HLS: Park Inverse

1.0

Generated by Doxygen 1.8.13

Contents

1 Introduction **1**

2 Vivado HLS Report for 'Park_Inverse' **3**

3 File Index **9**

3.1 File List 9

4 File Documentation **11**

4.1 doxygen/src/main_page.dox File Reference 11

4.2 doxygen/src/Park_Inverse_csynth.dox File Reference 11

4.3 park_inverse.cpp File Reference 11

4.3.1 Detailed Description 11

4.3.2 Function Documentation 12

4.3.2.1 Park_Inverse() 12

4.4 park_inverse.h File Reference 13

4.4.1 Detailed Description 13

4.4.2 Macro Definition Documentation 14

4.4.2.1 MAX_LIM 14

4.4.2.2 MIN_LIM 14

4.4.3 Function Documentation 14

4.4.3.1 Park_Inverse() 14

4.5 sin_cos_table.h File Reference 16

4.5.1 Detailed Description 16

4.5.2 Variable Documentation 16

4.5.2.1 cos_table 16

4.5.2.2 sin_table 16
4.6 test_park_inverse.cpp File Reference 16
4.6.1 Detailed Description 17
4.6.2 Macro Definition Documentation 17
4.6.2.1 M_PI 18
4.6.2.2 TEST_SIZE 18
4.6.3 Function Documentation 18
4.6.3.1 main() 18
4.6.4 Variable Documentation 19
4.6.4.1 Vd 19
4.6.4.2 Vq 19
Index **21**

Chapter 1

Introduction

Function

This IP core, implemented in the form of a C function with Vivado HLS, realizes the inverse Park's transform used in the field-oriented control (FOC) method.It transforms the input AXI4-Stream, consisting of values V_d, V_q and θ, to the output AXI4-Stream, consisting of values V_α, V_β and θ by using the following equations:

$$V_\alpha = V_d \cos\theta - V_q \sin\theta, \tag{1.1}$$

$$V_\beta = V_d \sin\theta + V_q \cos\theta, \tag{1.2}$$

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 is xc7z020clg400-1.

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

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

Simulation

A C-based testbench for C/RTL cosimulation is in the file test_park_inverse.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 'Park_Inverse'

Date:	Wed Jun 14 15:16:43 2017
Version:	2017.1 (Build 1846317 on Fri Apr 14 19:19:38 MDT 2017)
Project:	Park_Inverse
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	-	-	515	-
Total	2	4	515	211
Available	280	220	106400	53200
Utilization (%)	∼0	1	∼0	∼0

Detail

Instance: N/A

Table 2.5 DSP48

Instance	Module	Expression
Park_Inverse_mac_eOg_U2	Park_Inverse_mac_eOg	i0 - i1 ∗ i2
Park_Inverse_mac_fYi_U3	Park_Inverse_mac_fYi	i0 + i1 ∗ i2
Park_Inverse_mul_dEe_U0	Park_Inverse_mul_dEe	i0 ∗ i1
Park_Inverse_mul_dEe_U1	Park_Inverse_mul_dEe	i0 ∗ i1

Table 2.6 Memory

Memory	Module	BRAM_18K	FF	LUT	Words	Bits	Banks	W∗Bits∗Banks
cos_table_U	Park_Inverse_cos_bkb	1	0	0	1000	16	1	16000
sin_table_U	Park_Inverse_sin_cud	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_179_p2	icmp	0	0	1	2	1
icmp_fu_164_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_7_fu_191_p2	icmp	0	0	13	17	16

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
tmp_s_fu_203_p2	icmp	0	0	13	17	16
Valpha_fu_185_p3	select	0	0	17	1	15
Vbeta_fu_197_p3	select	0	0	17	1	15
tmp_12_fu_225_p3	select	0	0	16	1	16
tmp_17_cast_fu_213_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
Vd_cos_reg_320	32	0	32	0
Vd_reg_310	32	0	32	0
Vq_cos_reg_325	32	0	32	0
Vq_reg_315	32	0	32	0
ap_CS_fsm	7	0	7	0
cos_table_load_reg_300	16	0	16	0
icmp3_reg_345	1	0	1	0
icmp_reg_340	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_305	16	0	16	0
tmp_2_reg_330	17	0	17	0
tmp_3_reg_280	16	0	16	0
tmp_4_reg_335	17	0	17	0
tmp_5_reg_285	16	0	16	0
tmp_reg_275	16	0	16	0

Name	FF	LUT	Bits	Const Bits
Total	515	0	515	0

Interface

Table 2.10 Summary

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

park_inverse.cpp
Implementation of the inverse Park's transform 11
park_inverse.h
Header file for the inverse Park's transform 13
sin_cos_table.h
Sinus and cosinus tables for foc function 16
test_park_inverse.cpp
Testbench for the inverse Park's transform 16

Chapter 4

File Documentation

4.1 doxygen/src/main_page.dox File Reference

4.2 doxygen/src/Park_Inverse_csynth.dox File Reference

4.3 park_inverse.cpp File Reference

Implementation of the inverse Park's transform.

```
#include "park_inverse.h"
#include "sin_cos_table.h"
```

Functions

- void Park_Inverse (hls::stream< int64_t > &s_axis, hls::stream< int64_t > &m_axis)
  Inverse Park's transform as an AXI4-Stream IP core.

4.3.1 Detailed Description

Implementation of the inverse Park's transform.

Author

Oleksandr Kiyenko

Version

1.0

Date

2017

Copyright

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

4.3.2 Function Documentation

4.3.2.1 Park_Inverse()

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

Inverse Park's transform as an AXI4-Stream IP core.

It calculates the values V_α and V_β in the output AXI4-Stream `m_axis` by using the following equations:

$$V_\alpha = V_d \cos\theta - V_q \sin\theta, \tag{4.1}$$

$$V_\beta = V_d \sin\theta + V_q \cos\theta, \tag{4.2}$$

where V_d, V_q and θ are from the input AXI4-Stream `s_axis`

Parameters

s_axis	Input AXI4-Stream with the following layout: • Bits 0..15: V_d. • Bits 16..31: V_q. • Bits 32..47: Angle θ, in encoder steps. • Bits 48..63: Not used. All values are 16-bit signed integers.
m_axis	Output AXI4-Stream with the following layout: • Bits 0..15: V_α • Bits 16..31: V_β • Bits 32..47: Angle θ, in encoder steps. • Bits 48..63: 0. All values are 16-bit signed integers.

Returns

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

Definition at line 23 of file park_inverse.cpp.

```
23                                                                    {
24
25 #pragma HLS interface axis port=m_axis
26 #pragma HLS interface axis port=s_axis
27     int64_t in_data, res;
28     int32_t Vd, Vq, Theta;
29     int32_t Vd_cos, Vq_sin, Vq_cos, Vd_sin;
```

```
    int32_t Valpha, Vbeta;
    int32_t cos_theta, sin_theta;

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

    // Process data
    cos_theta = (int32_t)cos_table[Theta];
    sin_theta = (int32_t)sin_table[Theta];
    Vd_cos = Vd * cos_theta;
    Vq_sin = Vq * sin_theta;
    Vq_cos = Vq * cos_theta;
    Vd_sin = Vd * sin_theta;
    Valpha = (Vd_cos - Vq_sin) >> 15;
    Vbeta  = (Vq_cos + Vd_sin) >> 15;
    Valpha = (Valpha > MAX_LIM) ? MAX_LIM : Valpha;   // Clip max
    Valpha = (Valpha < MIN_LIM) ? MIN_LIM : Valpha;   // Clip min
    Vbeta  = (Vbeta  > MAX_LIM) ? MAX_LIM : Vbeta;    // Clip max
    Vbeta  = (Vbeta  < MIN_LIM) ? MIN_LIM : Vbeta;    // Clip min

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

4.4 park_inverse.h File Reference

Header file for the inverse 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_Inverse (hls::stream< int64_t > &s_axis, hls::stream< int64_t > &m_axis)

  Inverse Park's transform as an AXI4-Stream IP core.

4.4.1 Detailed Description

Header file for the inverse 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_inverse.h.

4.4.2.2 MIN_LIM

```
#define MIN_LIM -32767
```

Minimum negative value for saturated arithmetic.

Definition at line 23 of file park_inverse.h.

4.4.3 Function Documentation

4.4.3.1 Park_Inverse()

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

Inverse Park's transform as an AXI4-Stream IP core.

It calculates the values V_α and V_β in the output AXI4-Stream `m_axis` by using the following equations:

$$V_\alpha = V_d \cos\theta - V_q \sin\theta, \tag{4.3}$$

$$V_\beta = V_d \sin\theta + V_q \cos\theta, \tag{4.4}$$

where V_d, V_q and θ are from the input AXI4-Stream `s_axis`

Parameters

s_axis	Input AXI4-Stream with the following layout: • Bits 0..15: V_d. • Bits 16..31: V_q. • Bits 32..47: Angle θ, in encoder steps. • Bits 48..63: Not used. All values are 16-bit signed integers.
m_axis	Output AXI4-Stream with the following layout: • Bits 0..15: V_α • Bits 16..31: V_β • Bits 32..47: Angle θ, in encoder steps. • Bits 48..63: 0. All values are 16-bit signed integers.

Returns

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

Definition at line 23 of file park_inverse.cpp.

```
23                                                                          {
24
25 #pragma HLS interface axis port=m_axis
26 #pragma HLS interface axis port=s_axis
27     int64_t in_data, res;
28     int32_t Vd, Vq, Theta;
29     int32_t Vd_cos, Vq_sin, Vq_cos, Vd_sin;
30     int32_t Valpha, Vbeta;
31     int32_t cos_theta, sin_theta;
32
33     // Decode Input stream
34     in_data = s_axis.read();                    // Read one value from AXI4-Stream
35     Vd = int16_t(in_data & 0xFFFF);             // Extract Vd - bits[15..0] from input stream
36     Vq = int16_t((in_data >> 16) & 0xFFFF);     // Extract Vq - bits[32..16] from input stream
37     Theta = int16_t((in_data >> 32) & 0xFFFF);  // Extract Theta - bits[47..32] from input stream
38
39     // Process data
40     cos_theta = (int32_t)cos_table[Theta];
41     sin_theta = (int32_t)sin_table[Theta];
42     Vd_cos = Vd * cos_theta;
43     Vq_sin = Vq * sin_theta;
44     Vq_cos = Vq * cos_theta;
45     Vd_sin = Vd * sin_theta;
46     Valpha = (Vd_cos - Vq_sin) >> 15;
47     Vbeta  = (Vq_cos + Vd_sin) >> 15;
48     Valpha = (Valpha > MAX_LIM) ? MAX_LIM : Valpha;    // Clip max
49     Valpha = (Valpha < MIN_LIM) ? MIN_LIM : Valpha;    // Clip min
50     Vbeta  = (Vbeta  > MAX_LIM) ? MAX_LIM : Vbeta;     // Clip max
51     Vbeta  = (Vbeta  < MIN_LIM) ? MIN_LIM : Vbeta;     // Clip min
52
53     // Write output stream
54     res =    (((int64_t)Theta << 32)      & 0x0000FFFF00000000) | // Put Theta bits[47:32]
55              (((int64_t)Vbeta << 16)      & 0x00000000FFFF0000) | // Put Vbeta bits[31:16]
56              ( (int64_t)Valpha            & 0x000000000000FFFF);  // Put Valpha bits[15:0]
57     m_axis.write(res);                                   // Write result to the output stream
58 }
```

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_inverse.cpp File Reference

Testbench for the inverse Park's transform.

```
#include "park_inverse.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 Vd [TEST_SIZE] = {-600, 2000, 100, 555, -255, 3333, -765, 333, 200, -543}

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

    Values of V_q to test Park_Inverse() with.

4.6.1 Detailed Description

Testbench for the inverse Park's transform.

Author

Oleksandr Kiyenko

Version

1.0

Date

2017

Copyright

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

4.6.2 Macro Definition Documentation

4.6.2.1 M_PI

```
#define M_PI 3.14159265358979323846
```

Mathematical constant π.

Definition at line 18 of file test_park_inverse.cpp.

4.6.2.2 TEST_SIZE

```
#define TEST_SIZE 10
```

Number of values to test with.

Definition at line 15 of file test_park_inverse.cpp.

4.6.3 Function Documentation

4.6.3.1 main()

```
int main ( )
```

Main function of the C testbench.

The function Park_Inverse() will be called with the values of V_d and V_q in Vd and Vq and the results will be printed along with separately calculated values.

Definition at line 34 of file test_park_inverse.cpp.

```
34          {
35     hls::stream<int32_t> inputStream;
36     hls::stream<int32_t> outputStream;
37     int32_t tx_data, rx_data;
38     int16_t Vdlpha, Vqeta;
39     int16_t Theta;
40     float Vdf, Vqf, Thetaf;
41 
42 
43     Theta = 100;
44     for(int i=0; i<TEST_SIZE; i++){
45         tx_data = (int32_t(Vq[i]) << 16) | (int32_t(Vd[i]) & 0x0000FFFF);
46         inputStream << tx_data;
47 
48         Park_Inverse(inputStream, outputStream, Theta);
49 
50         outputStream.read(rx_data);
51         Vdlpha = int16_t(rx_data & 0xFFFF);
52         Vqeta = int16_t(rx_data >> 16);
53 
54         Thetaf = ((2*M_PI*2)/1000.0)*Theta;
55         Vdf = float(Vd[i])*cos(Thetaf) - float(Vq[i]) * sin(Thetaf);
56         Vqf = float(Vq[i])*cos(Thetaf) + float(Vd[i]) * sin(Thetaf);
57 
58         printf("Values is Vd=%d Vq=%d (%f %f)\n",Vdlpha, Vqeta, Vdf, Vqf);
59     }
60 }
```

4.6.4 Variable Documentation

4.6.4.1 Vd

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

Values of V_d to test Park_Inverse() with.

Definition at line 21 of file test_park_inverse.cpp.

4.6.4.2 Vq

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

Values of V_q to test Park_Inverse() with.

Definition at line 24 of file test_park_inverse.cpp.

Index

cos_table
    sin_cos_table.h, 16

doxygen/src/Park_Inverse_csynth.dox, 11
doxygen/src/main_page.dox, 11

M_PI
    test_park_inverse.cpp, 17
MAX_LIM
    park_inverse.h, 14
MIN_LIM
    park_inverse.h, 14
main
    test_park_inverse.cpp, 18

Park_Inverse
    park_inverse.cpp, 12
    park_inverse.h, 14
park_inverse.cpp, 11
    Park_Inverse, 12
park_inverse.h, 13
    MAX_LIM, 14
    MIN_LIM, 14
    Park_Inverse, 14

sin_cos_table.h, 16
    cos_table, 16
    sin_table, 16
sin_table
    sin_cos_table.h, 16

TEST_SIZE
    test_park_inverse.cpp, 18
test_park_inverse.cpp, 16
    M_PI, 17
    main, 18
    TEST_SIZE, 18
    Vd, 19
    Vq, 19

Vd
    test_park_inverse.cpp, 19
Vq
    test_park_inverse.cpp, 19