HLS: Park Direct

1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Introduction

Function

This IP core, implemented in the form of a C function with Vivado HLS, realizes the `Park's transform` used in the `field-oriented control (FOC)` method.It transforms the input AXI4-Stream, consisting of values I_α and I_β, to the output AXI4-Streams, consisting of the values I_d and I_s, by using the following equations:

$$I_d = I_\alpha \cos\theta + I_\beta \sin\theta, \tag{1.1}$$

$$I_q = -I_\alpha \sin\theta + I_\beta \cos\theta. \tag{1.2}$$

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: xc7z020clg400-1.

See the chapter Vivado HLS Report for 'Park_Direct' 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 Park_Direct(
    hls::stream<int64_t> &inputStream,
    hls::stream<int64_t> &outputStream);
```

See the description of the function Park_Direct() for the encoding of the input and output streams.

Simulation

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

Synthesis Report

See the chapter Vivado HLS Report for 'Park_Direct'

Chapter 2

Vivado HLS Report for 'Park_Direct'

Date:	Wed Jun 14 13:57:52 2017
Version:	2017.1 (Build 1846317 on Fri Apr 14 19:19:38 MDT 2017)
Project:	Park_Direct
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	8.45	1.25

Latency (clock cycles)

Table 2.3 Summary

Latency		Interval		Pipeline
min	max	min	max	Type
6	6	7	7	none

Detail

Instance: N/A

Loop: N/A

Utilization Estimates

Table 2.4 Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	4	-	-
Expression	-	-	0	104
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	2	-	0	0
Multiplexer	-	-	-	107
Register	-	-	531	-
Total	2	4	531	211
Available	280	220	106400	53200
Utilization (%)	∼0	1	∼0	∼0

Detail

Instance: N/A

Table 2.5 DSP48

Instance	Module	Expression
Park_Direct_mac_meOg_U2	Park_Direct_mac_meOg	i0 + i1 ∗ i2
Park_Direct_mac_mfYi_U3	Park_Direct_mac_mfYi	i0 - i1 ∗ i2
Park_Direct_mul_mdEe_U0	Park_Direct_mul_mdEe	i0 ∗ i1
Park_Direct_mul_mdEe_U1	Park_Direct_mul_mdEe	i0 ∗ i1

Table 2.6 Memory

Memory	Module	BRAM_18K	FF	LUT	Words	Bits	Banks	W∗Bits∗↩ Banks
cos_table↩ _U	Park_Direct↩ _cos_tbkb	1	0	0	1000	16	1	16000
sin_table_U	Park_Direct↩ _sin_tcud	1	0	0	1000	16	1	16000
Total		2	0	0	2000	32	2	32000

FIFO: N/A

Table 2.7 Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
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
icmp3_fu_193_p2	icmp	0	0	1	2	1
icmp_fu_178_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_8_fu_205_p2	icmp	0	0	13	17	16
tmp_s_fu_217_p2	icmp	0	0	13	17	16
ld_fu_199_p3	select	0	0	17	1	15

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
Iq_fu_211_p3	select	0	0	17	1	15
tmp_12_fu_239_p3	select	0	0	16	1	16
tmp_20_cast_fu_227_p3	select	0	0	16	1	16
Total		0	0	104	50	102

Table 2.8 Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	41	8	1	8
m_axis_V_1_data_out	9	2	64	128
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	64	128
s_axis_V_0_state	15	3	2	6
s_axis_V_TDATA_blk↩_n	9	2	1	2
Total	107	22	135	280

Table 2.9 Register

Name	FF	LUT	Bits	Const Bits
Ia_cos_reg_325	32	0	32	0
Ialpha_reg_285	16	0	16	0
Ib_cos_reg_335	32	0	32	0
Ibeta_reg_290	16	0	16	0
ap_CS_fsm	7	0	7	0
cos_table_load_reg_310	16	0	16	0
icmp3_reg_355	1	0	1	0
icmp_reg_350	1	0	1	0
m_axis_V_1_payload↩_A	64	0	64	0
m_axis_V_1_payload↩_B	64	0	64	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
s_axis_V_0_payload_A	64	0	64	0
s_axis_V_0_payload_B	64	0	64	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
sin_table_load_reg_315	16	0	16	0
tmp_13_reg_305	32	0	32	0
tmp_1_reg_340	17	0	17	0
tmp_2_reg_345	17	0	17	0
tmp_5_reg_320	32	0	32	0
tmp_6_reg_330	32	0	32	0
Total	531	0	531	0

Interface

Table 2.10 Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	Park_Direct	return value
ap_rst_n	in	1	ap_ctrl_hs	Park_Direct	return value
ap_start	in	1	ap_ctrl_hs	Park_Direct	return value
ap_done	out	1	ap_ctrl_hs	Park_Direct	return value
ap_idle	out	1	ap_ctrl_hs	Park_Direct	return value
ap_ready	out	1	ap_ctrl_hs	Park_Direct	return value
s_axis_V_TDATA	in	64	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	64	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

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/Park_Direct_csynth.dox File Reference

4.3 park_direct.cpp File Reference

Implementation of the Park's transform.

```
#include "park_direct.h"
#include "sin_cos_table.h"
```

Functions

- void Park_Direct (hls::stream< int64_t > &s_axis, hls::stream< int64_t > &m_axis)

  Park's transform as AXI4-Stream IP core.

4.3.1 Detailed Description

Implementation of the Park's transform.

Author

Oleksandr Kiyenko

Version

1.0

Date

2017

Copyright

4.3.2 Function Documentation

4.3.2.1 Park_Direct()

```
void Park_Direct (
            hls::stream< int64_t > & s_axis,
            hls::stream< int64_t > & m_axis )
```

Park's transform as AXI4-Stream IP core.

It calculates the values I_d and I_s in the ouput AXI4-Stream `m_axis` by using the following equations:

$$I_d = I_\alpha \cos\theta + I_\beta \sin\theta, \tag{4.1}$$

$$I_q = -I_\alpha \sin\theta + I_\beta \cos\theta, \tag{4.2}$$

where I_α, I_β and θ are from the input AXI4-Stream `s_axis`.

Parameters

s_axis	Input AXI4-Stream with the following layout: • Bits 0..15: I_a, from the ADC. • Bits 16..31: I_b, from the ADC. • Bits 32..47: Speed, in RPM, just passed through. • Bits 48..63: Angle, in encoder steps. All values are 16-bit signed integers.
m_axis	Output AXI4-Stream with the following layout: • Bits 0..16: I_d. • Bits 17..31: I_q. • Bits 32..47: Speed, in RPM. • Bits 48..63: Angle, in encoder steps. All values are 16-bit signed integers.

Returns

void - functions implementing an IP core do not return a value.

Definition at line 24 of file park_direct.cpp.

```
24                                                                  {
25 
26 #pragma HLS interface axis port=m_axis
27 #pragma HLS interface axis port=s_axis
28     int64_t in_data, res;
29     int16_t Ialpha, Ibeta, Theta, RPM;
30     int32_t Id, Iq;
31     int32_t cos_theta, sin_theta;
32     int32_t Ia_cos, Ib_sin, Ib_cos, Ia_sin;
```

```

    // Decode Input stream
    in_data = s_axis.read();                    // Read one value from AXI4-Stream
    Ialpha = int16_t(in_data & 0xFFFF);             // Extract Ialpha - bits[15..0] from input stream
    Ibeta = int16_t((in_data >> 16) & 0xFFFF);      // Extract Ibeta - bits[32..16] from input stream
    RPM = int16_t((in_data >> 32) & 0xFFFF);        // Extract RPM - bits[47..32] from input stream
    Theta = int16_t((in_data >> 48) & 0xFFFF);      // Extract Angle - bits[63..48] from input stream

    // Process data
    cos_theta = (int32_t)cos_table[Theta];
    sin_theta = (int32_t)sin_table[Theta];
    Ia_cos = (int32_t)Ialpha * cos_theta;
    Ib_sin = (int32_t)Ibeta * sin_theta;
    Ib_cos = (int32_t)Ibeta * cos_theta;
    Ia_sin = (int32_t)Ialpha * sin_theta;
    Id = (Ia_cos + Ib_sin) >> 15;
    Iq = (Ib_cos - Ia_sin) >> 15;
    Id = (Id > MAX_LIM) ? MAX_LIM : Id;               // Clip max
    Id = (Id < MIN_LIM) ? MIN_LIM : Id;               // Clip min
    Iq = (Iq > MAX_LIM) ? MAX_LIM : Iq;               // Clip max
    Iq = (Iq < MIN_LIM) ? MIN_LIM : Iq;               // Clip min

    // Write output stream
    res =   (((int64_t)Theta << 48) & 0xFFFF000000000000) | // Put Angle bits[63:48]
            (((int64_t)RPM << 32)   & 0x0000FFFF00000000) | // Put RPM bits[47:32]
            (((int64_t)Iq << 16)    & 0x00000000FFFF0000) | // Put Iq bits[31:16]
            ( (int64_t)Id           & 0x000000000000FFFF);  // Put Id bits[15:0]
    m_axis.write(res);                              // Write result to the output stream
}
```

4.4 park_direct.h File Reference

Header file for the Park's transform.

```
#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 Park_Direct (hls::stream< int64_t > &s_axis, hls::stream< int64_t > &m_axis)

  Park's transform as AXI4-Stream IP core.

4.4.1 Detailed Description

Header file for the Park's transform.

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 park_direct.h.

4.4.2.2 MIN_LIM

```
#define MIN_LIM -32767
```

Minimum negative value for saturated arithmetic.

Definition at line 23 of file park_direct.h.

4.4.3 Function Documentation

4.4.3.1 Park_Direct()

```
void Park_Direct (
            hls::stream< int64_t > & s_axis,
            hls::stream< int64_t > & m_axis )
```

Park's transform as AXI4-Stream IP core.

It calculates the values I_d and I_s in the ouput AXI4-Stream `m_axis` by using the following equations:

$$I_d = I_\alpha \cos\theta + I_\beta \sin\theta, \tag{4.3}$$

$$I_q = -I_\alpha \sin\theta + I_\beta \cos\theta, \tag{4.4}$$

where I_α, I_β and θ are from the input AXI4-Stream `s_axis`.

Parameters

s_axis	Input AXI4-Stream with the following layout: • Bits 0..15: I_a, from the ADC. • Bits 16..31: I_b, from the ADC. • Bits 32..47: Speed, in RPM, just passed through. • Bits 48..63: Angle, in encoder steps. All values are 16-bit signed integers.
m_axis	Output AXI4-Stream with the following layout: • Bits 0..16: I_d. • Bits 17..31: I_q. • Bits 32..47: Speed, in RPM. • Bits 48..63: Angle, in encoder steps. All values are 16-bit signed integers.

Returns

void - functions implementing an IP core do not return a value.

Definition at line 24 of file park_direct.cpp.

```
24                                                                         {
25 
26 #pragma HLS interface axis port=m_axis
27 #pragma HLS interface axis port=s_axis
28     int64_t in_data, res;
29     int16_t Ialpha, Ibeta, Theta, RPM;
30     int32_t Id, Iq;
31     int32_t cos_theta, sin_theta;
32     int32_t Ia_cos, Ib_sin, Ib_cos, Ia_sin;
33 
34     // Decode Input stream
35     in_data = s_axis.read();                    // Read one value from AXI4-Stream
36     Ialpha = int16_t(in_data & 0xFFFF);             // Extract Ialpha - bits[15..0] from input stream
37     Ibeta = int16_t((in_data >> 16) & 0xFFFF);      // Extract Ibeta - bits[32..16] from input stream
38     RPM = int16_t((in_data >> 32) & 0xFFFF);        // Extract RPM - bits[47..32] from input stream
39     Theta = int16_t((in_data >> 48) & 0xFFFF);      // Extract Angle - bits[63..48] from input stream
40 
41     // Process data
42     cos_theta = (int32_t)cos_table[Theta];
43     sin_theta = (int32_t)sin_table[Theta];
44     Ia_cos = (int32_t)Ialpha * cos_theta;
45     Ib_sin = (int32_t)Ibeta * sin_theta;
46     Ib_cos = (int32_t)Ibeta * cos_theta;
47     Ia_sin = (int32_t)Ialpha * sin_theta;
48     Id = (Ia_cos + Ib_sin) >> 15;
49     Iq = (Ib_cos - Ia_sin) >> 15;
50     Id = (Id > MAX_LIM) ? MAX_LIM : Id;               // Clip max
51     Id = (Id < MIN_LIM) ? MIN_LIM : Id;               // Clip min
52     Iq = (Iq > MAX_LIM) ? MAX_LIM : Iq;               // Clip max
53     Iq = (Iq < MIN_LIM) ? MIN_LIM : Iq;               // Clip min
54 
55     // Write output stream
56     res =   (((int64_t)Theta << 48) & 0xFFFF000000000000) | // Put Angle bits[63:48]
57             (((int64_t)RPM << 32)   & 0x0000FFFF00000000) | // Put RPM bits[47:32]
58             (((int64_t)Iq << 16)    & 0x00000000FFFF0000) | // Put Iq bits[31:16]
59             ( (int64_t)Id           & 0x000000000000FFFF);  // Put Id bits[15:0]
60     m_axis.write(res);                              // Write result to the output stream
61 }
```

4.5 sin_cos_table.h File Reference

Sinus and cosinus tables for foc function.

Variables

- short sin_table [1000]

  Lookup table for the sine function in the Q16.16 format.
- short cos_table [1000]

4.5.1 Detailed Description

Sinus and cosinus tables for foc function.

This file contains the lookup tables used by the foc() function.

Important: This file has to be updated whenever encoder has been changed to another one with different resolution.

4.5.2 Variable Documentation

4.5.2.1 cos_table

```
short cos_table[1000]
```

Definition at line 70 of file sin_cos_table.h.

4.5.2.2 sin_table

```
short sin_table[1000]
```

Lookup table for the sine function in the Q16.16 format.

Important: Update this table whenever encoder has been changed to another one with different resolution.

Definition at line 18 of file sin_cos_table.h.

4.6 test_park_direct.cpp File Reference

Testbench for the Park's transform.

```
#include "park_direct.h"
#include <math.h>
```

Macros

- #define TEST_SIZE 10

  Number of values to test with.
- #define M_PI 3.14159265358979323846

  Mathematical constant π.

Functions

- int main ()

  Main function of the C testbench.

Variables

- int Ia [TEST_SIZE] = {-600, 2000, 100, 555, -255, 3333, -765, 333, 200, -543}

  Values of I_a to test Park_Direct() with.
- int Ib [TEST_SIZE] = {-888, 3000, -500, 7000, 1000, -123, -800, 9000, 789, -444}

  Values of I_b to test Park_Direct() with.

4.6.1 Detailed Description

Testbench for the Park's transform.

Author

Oleksandr Kiyenko

Version

1.0

Date

2017

Copyright

4.6.2 Macro Definition Documentation

4.6.2.1 M_PI

```
#define M_PI 3.14159265358979323846
```

Mathematical constant π.

Definition at line 16 of file test_park_direct.cpp.

4.6.2.2 TEST_SIZE

```
#define TEST_SIZE 10
```

Number of values to test with.

Definition at line 13 of file test_park_direct.cpp.

4.6.3 Function Documentation

4.6.3.1 main()

```
int main ( )
```

Main function of the C testbench.

The function Park_Direct() will be called with the values of I_a and I_b in Ia and Ib and the results will be printed along with separately calculated values.

Definition at line 32 of file test_park_direct.cpp.

```
32           {
33     hls::stream<int32_t> inputStream;
34     hls::stream<int32_t> outputStream;
35     int32_t tx_data, rx_data;
36     int16_t Ialpha, Ibeta;
37     int16_t Theta;
38     float Iaf, Ibf, Thetaf;
39 
40 
41     Theta = 100;
42     for(int i=0; i<TEST_SIZE; i++){
43         tx_data = (int32_t(Ib[i]) << 16) | (int32_t(Ia[i]) & 0x0000FFFF);
44         inputStream << tx_data;
45 
46         Park_Direct(inputStream, outputStream, Theta);
47 
48         outputStream.read(rx_data);
49         Ialpha = int16_t(rx_data & 0xFFFF);
50         Ibeta = int16_t(rx_data >> 16);
51 
52         Thetaf = ((2*M_PI*2)/1000.0)*Theta;
53         Iaf = float(Ia[i])*cos(Thetaf) + float(Ib[i]) * sin(Thetaf);
54         Ibf = float(Ib[i])*cos(Thetaf) - float(Ia[i]) * sin(Thetaf);
55 
56         printf("Values is Ia=%d Ib=%d (%f %f)\n",Ialpha, Ibeta, Iaf, Ibf);
57     }
58 }
```

4.6.4 Variable Documentation

4.6.4.1 Ia

```
int Ia[TEST_SIZE] = {-600, 2000, 100, 555, -255, 3333, -765, 333, 200, -543}
```

Values of I_a to test Park_Direct() with.

Definition at line 19 of file test_park_direct.cpp.

4.6.4.2 Ib

```
int Ib[TEST_SIZE] = {-888, 3000, -500, 7000, 1000, -123, -800, 9000, 789, -444}
```

Values of I_b to test Park_Direct() with.

Definition at line 22 of file test_park_direct.cpp.

Index