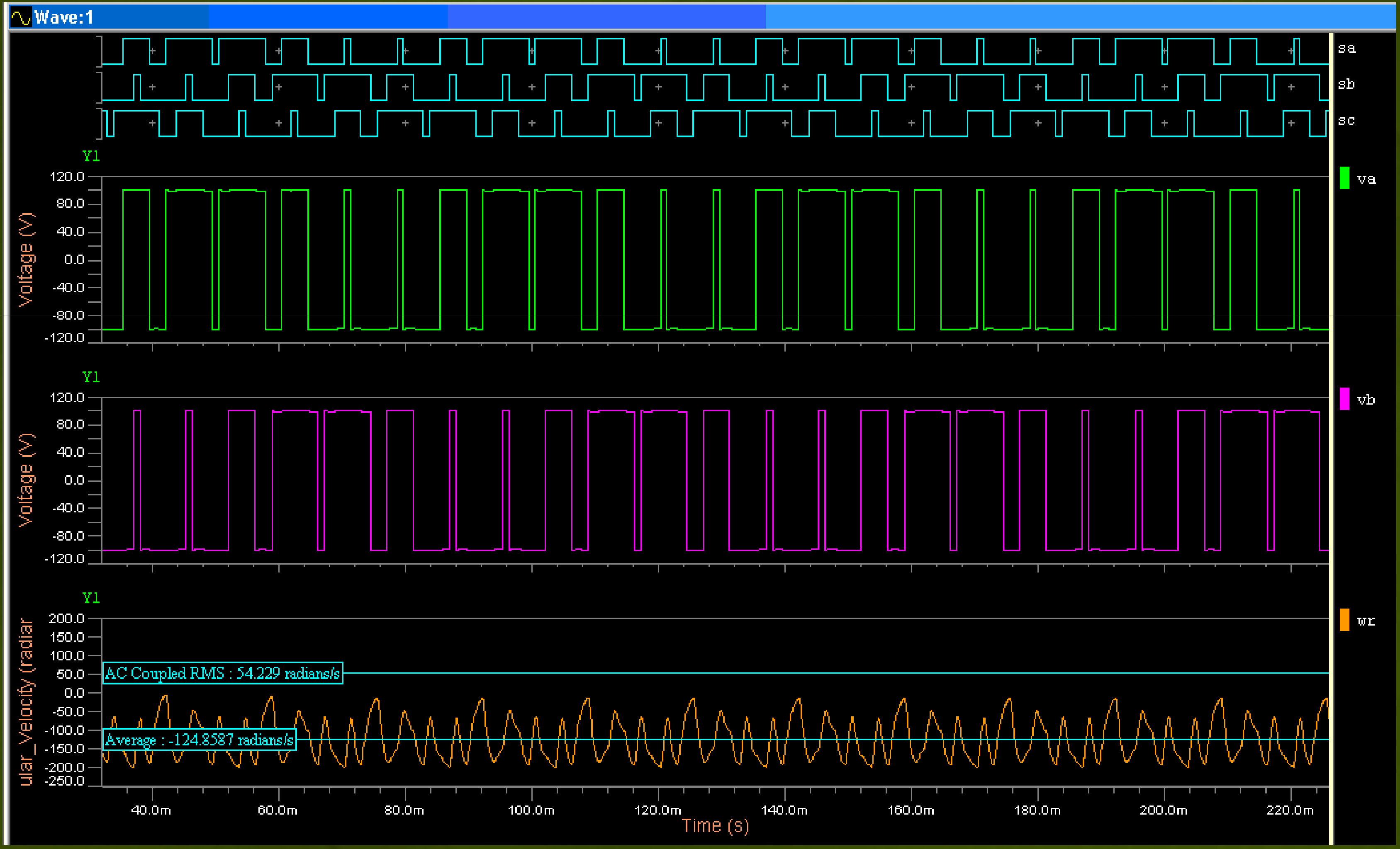
Case Study
Testing Firing Controller, Inverter, & Motor



Case Study
Testing Firing Controller, Inverter, & Motor



Case Study
Testing Firing Controller, Inverter, & Motor



Specific Phased Approach: SLD or SoC

- Phase I: Behavioral VHDL-AMS modeling and simulation
- Phase II: Behavioral VHDL-AMS architectural modeling and simulation
- Phase III: Digital properties modeling and simulation
- Phase IV: Synthesizable modeling and simulation
- Phase V: Logic Synthesis
- Phase VI: Overall final test, verification, and validation

Final Output: FPGA/SoC Design of the AC motor controller

Specific Phased Approach: SLD or SoC

Phase III: Digital properties modeling and simulation

- Detailed characteristics of target SoC are studied
    - e.g., a Xilinx Spartan 3A FPGA

- Typical considerations in a digital design:
    - use of embedded processor (microblaze or Power PC)
    - accuracy of ADC
    - fixed point or floating point arithmetic
    - accuracy of calculations
    - dsp calculations

Case Study

Digital properties model

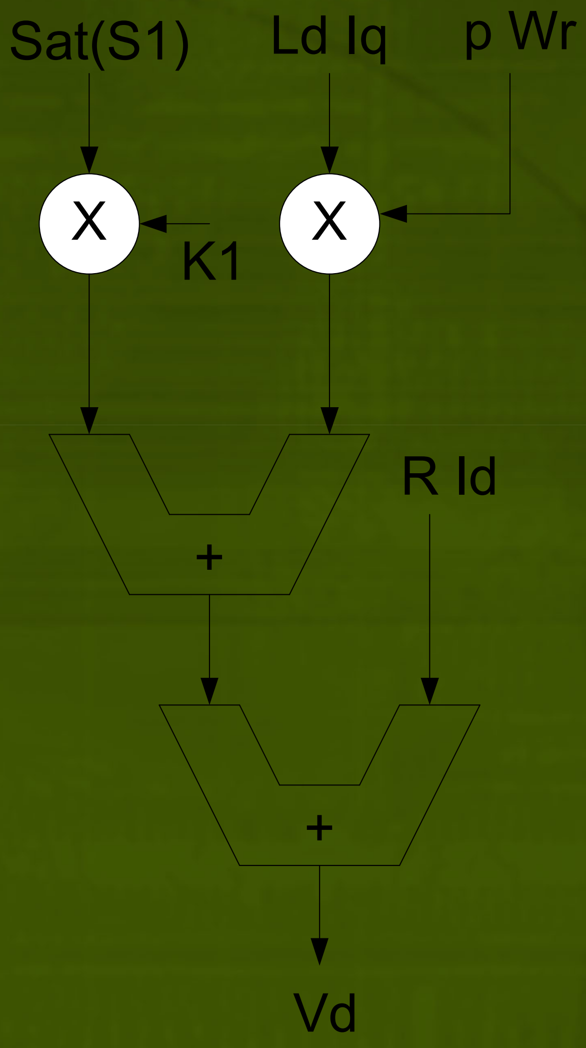
- Controller equations are of DSP type Amenable for FPGA implementation

- Sample control equation:

Vd = R Id – p Wr Lq Iq – K1 sat(S1)

Digital format 1: Fixed point
- 32 bits
- 1 sign bit
- 11 bit integer part
- 20 bit fractional part

Digital format 2: Integer (12 bit, 2’s C)



Case Study: Phase III Digital Properties: ADC

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity adc_interface is
  port
    (
    clk_i      : in     std_logic;
    rst_i      : in     std_logic;
    sck_o   : out  std_logic;
    ad_conv_o :   out   std_logic;
    sdo_i      : in     std_logic;
    ch0_o      :   out   std_logic_vector(11 downto 0);
    ch1_o      :   out   std_logic_vector(11 downto 0)
    );
end; -- entity adc_interface
```

Case Study: Phase III Digital Properties: ADC

```
data_read : process (clk_i, rst_i) is
    constant rd_tmo_c : std_logic_vector(7 downto 0) := x"22"; -- 34
  begin
    if (rst_i = '1') then  rd_cntr_s <= rd_tmo_c;
      shift_reg_s <= (others => '0');  ch0_s <= (others => '0');
      ch1_s <= (others => '0');

    elsif (rising_edge(clk_i)) then
      -- Load the read counter when a conversion is commanded, then
      -- decrement it on each ADC clock falling edge until terminal count
      -- is reached.
      --
      if (sck_fe_s = '1') then
        if (ad_conv_s = '1') then
          rd_cntr_s <= rd_tmo_c;
        elsif (rd_cntr_s /= 0) then
          rd_cntr_s <= rd_cntr_s - 1;
        end if;
      end if;
```

Case Study: Phase III
Digital Properties: ADC

```
-- Shift ADC data into the shift register left to right as long as
   -- the read counter has not terminated.  This will result in the
   -- 34-bit shift register containing both 12-bit samples plus a few
   -- don't care bits.
   if (sck_fe_s = '1') then  if (rd_cntr_s /= 0) then
       shift_reg_s <= shift_reg_s(32 downto 0) & sdo_i;
     end if;
   end if;
   --
   -- Enable the shift register data into the corresponding sample data
   -- output registers.
   --
   if (sck_fe_s = '1') then  if (ad_conv_s = '1') then
       ch0_s <= shift_reg_s(31 downto 20);
       ch1_s <= shift_reg_s(15 downto 4);
     end if;
   end if;
  end if;
 end process; -- data_read
```

Case Study

Digital properties model

```
--
-- f3 = f2*(a*(f1) + b)
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity signal_proc is
  port
    (
    f1_i      : in    std_logic_vector(11 downto 0);
    f2_i      : in    std_logic_vector(11 downto 0);
    coef_a_i  : in    std_logic_vector(11 downto 0);
    coef_b_i  : in    std_logic_vector(11 downto 0);
    f3_o      :   out std_logic_vector(11 downto 0)
    );
end; -- entity signal_proc
```

Case Study

Digital properties model

```
--
-- f3 = f2*(a*(f1) + b)
--
architecture rtl of signal_proc is

  signal prod1_s    : std_logic_vector(23 downto 0);
  signal prod2_s    : std_logic_vector(23 downto 0);
  signal sum1_s     : std_logic_vector(11 downto 0);

begin  -- rtl

  prod1_s <= coef_a_i * f1_i;
  sum1_s <= prod1_s + coef_b_i;
  prod2_s <= f2_i * sum1_s;
  f3_o <= prod2_s(23 downto 12);

end rtl;
```

Specific Phased Approach: SLD or SoC

- Phase I: Behavioral VHDL-AMS modeling and simulation
- Phase II: Behavioral VHDL-AMS architectural modeling and simulation
- Phase III: Digital properties modeling and simulation
- Phase IV: Synthesizable modeling and simulation
- Phase V: Logic Synthesis
- Phase VI: Overall final test, verification, and validation

Final Output: FPGA/SoC Design of the AC motor controller

Specific Phased Approach: SLD or SoC

Phase IV: Synthesizable modeling and simulation

- Details of target SoC are considered
    - Clock signals
    - Hardware initialization
    - Synchronization signals
    - Data buffering

Case Study: Phase IV Synthesis Model (PWM)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity pwm is
  port
    (
    clk_i     : in    std_logic;
    rst_i     : in    std_logic;
    f3_i      : in    std_logic_vector(11 downto 0);
    fout_o    :   out std_logic
    );
end; -- entity pwm
```

Case Study: Phase IV Synthesis Model (PWM)

```
begin
    if (rst_i = '1') then pwm_cntr_s <= (others => '0');
      duty_s      <= (others => '0');      fout_s      <= '0';

    elsif (rising_edge(clk_i)) then pwm_cntr_s <= pwm_cntr_s + 1;

if (pwm_cntr_s = 0) then duty_s <= f3_i;
      end if;
      --
      -- Initialize the PWM output when the counter rolls over, --
                           pizza <= x"7FF" - duty_s(11 downto 1);
                           pancake <= '0' & pizza;
      if (pwm_cntr_s = pancake) then      fout_s <= '1';
      elsif (pwm_cntr_s = (x"FFF" - pancake)) then      fout_s <= '0';
      end if;

    end if;
  end process; -- pwm_ctrl
```

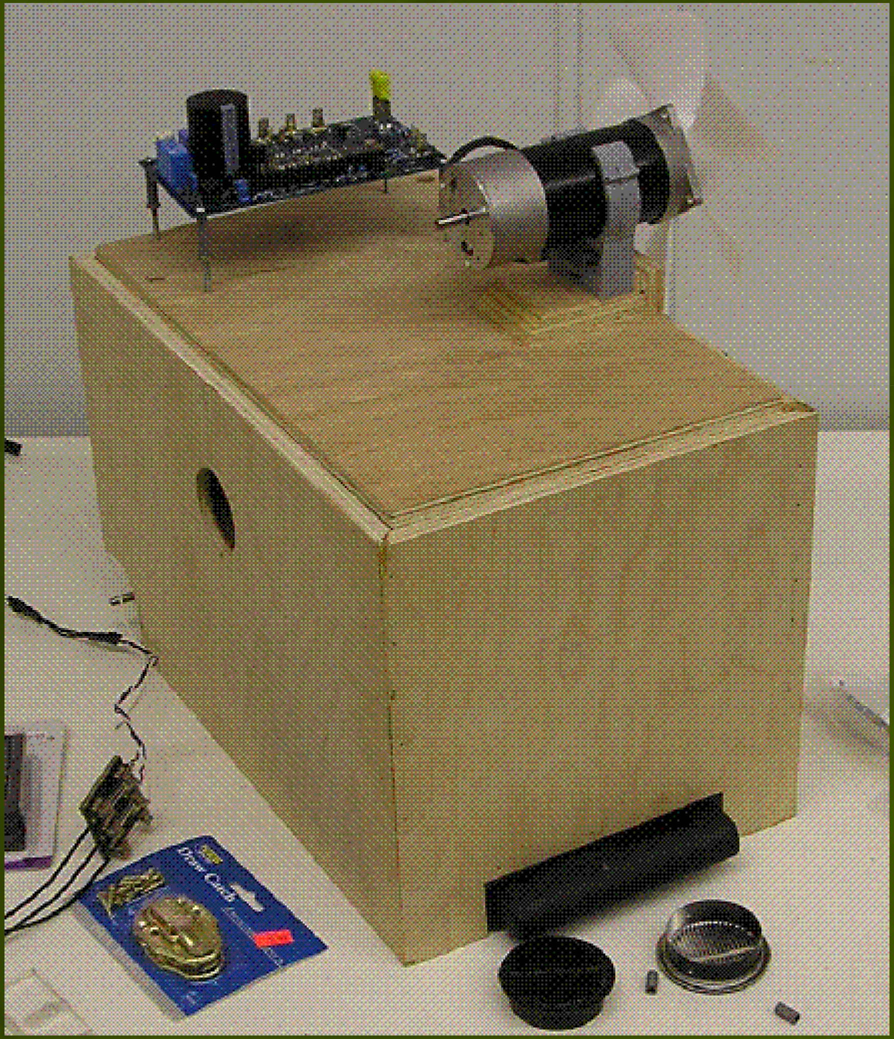
Case Study: Phases V and VI

- Synthesizable VHDL model
- Logic synthesis

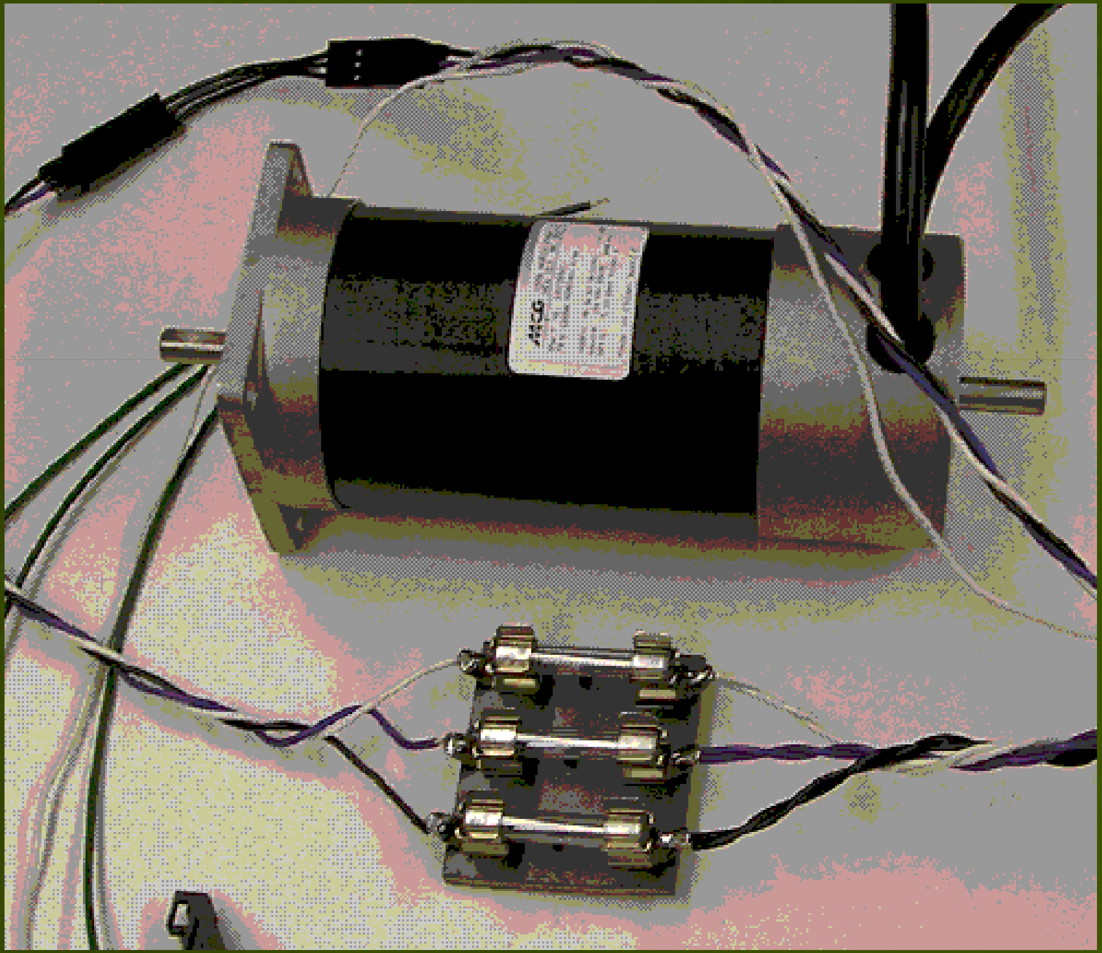
Ongoing using the following:

- Xilinx Spartan 3A FPGA
- ISE 9.2 Development environment (Xilinx)

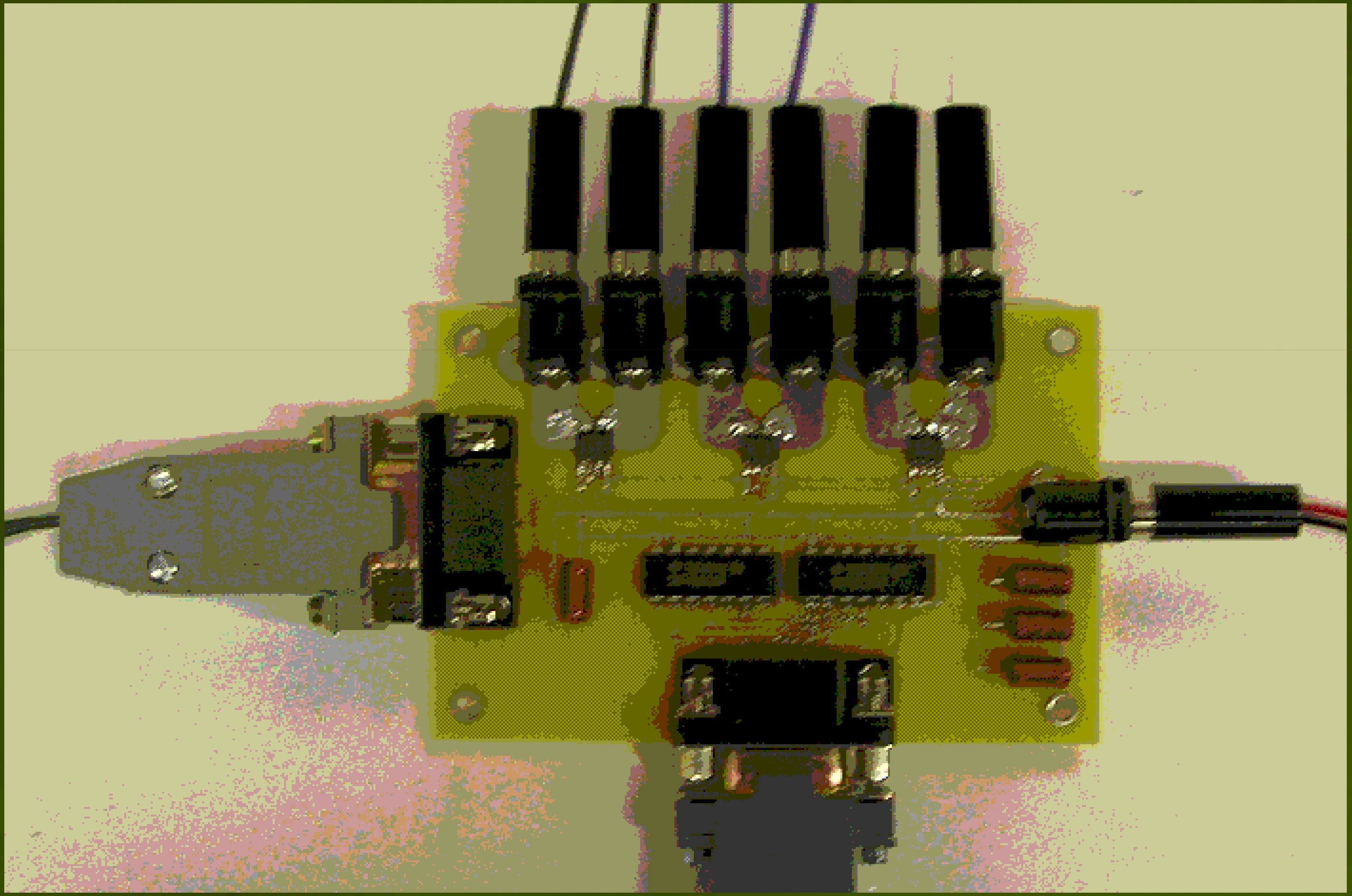
Case Study: Portable Demonstration Unit



Case Study: 3-Phase PMSM

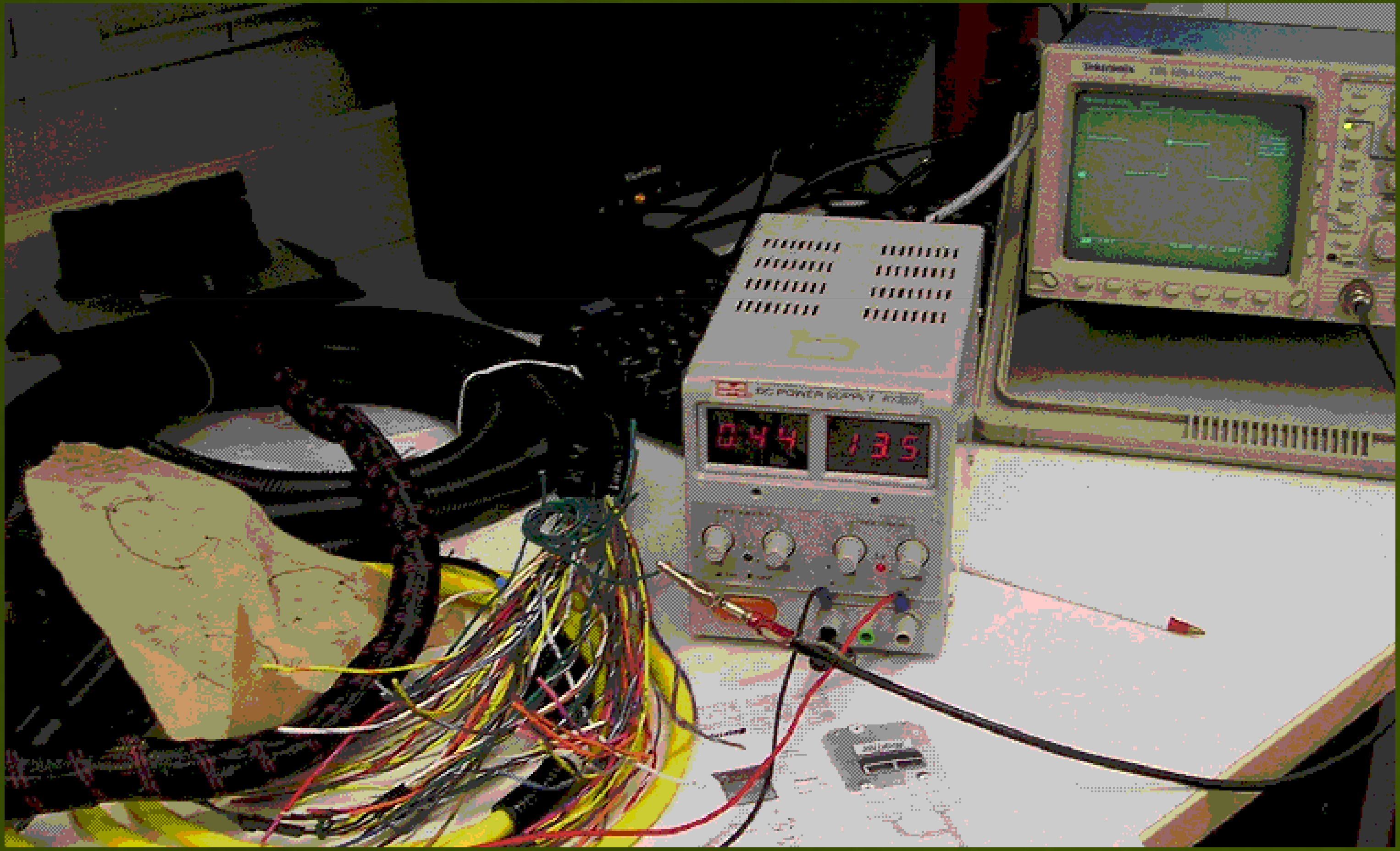


Case Study: Signal Conditioning Board



Case Study: Spartan 3A Board

Case Study: Microcontroller Synthesis



Conclusions

A system approach focused on Architecture and Modeling and simulation:

- Works to keep things simple
- Reduces complexity
- Designer knows what is going on at all times
- Test, verification, and validation is simplified
- Keep project manageable

The VHDL-AMS language is powerful and advantageous for phases I through IV of the SoC design and synthesis methodology:

- Simplifies the design and synthesis process
- Enables engineers with limited experience to be productive
- Enables and increases productivity gains
- Enables a high level of understanding
- Multiple uses: modeling, simulation, synthesis