HLS: PI Control

1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Introduction

1.1 Function

This IP core, implemented in the form of a C function with Vivado HLS, realizes a proportional-integral controller, a `PI controller`. This has many uses, among them as speed and/or torque control in the `field-oriented control (FOC)` method.

It transforms the input AXI4-Stream of process variables to the output AXI4-Stream of control variables using the following equation:

$$u(t) = \frac{1}{256}\left(K_p e(t) + \frac{K_i}{256}\sum_{\tau=0}^{t} e(\tau)\right), \tag{1.1}$$

where t is the current time and

$$e(t) = S_p - r(t) \tag{1.2}$$

and the parameters setpoint S_p, proportional coefficient K_p and integral coefficient K_i are input signals to the IP core.

The integral of the error value $e(t)$ is reset to zero when the operating mode of the FOC is changed, effectively starting the time t from the beginning. The values of $e(t)$ are clipped to the range MIN_LIM ... MAX_LIM; the integral of the error values is clipped to the range specified by the input signal *limit* of the IP core. The clipping avoids runaway of the PI loop when the coefficient K_i is not zero and the desired setpoint S_p can not be reached.

Implementation

Applicable Devices

This HLS C function and generated IP core can be used on any Xilinx devices supported by Vivado HLS.

Synthesis Report

The target device used for synthesis is xc7z020clg400-1.

See the chapter Vivado HLS Report for 'PI_Control' for the synthesis report, including the following:

- Estimates of the used primitives in the section "Utilization Estimates".
- Timing performance estimates in the section "Performance Estimates" for the following:
  - Maximum clock frequency.
  - Latency, both minimum and maximum.
  - Interval, both minimum and maximum.
- RTL interfaces, including AXI4-Stream interfaces and additional RTL ports added by the HLS synthesis, in the section "Interface".

Interface

The interface described in the form of a C function is as follows:

```
void PI_Control(
    hls::stream<int16_t> &s_axis,
    hls::stream<int16_t> &m_axis,
    int16_t Sp,
    int16_t Kp,
    int16_t Ki,
    int32_t mode,
    int32_t limit);
```

Simulation

A C-based testbench for C/RTL cosimulation is in the file test_pi_controller.cpp.

Tools

Vivado HLS is needed for C to RTL synthesis, for C simulation and for IP packaging (export). The function itself can be implemented with Vivado.

Doxygen is used for generating documentation from the comments included in the C source code.

Tool	Version	Notes
Vivado HLS	2017.1	Synthesis, C simulation, RTL export
Vivado	2017.1	Implementation
Doxygen	1.8.11	Documentation extraction
MiKTeX	2.9	PDF generation

Chapter 2

Vivado HLS Report for 'PI_Control'

Date:	Tue Aug 29 15:26:58 2017
Version:	2017.1 (Build 1846317 on Fri Apr 14 19:19:38 MDT 2017)
Project:	PI_Control
Solution:	solution1
Product family:	zynq
Target device:	xc7z020clg400-1

Performance Estimates

Timing (ns)

Table 2.2 Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.59	1.25

Latency (clock cycles)

Table 2.3 Summary

Latency		Interval		Pipeline
min	max	min	max	Type
8	8	9	9	none

Detail

Instance: N/A

Loop: N/A

Utilization Estimates

Table 2.4 Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	1	-	-
Expression	-	-	0	477
FIFO	-	-	-	-
Instance	-	4	166	49
Memory	-	-	-	-
Multiplexer	-	-	-	113
Register	-	-	425	-
Total	0	5	591	639
Available	280	220	106400	53200
Utilization (%)	0	2	∼0	1

Detail

Table 2.5 Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT
PI_Control_mul_16bkb_U0	PI_Control_mul_16bkb	0	4	166	49
Total		0	4	166	49

Table 2.6 DSP48

Instance	Module	Expression
PI_Control_mul_mucud_U1	PI_Control_mul_mucud	i0 ∗ i1

Memory: N/A

FIFO: N/A

Table 2.7 Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
r_V_1_fu_290_p2	+	0	0	97	90	90
x_assign_1_fu_211_p2	+	0	0	39	32	32
x_assign_fu_131_p2	-	0	0	24	17	17
x_min_assign_fu_217_p2	-	0	0	39	1	32
m_axis_V_1_load_A	and	0	0	2	1	1
m_axis_V_1_load_B	and	0	0	2	1	1
s_axis_V_0_load_A	and	0	0	2	1	1
s_axis_V_0_load_B	and	0	0	2	1	1
icmp8_fu_322_p2	icmp	0	0	17	33	1
icmp_fu_153_p2	icmp	0	0	1	2	1
m_axis_V_1_state_cmp_full	icmp	0	0	1	2	1
s_axis_V_0_state_cmp_full	icmp	0	0	1	2	1
tmp_3_fu_229_p2	icmp	0	0	16	32	32
tmp_4_fu_137_p2	icmp	0	0	13	17	16
tmp_5_fu_306_p2	icmp	0	0	24	48	16
tmp_6_fu_185_p2	icmp	0	0	16	32	32
tmp_s_fu_223_p2	icmp	0	0	16	32	32

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
tmp_11_fu_344_p2	or	0	0	2	1	1
tmp_1_fu_167_p2	or	0	0	2	1	1
Err_fu_173_p3	select	0	0	17	1	17
GiE_fu_240_p3	select	0	0	32	1	32
Res_Out_fu_348_p3	select	0	0	16	1	16
p_phitmp1_fu_337_p3	select	0	0	16	1	16
tmp_7_fu_204_p3	select	0	0	32	1	32
x_max_x_i1_cast_fu_159_p3	select	0	0	16	1	16
x_max_x_i_fu_235_p3	select	0	0	32	1	32
Total		0	0	477	353	470

Table 2.8 Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	47	10	1	10
m_axis_V_1_data_out	9	2	16	32
m_axis_V_1_state	15	3	2	6
m_axis_V_TDATA_blk↩_n	9	2	1	2
s_axis_V_0_data_out	9	2	16	32
s_axis_V_0_state	15	3	2	6
s_axis_V_TDATA_blk↩_n	9	2	1	2
Total	113	24	39	90

Table 2.9 Register

Name	FF	LUT	Bits	Const Bits
Err_cast_reg_377	32	0	32	0
Err_reg_362	17	0	17	0
GiE_prev	32	0	32	0
Mode_prev	32	0	32	0
ap_CS_fsm	9	0	9	0
icmp8_reg_433	1	0	1	0
m_axis_V_1_payload↩_A	16	0	16	0
m_axis_V_1_payload↩_B	16	0	16	0
m_axis_V_1_sel_rd	1	0	1	0
m_axis_V_1_sel_wr	1	0	1	0
m_axis_V_1_state	2	0	2	0
r_V_1_reg_422	90	0	90	0
s_axis_V_0_payload_A	16	0	16	0
s_axis_V_0_payload_B	16	0	16	0
s_axis_V_0_sel_rd	1	0	1	0
s_axis_V_0_sel_wr	1	0	1	0
s_axis_V_0_state	2	0	2	0
tmp_3_reg_397	1	0	1	0
tmp_5_reg_427	1	0	1	0
tmp_6_reg_367	1	0	1	0

Name	FF	LUT	Bits	Const Bits
tmp_9_reg_417	40	0	40	0
tmp_s_reg_392	1	0	1	0
val_assign_reg_412	32	0	32	0
x_assign_1_reg_382	32	0	32	0
x_min_assign_reg_387	32	0	32	0
Total	425	0	425	0

Interface

Table 2.10 Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	PI_Control	return value
ap_rst_n	in	1	ap_ctrl_hs	PI_Control	return value
ap_start	in	1	ap_ctrl_hs	PI_Control	return value
ap_done	out	1	ap_ctrl_hs	PI_Control	return value
ap_idle	out	1	ap_ctrl_hs	PI_Control	return value
ap_ready	out	1	ap_ctrl_hs	PI_Control	return value
s_axis_V_TDATA	in	16	axis	s_axis_V	pointer
s_axis_V_TVALID	in	1	axis	s_axis_V	pointer
s_axis_V_TREADY	out	1	axis	s_axis_V	pointer
m_axis_V_TDATA	out	16	axis	m_axis_V	pointer
m_axis_V_TVALID	out	1	axis	m_axis_V	pointer
m_axis_V_TREADY	in	1	axis	m_axis_V	pointer
Sp	in	16	ap_none	Sp	scalar
Kp	in	16	ap_none	Kp	scalar
Ki	in	16	ap_none	Ki	scalar
mode	in	32	ap_none	mode	scalar
limit	in	32	ap_none	limit	scalar

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

File Documentation

4.1 doxygen/src/main_page.dox File Reference

4.2 doxygen/src/PI_Control_csynth.dox File Reference

4.3 pi_control.cpp File Reference

PI Controller.

```
#include "pi_control.h"
#include "ap_int.h"
```

Typedefs

- typedef ap_int< 48 > int48_t

  A 48-bit signed integer type.

Functions

- void PI_Control (hls::stream< int16_t > &s_axis, hls::stream< int16_t > &m_axis, int16_t Sp, int16_t Kp, int16_t Ki, int32_t mode, int32_t limit)

  PI Controller as AXI4-Stream IP core.

4.3.1 Detailed Description

PI Controller.

Author

Oleksandr Kiyenko

Version

1.0

Date

2017

Copyright

SPDX: BSD-3-Clause 2016-2017 Trenz Electronic GmbH

4.3.2 Typedef Documentation

4.3.2.1 int48_t

```
typedef ap_int<48> int48_t
```

A 48-bit signed integer type.

Definition at line 39 of file pi_control.cpp.

4.3.3 Function Documentation

4.3.3.1 PI_Control()

```
void PI_Control (
            hls::stream< int16_t > & s_axis,
            hls::stream< int16_t > & m_axis,
            int16_t Sp,
            int16_t Kp,
            int16_t Ki,
            int32_t mode,
            int32_t limit )
```

PI Controller as AXI4-Stream IP core.

Parameters

s_axis	Input AXI4-Stream Feedback data as 16 bit signed integer values
m_axis	Output AXI4-Stream Control
Sp	Value of the setpoint
Kp	Proportional coefficient
Ki	Integral coefficient
mode	Current operation mode of the FOC
limit	Limit of the integral part of the control variable.

Returns

Functions implementing an IP core do not return a value.

Definition at line 62 of file pi_control.cpp.

```
63 {
64 #pragma HLS interface axis port=m_axis
65 #pragma HLS interface axis port=s_axis
66     int32_t Err, GpE, GiE;
67     int16_t Res_Out;
68     int16_t in_data;
69
70     static int32_t GiE_prev = 0;
71     static int32_t Mode_prev = 0;
72
73     in_data = s_axis.read();                // Read one value from AXI4-Stream
74
75     Err = Clip32(Sp - in_data, MIN_LIM, MAX_LIM); // Calculate Error
76     GpE = Err;
77     GiE = Clip32(Err + (mode != Mode_prev ? 0 : GiE_prev), -limit, limit);
78     Res_Out = Clip48((int48_t(Kp*GpE) + ((int48_t(Ki)*int48_t(GiE)) >> 8)) >> 8,
      MIN_LIM, MAX_LIM);
79
80     GiE_prev = GiE;
81     Mode_prev = mode;
82     // Write output stream
83     m_axis.write(Res_Out);                              // Write result to the output stream
84 }
```

4.4 pi_control.h File Reference

PI Controller.

```
#include <hls_stream.h>
#include <ap_axi_sdata.h>
#include <ap_int.h>
#include <ap_cint.h>
#include <stdint.h>
```

Macros

- #define MAX_LIM 32767

  Maximum positive value for saturated arithmetic.
- #define MIN_LIM -32767

  Minimum negative value for saturated arithmetic.

Functions

- void PI_Control (hls::stream< int16_t > &s_axis, hls::stream< int16_t > &m_axis, int16_t Sp, int16_t Kp, int16_t Ki, int32_t mode, int32_t limit)

  PI Controller as AXI4-Stream IP core.

4.4.1 Detailed Description

PI Controller.

Author

Oleksandr Kiyenko

Version

1.0

Date

2017

Copyright

SPDX: BSD-3-Clause 2016-2017 Trenz Electronic GmbH

4.4.2 Macro Definition Documentation

4.4.2.1 MAX_LIM

```
#define MAX_LIM 32767
```

Maximum positive value for saturated arithmetic.

Definition at line 20 of file pi_control.h.

4.4.2.2 MIN_LIM

```
#define MIN_LIM -32767
```

Minimum negative value for saturated arithmetic.

Definition at line 23 of file pi_control.h.

4.4.3 Function Documentation

4.4.3.1 PI_Control()

```
void PI_Control (
            hls::stream< int16_t > & s_axis,
            hls::stream< int16_t > & m_axis,
            int16_t Sp,
            int16_t Kp,
            int16_t Ki,
            int32_t mode,
            int32_t limit )
```

PI Controller as AXI4-Stream IP core.

Parameters

s_axis	Input AXI4-Stream Feedback data as 16 bit signed integer values
m_axis	Output AXI4-Stream Control
Sp	Value of the setpoint
Kp	Proportional coefficient
Ki	Integral coefficient
mode	Current operation mode of the FOC
limit	Limit of the integral part of the control variable.

Returns

Functions implementing an IP core do not return a value.

Definition at line 62 of file pi_control.cpp.

```
{
#pragma HLS interface axis port=m_axis
#pragma HLS interface axis port=s_axis
    int32_t Err, GpE, GiE;
    int16_t Res_Out;
    int16_t in_data;

    static int32_t GiE_prev = 0;
    static int32_t Mode_prev = 0;

    in_data = s_axis.read();                // Read one value from AXI4-Stream

    Err = Clip32(Sp - in_data, MIN_LIM, MAX_LIM); // Calculate Error
    GpE = Err;
    GiE = Clip32(Err + (mode != Mode_prev ? 0 : GiE_prev), -limit, limit);
    Res_Out = Clip48((int48_t(Kp*GpE) + ((int48_t(Ki)*int48_t(GiE)) >> 8)) >> 8, MIN_LIM, MAX_LIM);

    GiE_prev = GiE;
    Mode_prev = mode;
    // Write output stream
    m_axis.write(Res_Out);                              // Write result to the output stream
}
```

4.5 test_pi_controller.cpp File Reference

C testbench for the PI Controller.

```
#include "pi_controller.h"
```

Macros

- #define TEST_SIZE 20

  Loop count for the testbench.

Functions

- int main ()

  Main function of the C testbench.

4.5.1 Detailed Description

C testbench for the PI Controller.

Author

Oleksandr Kiyenko

Version

1.0

Date

2017

Copyright

SPDX: BSD-3-Clause 2016-2017 Trenz Electronic GmbH

4.5.2 Macro Definition Documentation

4.5.2.1 TEST_SIZE

```
#define TEST_SIZE 20
```

Loop count for the testbench.

Definition at line 15 of file test_pi_controller.cpp.

4.5.3 Function Documentation

4.5.3.1 main()

```
int main ( )
```

Main function of the C testbench.

Just calls PI_Controller() with some test data and prints the results.

Definition at line 22 of file test_pi_controller.cpp.

```
22           {
23     int i;
24     hls::stream<int16_t> inputStream;
25     hls::stream<int16_t> outputStream;
26     int16_t tx_data, Sp, Kp, Ki;
27     int16_t rx_data;
28     float inf, Spf, Kpf, Kif, outf;
29 
30     Sp = 1000;
31     Kp = 128;
32     Ki = 128;
33     rx_data = 0;
34 
35     Spf = float(Sp);
36     Kpf = Kp/256.0;
37     Kif = Ki/256.0;
38     outf = 0.0;
39 
40     for(i = 0; i < TEST_SIZE; i++){
41         tx_data = rx_data;
42         inputStream << tx_data;
43 
44         PI_Controller(inputStream, outputStream, Sp, Kp, Ki, 3, 1 << 24);
45 
46         outputStream.read(rx_data);
47         printf("Values out=%d (%f)\n",rx_data, outf);
48     }
49 }
```

Index