When to use an FPGA to prototype a controller and how to start

Mark Corless, Principal Application Engineer, Novi MI
Brad Hieb, Principal Application Engineer, Novi MI
When to use an FPGA to prototype a controller and how to start

- Why would you use a processor and an FPGA to prototype a controller?
- How can the Simulink platform help you get started?
- How is modeling for an FPGA different than modeling for a processor?
Why would I use an FPGA to prototype a controller component?

- Position sensing
- High resolution voltage modulation
- Critical diagnostics
- High speed control loops
- System cost reduction

FPGA = Field Programmable Gate Array
Consider execution rate and development time when partitioning a design

**Component Execution Rate**
- 100’s Hz
- 100’s MHz

**Development Time**
- Processor (C Code)
- FPGA (HDL Code)
Let’s look at an example
How would you partition this controller with disparate rate components?

1 kHz Rate
- Velocity control
- Mode scheduler
- Encoder calibration

10 kHz Rate
- Field oriented control
- ADC to current
- Encoder to position
- Position to velocity
- Voltage to PWM

50 MHz Rate
- PWM peripheral
- Encoder peripheral
### Example of partitioning controller components based on execution rates

<table>
<thead>
<tr>
<th>Microprocessor</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 kHz Rate</strong></td>
<td></td>
</tr>
<tr>
<td>- Velocity control</td>
<td></td>
</tr>
<tr>
<td>- Mode scheduler</td>
<td></td>
</tr>
<tr>
<td>- Encoder calibration</td>
<td></td>
</tr>
<tr>
<td><strong>10 kHz Rate</strong></td>
<td></td>
</tr>
<tr>
<td>- Field oriented control</td>
<td></td>
</tr>
<tr>
<td>- ADC to current</td>
<td></td>
</tr>
<tr>
<td>- Encoder to position</td>
<td></td>
</tr>
<tr>
<td>- Position to velocity</td>
<td></td>
</tr>
<tr>
<td>- Voltage to PWM</td>
<td></td>
</tr>
<tr>
<td><strong>50 MHz Rate</strong></td>
<td></td>
</tr>
<tr>
<td>- PWM peripheral</td>
<td></td>
</tr>
<tr>
<td>- Encoder peripheral</td>
<td></td>
</tr>
</tbody>
</table>
How do I get started?
Get started building one piece at a time

- FPGA
- 50 MHz Rate
  - PWM peripheral
Video: How did the Simulink platform help us get started?
Example of partitioning controller components based on execution rates

- **Microprocessor**
  - **1 kHz Rate**
    - Velocity control
    - Mode scheduler
    - Encoder calibration
  - **10 kHz Rate**
    - Field oriented control
    - ADC to current
    - Encoder to position
    - Position to velocity
    - Voltage to PWM

- **FPGA**
  - **50 MHz Rate**
    - PWM peripheral
    - Encoder peripheral
10 kHz worked!

**Rotor Velocity (10kHz PWM)**

- velocityCommand
- rotorVelocity (Sim)
- rotorVelocity (Hw)

**Phase A Current (10kHz PWM)**

- phaseCurrentA (Hw)
- phaseCurrentA (Sim)
Where should a 50 kHz control loop be implemented?

- **Microprocessor**
  - 1 kHz Rate
    - Velocity control
    - Mode scheduler
    - Encoder calibration
  - 50 kHz Rate
    - Field oriented control
    - ADC to current
    - Encoder to position
    - Position to velocity
    - Voltage to PWM
  - 50 MHz Rate
    - PWM peripheral
    - Encoder peripheral

- **FPGA**
Timing analysis at baseline 10 kHz on CPU

Key:
- Interrupt Handling Time
- Algorithm Execution Time
- I/O Data Transfer Time
- Minimum Execution Headroom Time

Model Execution Trigger

\[ T_{10kHz} \]

Time
Timing estimation of initial design at 50 kHz

Key:
- Interrupt Handling Time
- Algorithm Execution Time
- I/O Data Transfer Time
- Minimum Execution Headroom Time

$T_{50kHz}$
Trying the initial design at 50kHz
What else do we need to be aware of to implement the 50 kHz rate on the FPGA?

**Microprocessor**
- 1 kHz Rate
  - Velocity control
  - Mode scheduler
  - Encoder calibration

**FPGA**
- 50 kHz Rate
  - Field oriented control
  - ADC to current
  - Encoder to position
  - Position to velocity
  - Voltage to PWM

- 50 MHz Rate
  - PWM peripheral
  - Encoder peripheral
Simplify timing constraints by separating disparate rates with low rate delays

Integrate 50 kHz control loop with 50 MHz peripheral components
Simplify timing constraints by separating disparate rates with low rate delays

Using slow rate delays to at rate boundaries makes it simple to create a timing constraint file for Xilinx tools.
Add fixed-point implementation details to math operations

Math is typically implemented in fixed-point when modeling for FPGA implementation
Avoid function call triggers and explicitly reset states in enabled subsystems

Explicitly reset states using signals
CPU timing with 50 kHz components on FPGA

Key:
- Interrupt Handling Time
- Algorithm Execution Time
- I/O Data Transfer Time
- Minimum Execution Headroom Time

Time

\[ T_{1kHz} \sim 900 \mu S \]
Implementing 50 kHz components on FPGA behaves correctly.
Implementing 50 kHz components on FPGA behaves correctly
When to use an FPGA to prototype a controller and how to start

- **Why would you use a processor and an FPGA to prototype a controller?**
  - When a processor can not meet execution timing for the component
  - When the final implementation will not include a processor

- **How can the Simulink platform help you get started?**
  - Simulate algorithms to reduce dependency on hardware
  - Design and generate C/HDL algorithmic code from one environment
  - Automate deployment to prototyping hardware with processor and FPGA

- **How is modeling for an FPGA different than modeling for a processor?**
  - Simplify timing constraints by separating disparate rates with low rate delays
  - Add fixed-point implementation details to math operations
  - Avoid function call triggers and explicitly reset states in enabled subsystems