Clarke and Inverse ClarkeTransformations Hardware Implementation

User Guide

Microsemi

Table of Contents

Clarke and Inverse Clarke Transformations Theory

The behavior of three-phase machines is usually described by the machines voltage and current equations. The coefficients of the differential equations that describe their behavior are time varying (except when the rotor is stationary). The mathematical modeling of such a system tends to be complex since the flux linkages, induced voltages, and currents change continuously as the electric circuit is in relative motion. For such a complex electrical machine analysis, mathematical transformations are often used to decouple variables and to solve equations involving time varying quantities by referring all variables to a common frame of reference.

Clarke Transformation

The three-phase quantities are translated from the three-phase reference frame to the two-axis orthogonal stationary reference frame using Clarke Transformation as shown in Figure 1 on page 6.

The Clarke Transformation is expressed by the following equations:

$$I_\alpha = \frac{2}{3}(I_a) - \frac{1}{3}(I_b - I_c)$$

EQ1

$$I_\beta = \frac{2}{\sqrt{3}}(I_b - I_c)$$

EQ2

where,

I_a, I_b, and I_c are three-phase quantities

I_α and I_β are stationary orthogonal reference frame quantities

When I_α is superposed with I_a, and $I_a + I_b + I_c$ is zero, I_a, I_b, and I_c can be transformed to I_α and I_β as:

$$I_\alpha = I_a$$

EQ3

$$I_\beta = \frac{1}{\sqrt{3}}(I_a + 2I_b)$$

EQ4

where, $I_a + I_b + I_c = 0$

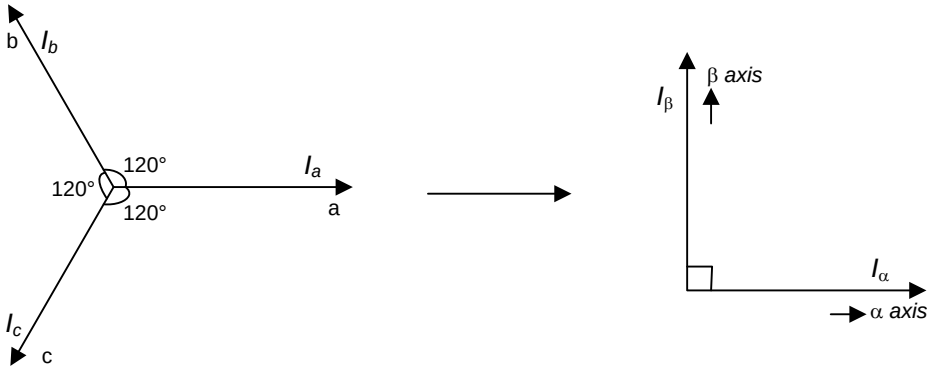

Figure 1 · Clarke Transformation

Inverse Clarke Transformation

The transformation from a two-axis orthogonal stationary reference frame to a three-phase stationary reference frame is accomplished using Inverse Clarke Transformation as shown in Figure 2. The Inverse Clarke Transformation is expressed by the following equations:

$$V_a = V_\alpha \quad \text{EQ5}$$

$$V_b = \frac{-V_\alpha + \sqrt{3} * V_\beta}{2} \quad \text{EQ6}$$

$$V_c = \frac{-V_\alpha - \sqrt{3} * V_\beta}{2} \quad \text{EQ7}$$

where,

V_a,V_b, and V_c are three-phase quantities

V_α, and V_β are stationary orthogonal reference frame quantities

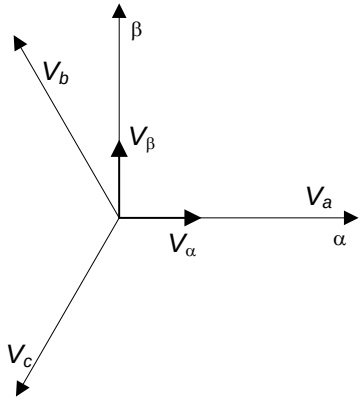

Figure 2 · Inverse Clarke Transformation

Clarke Transformation Hardware Implementation

This section describes the hardware implementation and the internal configuration details of the Clarke Transformation implemented on the SmartFusion2® device.

Clarke Transformation Implementation

The system level block diagram of the Clarke Transformation implemented is shown in Figure 3.

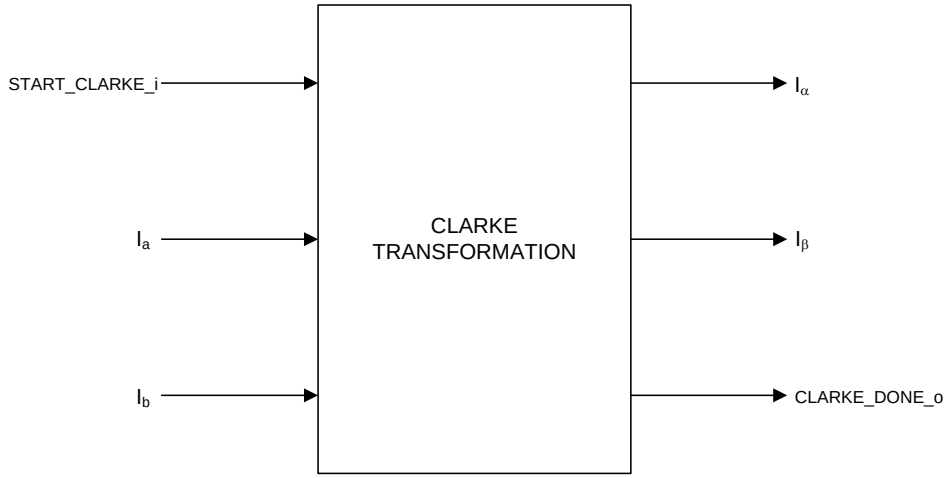

Figure 3 · System Level Block Diagram of Clarke Transformation

$$I_\alpha = I_a$$

EQ8

$$I_\beta = \frac{1}{\sqrt{3}}(I_a + 2I_b)$$

EQ9

where,

I_α and I_β are orthogonal stationary reference frame current components

I_a and I_b are input phase currents

The implementation of Clarke Transformation equations is done as shown in Figure 4. The Clarke Transformation block uses the MAS block, which performs some basic operations like multiplication, addition, and subtraction, for the computation of EQ8 and EQ9.

The START_CLARKE_i signal must undergo a Low to High transition to accept new inputs and compute the corresponding output. The CLARKE_DONE_o output signal goes High when the computations are completed and output is obtained. Once a set of inputs are given and the transformation process begins, new input will not be accepted before the CLARKE_DONE_o output signal goes High, even if the START_CLARKE_i signal undergoes a Low to High transition.

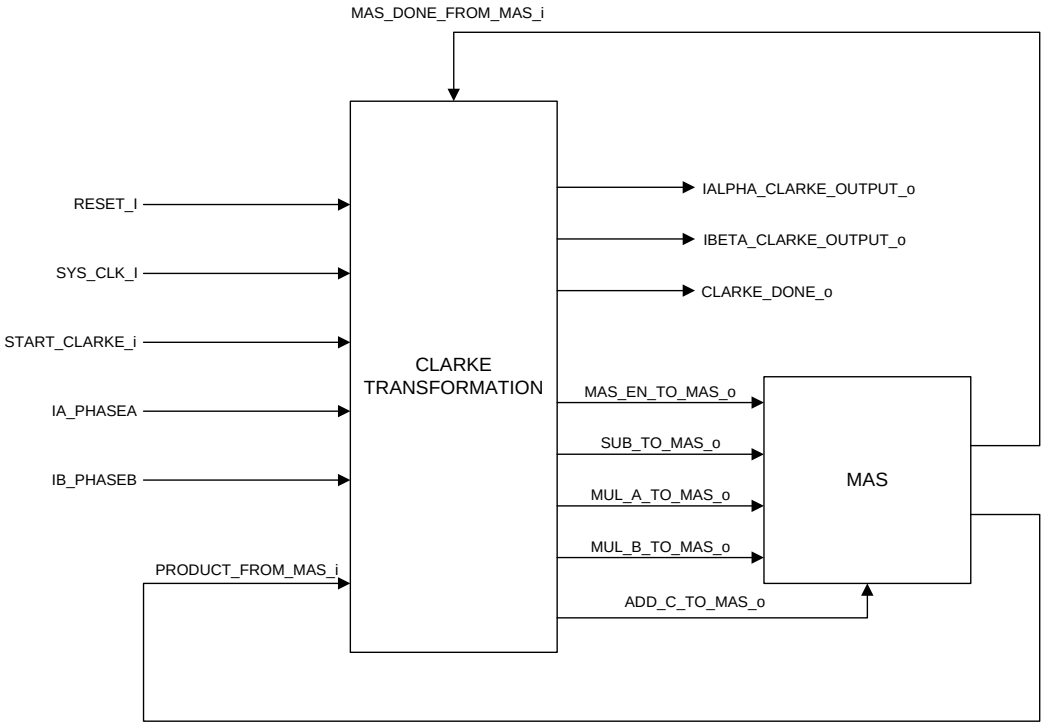

Figure 4 · Clarke Transformation Implementation

The inputs, IA_PHASEA and IB_PHASEB are obtained from the current measurement block that interfaces to an external analog to digital converter (ADC) on board.

Inputs and Outputs of Clarke Transformation Block

Table 1 lists and describes the input and output ports of Clarke Transformation block.

Table 1 · Input and Output Ports of Clarke Transformation

Signal Name	Direction	Description
RESET_I	Input	Asynchronous reset signal to design. Active state is defined by the generic g_RESET_STATE
SYS_CLK_I	Input	System clock
IA_PHASEA	Input	Phase A current component
IB_PHASEB	Input	Phase B current component
START_CLARKE_i	Input	Start signal for the Clarke Transformation block
MAS_DONE_FROM_MAS_i	Input	Done signal from the MAS block indicating that the computations by the MAS block are completed
PRODUCT_FROM_MAS_i	Input	Product from the MAS block
IALPHA_CLARKE_OUTPUT_o	Output	Current component in stationary orthogonal reference frame on alpha axis
IBETA_CLARKE_OUTPUT_o	Output	Current component in stationary orthogonal reference frame on beta axis

Signal Name	Direction	Description
CLARKE_DONE_o	Output	Signal indicating that the Clarke Transformation is complete
MAS_EN_TO_MAS_o	Output	Enable signal to the MAS block
SUB_TO_MAS_o	Output	Signal to give an indication to the MAS block to perform subtraction (when '1' => subtraction; 0 => addition)
MUL_A_TO_MAS_o	Output	Operand to the MAS block for multiplication
MUL_B_TO_MAS_o	Output	Operand to the MAS block for multiplication
ADD_C_TO_MAS_o	Output	Carry input to the MAS block

Configuration Parameters of Clarke Transformation Block

Table 2 lists and describes the configuration parameters used in the hardware implementation of the Clarke Transformation block. These are generic parameters and can be varied as per the requirement of the application.

Table 2 · Configuration Parameters of Clarke Transformation Block

Name	Description
g_RESET_STATE	When 0, supports active Low reset When 1, supports active High reset
g_IA_IB_WIDTH	Defines the bit length of the IA_PHASEA and IB_PHASEB registers
MUL_A_WIDTH	Defines the bit length of one of the operands to the MAS block for multiplication
MUL_B_WIDTH	Defines the bit length of one of the operands to the MAS block for multiplication
ADD_C_WIDTH	Defines the bit length of carry input to the MAS block

Clarke Transformation Block FSM Implementation

The finite state machine (FSM) of the Clarke Transformation block is shown in Figure 5.

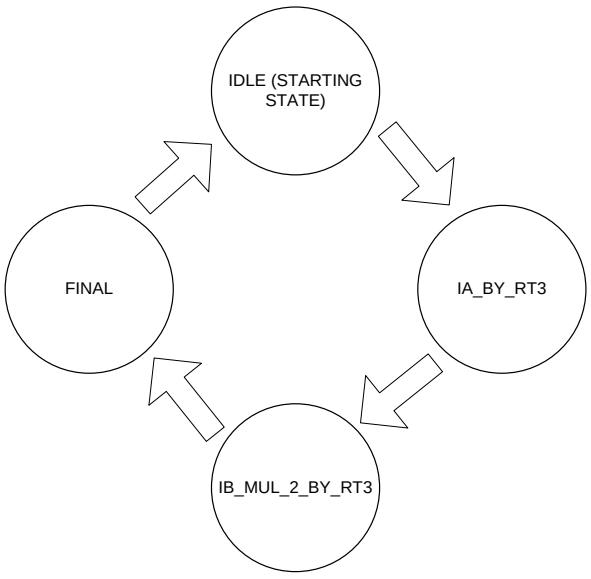

Figure 5 · Clarke Transformation FSM

There are four states in the FSM of the Clarke Transformation block namely:

- Idle
- IA_BY_RT3
- IB_MUL_2_BY_RT3
- FINAL

The FSM is synchronized to the rising-edge of the clock.

Idle State

This is the initial state of the FSM. The FSM moves to this state when a reset signal is given to the system or when the computations corresponding to the given inputs are completed and output is obtained. The FSM moves to IA_BY_RT3 state in the next clock cycle, when a rising-edge on the START_CLARKE_i input signal is detected.

IA_BY_RT3 State

In this state, the MAS block is enabled, and IA_PHASEA and 1/√3 (it is taken as constant) are given to the MAS block for multiplication. IALPHA_CLARKE_OUTPUT_o is assigned the value of IA_PHASEA in this state. The FSM moves to IB_MUL_2_BY_RT3 state in the next clock cycle.

IB_MUL_2_BY_RT3 State

The FSM remains in this state until the Done signal (MAS_DONE_FROM_MAS_i) of the MAS block goes High, indicating that the computation of previous state is completed. After the Done signal from the MAS block goes High, scaled value of IB_PHASEB (left shift by 1 bit, that is, scaled by 2) and 1/√3 are given to the MAS block for multiplication. The product obtained in the previous state is given as carry in ADD_C_TO_MAS_o to the MAS block for addition. The FSM moves to the FINAL state in the next clock cycle.

FINAL State

The FSM remains in this state until the Done signal of the MAS block goes High, indicating that the computation of the previous state is completed. After the Done signal of the MAS block goes High, IBETA_CLARKE_OUTPUT_o is assigned the product from the MAS block in this state, the CLARKE_DONE_o signal is made High (reflected in the next clock cycle) indicating that the Clarke Transformation is completed as shown in Figure 6 on page 11. The FSM moves to the Idle state in the next clock cycle.

Timing Diagram of Clarke Transformation Block

The timing waveform of the Clarke Transformation block is as shown in Figure 6.

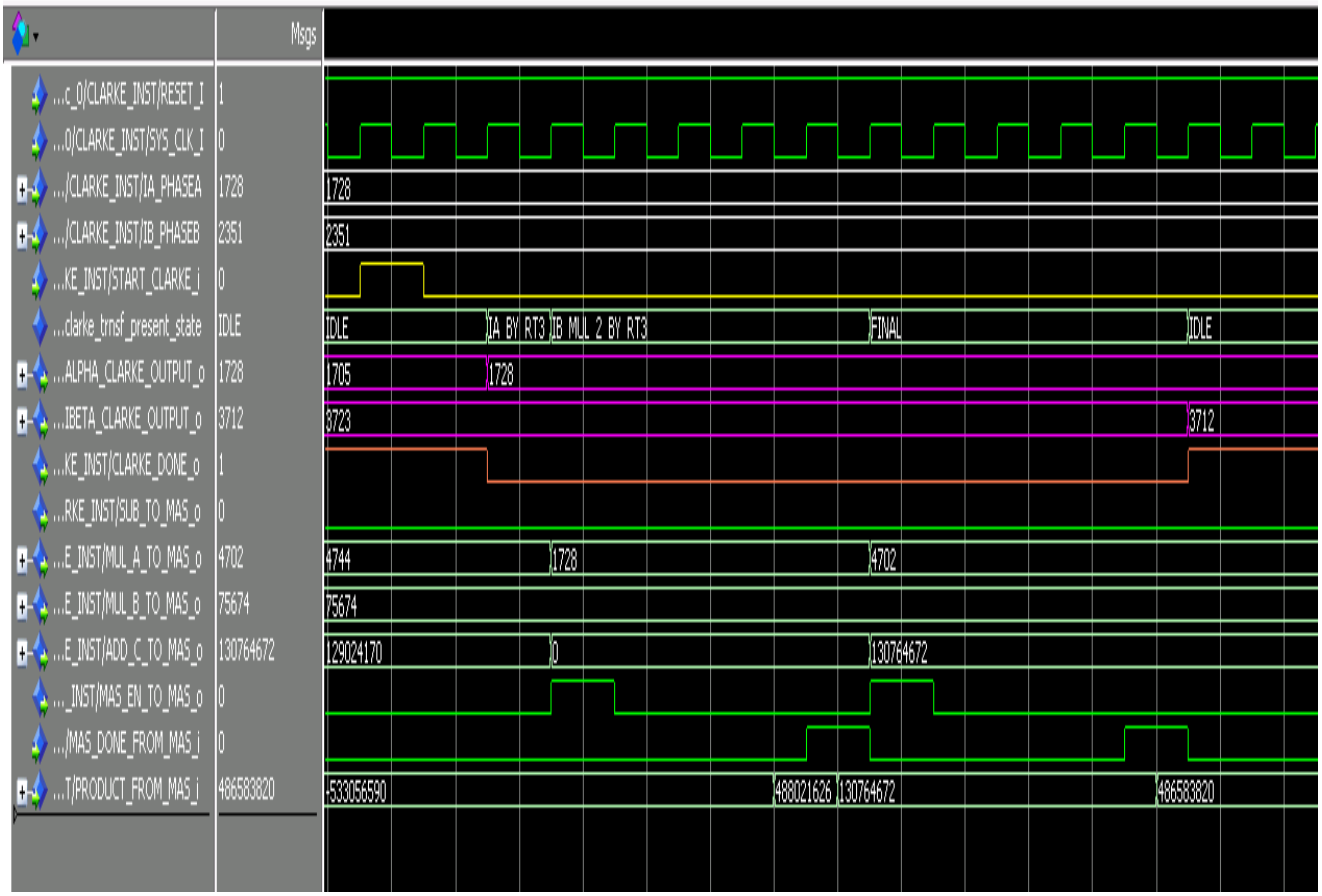

Figure 6 · Clarke Transformation Timing Diagram

Color Code:

- Yellow: START_CLARKE_i
- White: IA_PHASEA , IB_PHASEB (inputs)
- Purple: IALPHA_CLARKE_OUTPUT_o, IBETA_CLARKE_OUTPUT_o (outputs)
- Brown : CLARKE_DONE_o

Resource Utilization of Clarke Transformation Block

The resource utilization of Clarke Transformation implemented on the SmartFusion2 device is shown in Table 3.

Table 3 · Resource Utilization of Clarke Transformation Block

Resource Usage Report for Clarke Transformation Block	
Cell Usage	**Description**
CLKINT	2 uses
CFG2	45 uses
CFG3	21 uses
CFG4	2 uses
Sequential Cells	
SLE	119 uses
Registers not packed on I/O Pads	119
DSP Blocks	0
I/O Ports	189
I/O Primitives	189
INBUF	76 uses
OUTBUF	113 uses
Global Clock Buffers	2
Total LUTs	68

Inverse Clarke Transformation Hardware Implementation

This section describes the hardware implementation and the internal configuration details of the Inverse Clarke Transformation implemented on the SmartFusion2 device.

Inverse Clarke Transformation Implementation

The system level block diagram of the Inverse Clarke Transformation implemented is shown in Figure 7.

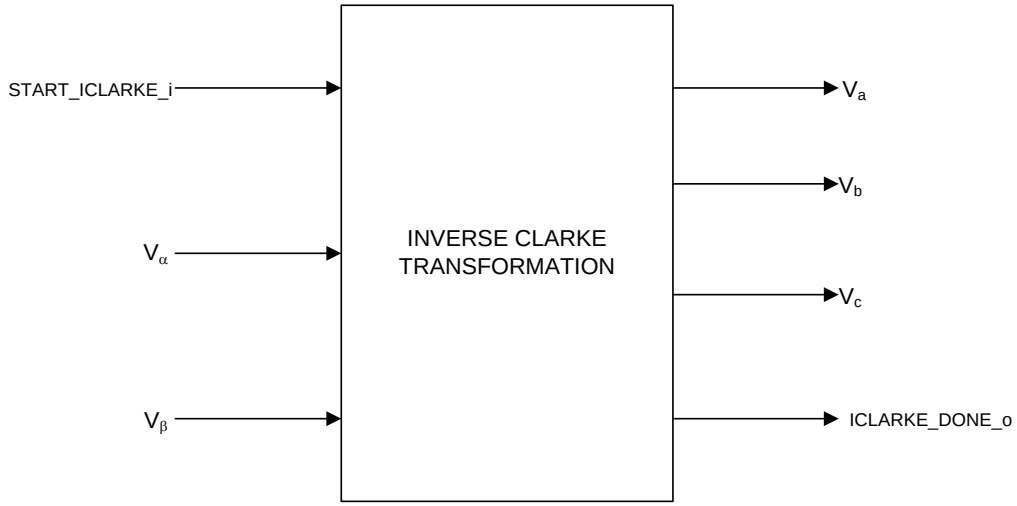

Figure 7 · System Level Block Diagram of Inverse Clarke Transformation

The above block implements the following equations:

$$V_a = V_\alpha$$

EQ10

$$V_b = \frac{-V_\alpha + \sqrt{3} * V_\beta}{2}$$

EQ11

$$V_c = \frac{-V_\alpha - \sqrt{3} * V_\beta}{2}$$

EQ12

where,

V_a, V_b, and V_c are three-phase quantities

V_α and V_β are stationary orthogonal reference frame quantities

The Inverse Clarke Transformation equations are implemented as shown in Figure 8 on page 14. The Inverse Clarke Transformation block uses the MAS block, which performs the basic operations like multiplication, addition, and subtraction, for the computation of EQ10 and EQ11.

The START_ICLARKE_i signal must undergo a Low to High transition to accept new inputs and compute the output. The ICLARKE_DONE_o goes High when the computations are completed and output is obtained. Once a set of

inputs are given and the transformation process begins, new input will not be accepted before the ICLARKE_DONE_o output signal goes High, even if the START_ICLARKE_i signal undergoes a Low to High transition

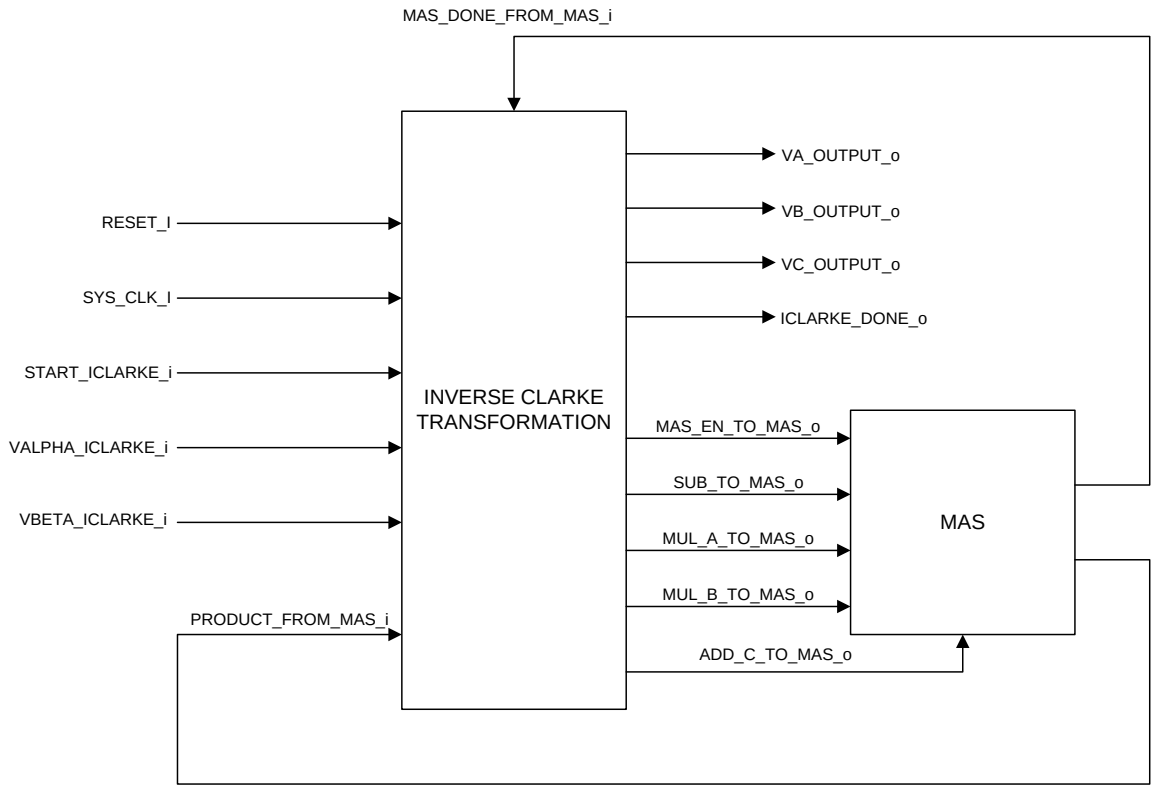

Figure 8 · Inverse Clarke Transformation Implementation

The inputs, VALPHA_ICLARKE_i and VBETA_ICLARKE_i are obtained from the Inverse Park Transformation block.

Inputs and Outputs of Inverse Clarke Transformation Block

Table 4 describes the input and output ports of the Inverse Clarke Transformation block.

Table 4 · Input and Output Ports of Inverse Clarke Transformation

Signal Name	Direction	Description
RESET_I	Input	Asynchronous reset signal to design. Active state is defined by the generic g_RESET_STATE.
SYS_CLK_I	Input	System clock
VALPHA_ICLARKE_i	Input	Voltage component in stationary orthogonal reference frame (Valpha)
VBETA_ICLARKE_i	Input	Voltage component in stationary orthogonal reference frame (Vbeta)
START_ICLARKE_i	Input	Start signal for the Inverse Clarke function
VA_OUTPUT_o	Output	Phase A voltage component
VB_OUTPUT_o	Output	Phase B voltage component
VC_OUTPUT_o	Output	Phase C voltage component
ICLARKE_DONE_o	Output	Signal indicating that the Inverse Clarke Transformation is complete

Signal Name	Direction	Description
MAS_EN_TO_MAS_o	Output	Enable signal to the MAS block
SUB_TO_MAS_o	Output	Signal to give an indication to the MAS block to perform subtraction (when ‘1’ => subtraction; ‘0’ => addition).
MUL_A_TO_MAS_o	Output	Operand to the MAS block for multiplication
MUL_B_TO_MAS_o	Output	Operand to the MAS block for multiplication
ADD_C_TO_MAS_o	Output	Carry input to the MAS block

Configuration Parameters of Inverse Clarke Transformation Block

Table 5 lists and describes the configuration parameters used in the hardware implementation of the Inverse Clarke Transformation block.. These are generic parameters and can be varied as per the requirement of the application.

Table 5 · Configuration Parameters of Inverse Clarke Transformation Block

Name	Description
g_RESET_STATE	When 0, supports active Low reset When 1, supports active High reset
g_VALPHA_VBETA_WIIDTH	Defines the bit length of the VALPHA_ICLARKE_i and VBETA_ICLARKE_i registers
MUL_A_WIDTH	Defines the bit length of one of the operands to the MAS block for multiplication
MUL_B_WIDTH	Defines the bit length of one of the operands to the MAS block for multiplication
ADD_C_WIDTH	Defines the bit length of carry input to the MAS block

Inverse Clarke Transformation Block FSM Implementation

The FSM of the Inverse Clarke Transformation block is shown in Figure 9.

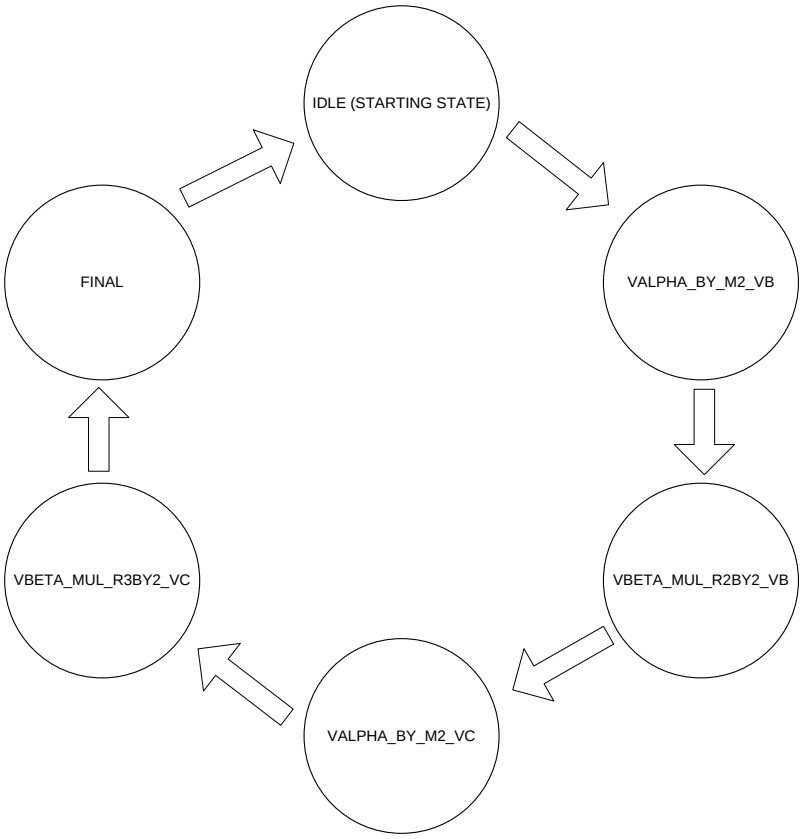

Figure 9 · Inverse Clarke Transformation FSM

Following are the six states in the FSM of the Inverse Clarke Transformation block:

Idle State

- VALPHA_BY_M2_VB
- VBETA_MUL_R3BY2_VB
- VALPHA_BY_M2_VC
- VBETA_MUL_R3BY2_VC
- FINAL

The FSM is synchronized to the rising-edge of the clock.

Idle State

This is the initial state of the FSM. The FSM moves to this state when a reset signal is given to the system or when the computations corresponding to the given inputs are completed and output is obtained. The FSM moves to the VALPHA_BY_M2_VB state in the next clock cycle, when a rising-edge on the START_ICLARKE_i input signal is detected.

VALPHA_BY_M2_VB State

In this state the MAS block is enabled and VALPHA_ICLARKE_i and (-1/2) (scaled and taken as constant) are given to the MAS block for multiplication. VA_OUTPUT_o is assigned the value of VALPHA_ICLARKE_i in this state. The FSM moves to the VBETA_MUL_R3BY2_VB state in the next clock cycle.

VBETA_MUL_R3BY2_VB State

The FSM remains in this state until the Done signal of the MAS block goes High, indicating that the computation of the previous state is complete. After the Done signal (MAS_DONE_FROM_MAS_i) from the MAS block goes High, VBETA_ICLARKE_i and (√3)/2 (scaled and taken as constant) are given to the MAS block for multiplication. The product obtained in the specify state is given as carry in (ADD_C_TO_MAS_o) to the MAS block for addition. The FSM moves to the VALPHA_BY_M2_VC state in the next clock cycle.

VALPHA_BY_M2_VC State

The FSM remains in this state until the Done signal of the MAS block goes High, indicating that the computation of the specify state is complete. After the Done signal from the MAS block goes High, VALPHA_ICLARKE_i and (-1/2) (scaled and taken as constant) are given to the MAS block for multiplication. The product obtained in the specify state is assigned to the VB_OUTPUT_o. The FSM moves to the VBETA_MUL_R3BY2_VC state in the next clock cycle.

VBETA_MUL_R3BY2_VC State

The FSM remains in this state until the Done signal of the MAS block goes High, indicating that the computation of the specify state is complete. After the Done signal from the MAS block goes High, VBETA_ICLARKE_i and (√3)/2 (scaled and taken as constant) are given to the MAS block for multiplication. The product obtained in the specify state is given as carry in for subtraction (SUB_TO_MAS_o ='1'). The FSM moves to the FINAL state in the next clock cycle.

FINAL State

The FSM remains in this state until the Done signal of the MAS block goes High, indicating that the computation of the specify state is complete. After the Done signal from the MAS block goes High, VC_OUPUT_o is assigned the product from the MAS block in this state and the ICLARKE_DONE_o signal is made High (reflected in the next clock cycle) indicating that the Inverse Clarke Transformation is completed as shown in Figure 10 on page 18. The FSM moves to the Idle state in the next clock cycle.

Timing Diagram of Inverse Clarke Transformation Block

The timing waveform of the Inverse Clarke Transformation block is shown in Figure 10.

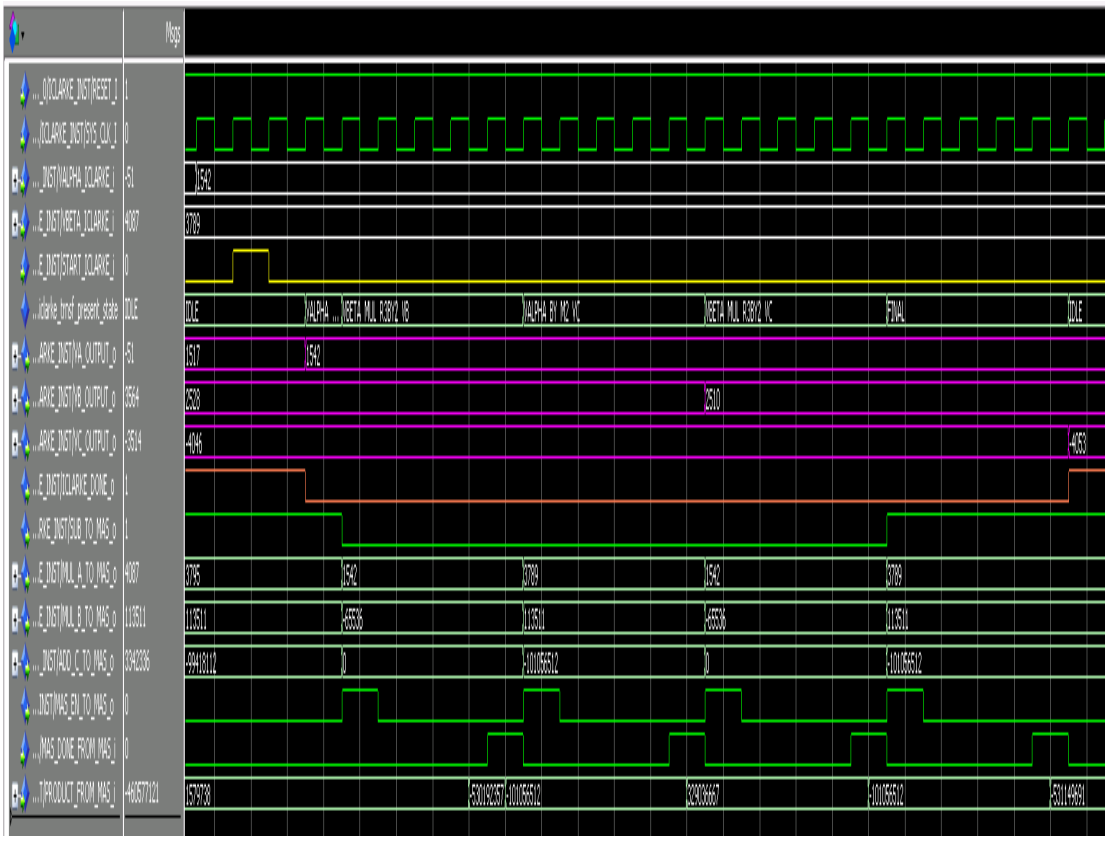

Figure 10 · Inverse Clarke Transformation Timing Diagram

Color code:

Yellow: START_ICLARKE_i

WHITE: VALPHA_ICLARKE_i, VBETA_ICLARKE_i (inputs)

Purple: VA_OUTPUT_o, VB_OUTPUT_o, VC_OUTPUT_o (outputs)

Brown: ICLARKE_DONE_o

Resource Utilization of Inverse Clarke Transformation Block

The resource utilization of Inverse Clarke Transformation implemented on the SmartFusion2 device is shown in Table 6.

Table 6 · Resource Utilization of Inverse Clarke Transformation Block

Resource Usage Report for Clarke Transformation Block	
Cell Usage	**Description**
CLKINT	2 uses
CFG2	45 uses
CFG3	21 uses
CFG4	2 uses
Sequential Cells	
SLE	119 uses
Registers not packed on I/O Pads	119
DSP Blocks	0
I/O Ports	189
I/O Primitives	189
INBUF	76 uses
OUTBUF	113 uses
Global Clock Buffers	2
Total LUTs	68

Appendix

<Add heading: Entity Definition of the Clarke Block >

The entity definition of the Clarke block is as follows:

```
ENTITY clarke IS
GENERIC(
g_RESET_STATE : STD_LOGIC := '0' ;
g_IA_IB_WIDTH : INTEGER := 14 ;
-- Multiplier with add/sub generics
MUL_A_WIDTH : INTEGER := 18 ;
MUL_B_WIDTH : INTEGER := 18 ;
ADD_C_WIDTH : INTEGER := 44
) ;
PORT (
RESET_I : IN STD_LOGIC ;
SYS_CLK_I : IN STD_LOGIC ;
START_CLARKE_i : IN STD_LOGIC ;
IA_PHASEA : IN STD_LOGIC_VECTOR((g_IA_IB_WIDTH-1) downto 0) ;
IB_PHASEB : IN STD_LOGIC_VECTOR((g_IA_IB_WIDTH-1) downto 0) ;
IALPHA_CLARKE_OUTPUT_o: OUT STD_LOGIC_VECTOR((g_IA_IB_WIDTH) downto 0) ;
IBETA_CLARKE_OUTPUT_o : OUT STD_LOGIC_VECTOR((g_IA_IB_WIDTH) downto 0) ;
CLARKE_DONE_o : OUT STD_LOGIC ;
-- MAS SIGNALS
MAS_EN_TO_MAS_o : OUT STD_LOGIC ;
SUB_TO_MAS_o : OUT STD_LOGIC ;
MUL_A_TO_MAS_o : OUT STD_LOGIC_VECTOR((MUL_A_WIDTH-1) downto 0) ;
MUL_B_TO_MAS_o : OUT STD_LOGIC_VECTOR((MUL_B_WIDTH-1) downto 0) ;
ADD_C_TO_MAS_o : OUT STD_LOGIC_VECTOR((ADD_C_WIDTH-1) downto 0) ;
PRODUCT_FROM_MAS_i : IN STD_LOGIC_VECTOR((ADD_C_WIDTH-1) downto 0) ;
MAS_DONE_FROM_MAS_i : IN STD_LOGIC
) ;
END Clarke
```

<Add heading: Entity Definition of the Inverse Clarke Block >

The entity definition of the Inverse Clarke block is as follows:

```
ENTITY Inverse_Clarke IS
GENERIC(
g_RESET_STATE : STD_LOGIC := '0' ;
g_VALPHA_VBETA_WIDTH : INTEGER := 18 ;
--g_VA_VB_VC_WIDTH : INTEGER := 18 ;
-- Multiplier with add/sub generics
MUL_A_WIDTH : INTEGER := 18 ;
MUL_B_WIDTH : INTEGER := 18 ;
ADD_C_WIDTH : INTEGER := 44
) ;
PORT (
RESET_I : IN STD_LOGIC ;
```

```
SYS_CLK_I : IN STD_LOGIC ;
START_ICLARKE_i : IN STD_LOGIC ;
VALPHA_ICLARKE_i : IN STD_LOGIC_VECTOR((g_VALPHA_VBETA_WIDTH-1) downto 0) ;
VBETA_ICLARKE_i : IN STD_LOGIC_VECTOR((g_VALPHA_VBETA_WIDTH-1) downto 0) ;
VA_OUTPUT_o : OUT STD_LOGIC_VECTOR((g_VALPHA_VBETA_WIDTH) downto 0) ;
VB_OUTPUT_o : OUT STD_LOGIC_VECTOR((g_VALPHA_VBETA_WIDTH) downto 0) ;
VC_OUTPUT_o : OUT STD_LOGIC_VECTOR((g_VALPHA_VBETA_WIDTH) downto 0) ;
ICLARKE_DONE_o : OUT STD_LOGIC ;
-- MAS SIGNALS
MAS_EN_TO_MAS_o : OUT STD_LOGIC ;
SUB_TO_MAS_o : OUT STD_LOGIC ;
MUL_A_TO_MAS_o : OUT STD_LOGIC_VECTOR((MUL_A_WIDTH-1) downto 0) ;
MUL_B_TO_MAS_o : OUT STD_LOGIC_VECTOR((MUL_B_WIDTH-1) downto 0) ;
ADD_C_TO_MAS_o : OUT STD_LOGIC_VECTOR((ADD_C_WIDTH-1) downto 0) ;
PRODUCT_FROM_MAS_i : IN STD_LOGIC_VECTOR((ADD_C_WIDTH-1) downto 0) ;
MAS_DONE_FROM_MAS_i : IN STD_LOGIC
) ;
END Inverse_Clarke
```

Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call **800.262.1060**
From the rest of the world, call **650.318.4460**
Fax, from anywhere in the world **408.643.6913**

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Microsemi SoC Products Group Customer Support website for more information and support (http://www.microsemi.com/soc/support/search/default.aspx). Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on website.

Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at http://www.microsemi.com/soc/.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

50200502-0/11.13