Morphologic
Embedded FPGA Solutions
www.morphologic.dk

XPS Graphics Engine (v2.00b)

Introduction

The XPS_GRAPHICS_ENGINE_V2_00_b IP core is 2D graphics rendering engine that interfaces to a PLB V4.6 bus for transfer of image data (master attachment) and for register access (slave attachment). The graphics engine uses a combination of dedicated graphics acceleration logic, a small footprint coprocessor and a PLB bus master attachment to render graphics. The host writes commands and parameters to the coprocessor which then executes the commands using the dedicated hardware units. This relieves the host CPU from the job of performing low level graphics operations, releasing time for other higher level functions.

This document provides a description of the functional blocks, signals and configuration parameters of the cores. It also explains the registers available for programming the core as well as a set of procedures for programming the core.

Features

- Supports PLBv4.6 bus
- Accelerates line and pixel plot operations.
- Accelerates bit block transfer operations.
- Support background masking and alpha blending.
- Small footprint due to optimisation of the combination of hardware and micro coded operations.
- Upgradeable/extensible microcode via user port
- Configurable small footprint core
- Supports 16 bits/pixel graphics mode.

Core short specs.	
Supported Device Family	Spartan-3 Spartan-6 Virtex-4 Virtex-5 Virtex-6
Core Version	V1_00_a
Resource utilization	
Slices	
Block RAMs	
Special Features	none
Core deliveries	
Documentation	Data Sheet
Design Files	VHDL & Netlist
Constraints File	N/A
Verification	On request
Instantiation Template	N/A
Ref. Design & Application Notes	N/A
Design Tool Requirements	
Xilinx Implementation Tools	ISE® 11.2 or later
Verification	Xilinx BFL tool set
Simulation	Modelsim SE/PE 6.5 or later
Synthesis	XST
Support	
Provided by Morphologic ApS	

1. Functional Description

1.1 Overview

The core contains two hardware blocks dedicated to pixel plotting and bit block transfers. The hardware engine requests bus operations to the PLB bus master DMA engine and reads/writes pixels via small FIFOs. Bit block transfer operations are always executed using bursts of 8 32-bit transfers.

The core also contains a small footprint graphics engine CPU (GECPU), which receives commands and parameters over the command FIFO. The GECPU breaks the graphics operations into bit plot and bitblt primitive operations and configures the relevant hardware block to execute the sub-operations needed to complete the graphic command. The GECPU can utilize the time it takes for the hardware engines and bus interface to complete the operation, to prepare the next primitive operation. This split between hardware and software reduces the amount of logic needed while providing good performance and flexibility.

The GECPU uses a single 18Kbit block RAM to store it's instructions. The block RAM can be accessed indirectly using the core control registers so that it's contents can be changed. The block RAM is preloaded with the default software upon configuration of the FPGA.

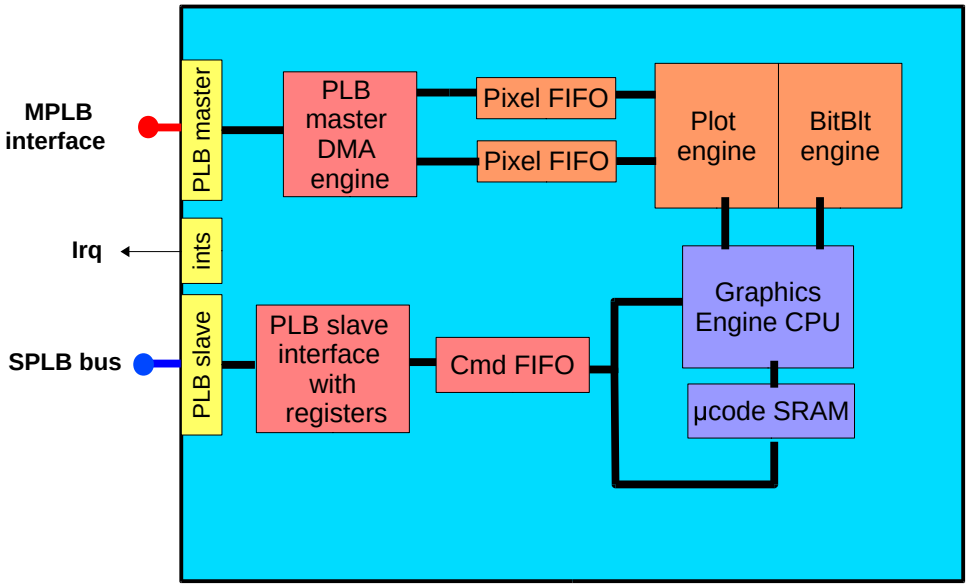

Fig 1. XPS_GRAPHICS_ENGINE IP block diagram.

1.2 PLB Master (MPLB) interface

The PLB bus master interface is used for accessing both on-screen and off-screen memory. The MPLB master uses a native 32-bit bus format, but will attach to 32, 64 or 128 bit buses. Both single and 8-word burst read and writes are issued on this interface.

1.3 PLB Slave (SPLB) Interface

The PLB slave uses a native 32-bit bus format, but will attach to 32, 64 or 128 bit buses. A block of 4 configuration registers are available through the interface. To simplify logic, the PLB slave only supports 32-bit read and writes. The PLB slave interface must use the same clock as the bus master port.

1.4 Plot engine

The plot engine is used for drawing single pixels. Depending on the drawing mode a single write or a read-modify-write sequence is issued via the MPLB interface. The plot engine receives the coordinates and the current drawing mode and colour from the GECPU. Typically the GECPU can use the time it takes for the pixel to be written to prepare the next coordinates. The plot engine is used for plotting single pixels as well as lines, ellipses, rectangles etc.

1.5 Bit block transfer (BitBlt) engine

The BitBlt engine is used for transferring rectangular blocks of pixels from one location to another location. The engine transfers n pixels from the source address to the destination address. The engine will always perform a read-modify write on the destination and a read operation from the source. All transfers are done using bursts of 8 32-bit words, which means that up to 16 pixels are read/written at a time. Since the engine only transfers data one row at a time, the GECPU is used for calculating source and destination addresses for each row transferred. The engine can perform alpha blending between the source and destination pixels. The alpha factor is programmed as a 5-bit value in unsigned 1.4 format. Alpha blending is performed using the following equation:

P = Ps * alpha + Pd * (1-alpha),

where Ps = source pixel, Pd = destination pixel and P = new pixel written at destination.

If no blending is wanted, the alpha value must be set to 1.0 (0x10). The calculation is performed on each colour component of the pixels (2 pixels are process simultaneously) and uses 4 multipliers.

The engine can also perform colour keying: When the source pixels are read, the pixels are checked for the special key colour (normally 100% green) and if found the destination pixel will not be written. If not found the pixel is written as normal, optionally using alpha blending.

1.6 Graphics Engine CPU (GECPU)

The CPU used is the KCSMP3 or Picoblaze. This is a small footprint 8-bit RISC architecture processor using a single block RAM for instructions. The GECPU receives graphics instructions from the command FIFO and responds accordingly by either just storing new mode settings or by starting the plot or BitBlt hardware engines. The GECPU will wait for the engines to be ready before starting new operations. A list of supported graphics commands is given in table 1 in section 5.

2. Core Interface Signals

2.1 Xilinx v4.6 PLB Slave Bus Signals

Signal Name	*Direction*	*Type*	*Description*
SPLB_Clk	I	std_logic	PLB main bus clock
SPLB_Rst	I	std_logic	PLB main bus reset
PLB_ABus	I	std_logic_vector(0 to 31)	PLB address bus
PLB_UABus	I	std_logic_vector(0 to 31)	PLB upper address bus
PLB_PAValid	I	std_logic	PLB primary address valid indicator
PLB_SAValid	I	std_logic	PLB secondary address valid indicator
PLB_rdPrim	I	std_logic	PLB secondary to primary read request indicator
PLB_wrPrim	I	std_logic	PLB secondary to primary write request indicator
PLB_masterID	I	std_logic_vector(0 to C_SPLB_MID_WIDTH-1)	PLB current master identifier
PLB_abort	I	std_logic	PLB abort request indicator
PLB_busLock	I	std_logic	PLB bus lock
PLB_RNW	I	std_logic	PLB read/not write
PLB_BE	I	std_logic_vector(0 to C_SPLB_DWIDTH/8-1)	PLB byte enables
PLB_MSize	I	std_logic_vector(0 to 1)	PLB master data bus size
PLB_size	I	std_logic_vector(0 to 3)	PLB transfer size
PLB_type	I	std_logic_vector(0 to 2)	PLB transfer type
PLB_lockErr	I	std_logic	PLB lock error indicator
PLB_wrDBus	I	std_logic_vector(0 to C_SPLB_DWIDTH-1)	PLB write data bus
PLB_wrBurst	I	std_logic	PLB burst write transfer indicator
PLB_rdBurst	I	std_logic	PLB burst read transfer indicator
PLB_wrPendReq	I	std_logic	PLB write pending bus request indicator
PLB_rdPendReq	I	std_logic	PLB read pending bus request indicator
PLB_wrPendPri	I	std_logic_vector(0 to 1)	PLB write pending request priority
PLB_rdPendPri	I	std_logic_vector(0 to 1)	PLB read pending request priority
PLB_reqPri	I	std_logic_vector(0 to 1)	PLB current request priority
PLB_TAttribute	I	std_logic_vector(0 to 15)	PLB transfer attribute
Sl_addrAck	O	std_logic	Slave address acknowledge
Sl_SSize	O	std_logic_vector(0 to 1)	Slave data bus size

Sl_wait	O	std_logic	Slave wait indicator
Sl_rearbitrate	O	std_logic	Slave re-arbitrate bus indicator
Sl_wrDAck	O	std_logic	Slave write data acknowledge
Sl_wrComp	O	std_logic	Slave write transfer complete indicator
Sl_wrBTerm	O	std_logic	Slave terminate write burst transfer
Sl_rdDBus	O	std_logic_vector(0 to C_SPLB_DWIDTH-1)	Slave read data bus
Sl_rdWdAddr	O	std_logic_vector(0 to 3)	Slave read word address
Sl_rdDAck	O	std_logic	Slave read data acknowledge
Sl_rdComp	O	std_logic	Slave read transfer complete indicator
Sl_rdBTerm	O	std_logic	Slave terminate read burst transfer
Sl_MBusy	O	std_logic_vector(0 to C_SPLB_NUM_MASTERS -1)	Slave busy indicator
Sl_MWrErr	O	std_logic_vector(0 to C_SPLB_NUM_MASTERS -1)	Slave write error indicator
Sl_MRdErr	O	std_logic_vector(0 to C_SPLB_NUM_MASTERS -1)	Slave read error indicator
Sl_MIRQ	O	std_logic_vector(0 to C_SPLB_NUM_MASTERS -1)	Slave interrupt indicator

2.2 Xilinx v4.6 PLB Master Bus Signals

Signal Name	*Direction*	*Type*	*Description*
MPLB_Clk	I	std_logic	PLB main bus Clock
MPLB_Rst	I	std_logic	PLB main bus Reset
M_request	O	std_logic	Master request
M_priority	O	std_logic_vector(0 to 1)	Master request priority
M_busLock	O	std_logic	Master buslock
M_RNW	O	std_logic	Master read/nor write
M_BE	O	std_logic_vector(0 to C_MPLB_DWIDTH/8-1)	Master byte enables
M_MSize	O	std_logic_vector(0 to 1)	Master data bus size
M_size	O	std_logic_vector(0 to 3)	Master transfer size
M_type	O	std_logic_vector(0 to 2)	Master transfer type

M_TAttribute	O	std_logic_vector(0 to 15)	Master transfer attribute
M_lockErr	O	std_logic	Master lock error indicator
M_abort	O	std_logic	Master abort bus request indicator
M_UABus	O	std_logic_vector(0 to 31)	Master upper address bus (unused)
M_ABus	O	std_logic_vector(0 to 31)	Master address bus
M_wrDBus	O	std_logic_vector(0 to C_MPLB_DWIDTH-1)	Master write data bus
M_wrBurst	O	std_logic	Master burst write transfer indicator
M_rdBurst	O	std_logic	Master burst read transfer indicator
PLB_MAddrAck	I	std_logic	PLB reply to master for address acknowledge
PLB_MSSize	I	std_logic_vector(0 to 1)	PLB reply to master for slave data bus size
PLB_MRearbitrate	I	std_logic	PLB reply to master for bus re-arbitrate indicator
PLB_MTimeout	I	std_logic	PLB reply to master for bus time out indicator
PLB_MBusy	I	std_logic	PLB reply to master for slave busy indicator
PLB_MRdErr	I	std_logic	PLB reply to master for slave read error indicator
PLB_MWrErr	I	std_logic	PLB reply to master for slave write error indicator
PLB_MIRQ	I	std_logic	PLB reply to master for slave interrupt indicator
PLB_MRdDBus	I	std_logic_vector(0 to (C_MPLB_DWIDTH-1))	PLB reply to master for read data bus
PLB_MRdWdAddr	I	std_logic_vector(0 to 3)	PLB reply to master for read word address
PLB_MRdDAck	I	std_logic	PLB reply to master for read data acknowledge
PLB_MRdBTerm	I	std_logic	PLB reply to master for terminate read burst indicator
PLB_MWrDAck	I	std_logic	PLB reply to master for write data acknowledge
PLB_MWrBTerm	I	std_logic	PLB reply to master for terminate write burst indicator

System signals:

Signal Name	***Direction***	***Type***	***Description***

IP2INTC_Irpt	O	std_logic	Active high interrupt line.

3. Core Parameters

Several generics allow customisation of the IP core to suit the type of display being attached to the IP signals, the pixel bitmap format, as well as the back-light and contrast control circuitry if any. Using generics allows for a smaller footprint core, leaving more room for other functions in the FPGA.

XPS_GRAPHICS_ENGINE Parameters

Generic name	***type***	***Default value***	***Description***
C_COLOR_KEY	std_logic_vector	11111100000	Selects the colour used for colour keying

System and PLB parameters.

Name	***Type***	***Allowable Values***	***Default value***	***Description***
C_BASEADDR	std_logic_vector	Lower 6 bits must be '0'.	None	Slave register base address.
C_HIGHADDR	std_logic_vector	Lower 6 bits must be '1'.	None	Slave register high address.
C_SPLB_AWIDTH	integer	32	32	PLB Address Bus Width
C_SPLB_DWIDTH	integer	32, 64, 128	32	PLB Data Bus Width
C_SPLB_NUM_MASTERS	integer	1-16	8	Number of PLB masters
C_SPLB_MID_WIDTH	integer	log2(C_SPLB_NUM_MASTERS) with a minimum value of 1	1	Width of the PLB_master ID vector
C_SPLB_NATIVE_DWIDTH	integer	32	32	Width of slave data bus
C_SPLB_P2P	integer	0 : Shared bus topology 1 : Point to Point topology	0	PLB Point-to-Point or shared bus topology
C_SPLB_SUPPORT_BURSTS	integer	0:SPLB does not support bursts 1:SPLB supports bursts	0	unused
C_SPLB_SMALLEST_MASTER	integer	32, 64, 128	32	Width of the smallest master that will be interacting with this slave. Unused
C_SPLB_CLK_PERIOD_PS	Integer	Any positive value	10000	SPLB clock period in picoseconds. Unused
C_INCLUDE_DPHASE_TIMER	Integer	0:do include dataphase timer 1:include dataphase timer	1	Select whether a data phase timer in the slave interface is included or not.

C_FAMILY	String	spartan3e, spartan3a, spartan6, virtex4, virtex5, virtex6	virtex5	XILINX FPGA Family
C_MPLB_AWIDTH	Integer	32	32	Width of master address bus
C_MPLB_DWIDTH	Integer	32, 64, 128	32	Width of master data bus
C_MPLB_NATIVE_DWIDTH	Integer	32	32	Native width of IP master bus
C_MPLB_P2P	Integer	0 : Shared bus topology 1 : Point to Point topology	0	Master bus topology
C_MPLB_SMALLEST_SLAVE	Integer	32,64,128	32	Width of the smallest slave attached to the master bus
C_MPLB_CLK_PERIOD_PS	Integer	Any positive value	10000	MPLB clock period in picoseconds

4. Programming model

The following sections describe the programming model of the IP core. First a register map is given followed by a detailed description of the individual registers. Note that all bits are normally active high, unless explicitly otherwise mentioned.

4.1 Register map

All of the registers listed below must be accessed as 32-bit wide registers. All fields marked as reserved are read as 0 and written bits are ignored.

Register Name	*Offset from base address (hex)*
GE_CTRL / GE_STATUS	0x00
GE_FIFO	0x04
GE_UC_ADDR	0x08
GE_UC_DATA	0x0C

4.1.1 GE_CTRL / GE_STATUS (offset 0x0)

Name	*Bit*	*Type*	*Reset value*	*Description*
GE_STAT_FIFO_EMPTY	[0]	RO	1	Command FIFO empty status
GE_STAT_FULL	[1]	RO	0	Command FIFO full status
GE_STAT_IRQ	[2]	RO	0	State of the interrupt line
GE_STAT_BUSY	[3]	RO	0	Graphics engine and associated hardware is executing commands
GE_STAT_FIFO_OCCUPANCY	[8:4]	RO	O	Number of words in the command FIFO
Reserved	[15:9]	RO	0	Reserved read as zero

Name	Bit	Type	Reset value	Description
GE_CTRL_FIFO_THRESHOLD	[20:16]	R/W	0	When then the number of words in the command FIFO is less than or equal to the programmed threshold, the interrupt line will be asserted.
Reserved	[27:21]	R/W	0	Ignored
GE_CTRL_RDY_IRQEN	28	R/W	0	Enables the GE completion interrupt
GE_CTRL_FIFO_IRQEN	29	R/W	0	Enables the FIFO level interrupt
GE_CTRL_RESET	30	R/W	0	Command FIFO reset
GE_CTRL_RUN	31	R/W	0	Active low GECPU reset

4.1.2 GE_FIFO (offset 0x4)

Name	Bit	Type	Reset value	Description
GE_FIFO	[31:0]	WO	0	Write port for the GE command FIFO

4.1.3 GE_UC_ADDR (offset 0x8)

Name	Bit	Type	Reset value	Description
GE_UC_ADDR	[9:0]	R/W	0	Current GECPU microcode RAM address
Reserved	[31:10]	RO	0	Reserved

4.1.4 GE_UC_DATA (offset 0x0C)

Name	Bit	Type	Reset value	Description
GE_UC_DATA	[17:0]	R/W	X	GECPU instruction RAM data port. When accessed the GE_UC_ADDR will increment automatically.

5. Programming the Core

This sections describes how to set up and control the XPS_GRAPHICS_ENGINE core so that graphics commands can be executed by the core.

5.1 Initialisation Sequence

After reset the GECPU can be enabled by setting the GE_CTRL_RESET bit in the GE_CTRL

register after which the GECPU will be ready to receive commands. If interrupts are used, they should be enabled and configured as described in section 5.2. If another set of microcode than what is included with the core is used, the 1K microcode RAM should be programmed first, before enabling the GECPU, as described in the section 5.2

5.2 Interrupts

The core has two interrupts which are logically OR'ed together before being output on the *IP2INTC_Irpt* interrupt pin.

1. Command FIFO level interrupt, enabled by GE_CTRL_FIFO_IRQEN
2. Command completion interrupt, enabled GE_CTRL_RDY_IRQEN

The command FIFO level interrupt is asserted when the number of words in the FIFO is less than or equal to the level programmed into the GE_CTRL_FIFO_THRESHOLD field of the GE_CTRL register. The command FIFO is 16 words deep and levels between 0 and 15 are relevant.

The command completion interrupt is used to detect when all GE operations have completed and no further command words exist in the FIFO. This includes any operations of the accelerator blocks. The interrupt can be enabled when the application needs to know when drawing operations have completed before continuing.

The level interrupt can be cleared by writing enough words into the command FIFO so the number of words is greater then the level programmed into the GE_CTRL_FIFO_THRESHOLD field of the GE_CTRL register. If no further commands need to be issued, the command completion can optionally be enabled instead of the command FIFO level interrupt.

The command completion interrupt can only be cleared by disabling it.

5.3 Issuing commands to the core

Commands are written to the 16-word deep command FIFO. Most commands take one or more words as parameters which follows the initial command word. Parameters are drawing coordinates, addresses, drawing modes etc. Ensure that the command FIFO is not full before writing to it. Writing to a full FIFO results in the word written to be ignored.

The commands available are shown in the table below:

Command	words	Format	Description
GE_CMD_SETXY	1	[31:24] = 0 [23:12] = y [11:0] = x	Sets the current plot location to (x,y).
GE_CMD_PLOTXY	1	[31:24] = 1 [23:12] = y [11:0] = x	Draws a single pixel at location (x,y) using the current mode and colour. The current plot coordinate is then set to (x,y).
GE_CMD_BITBLT	4	---- *word 1* ---- [31:24] = 2 [23:12] = height [11:0] = width ---- *word 2* ---- [31:16] = sinc [15:0] = dinc ---- *word 3* ---- *[31:0] = saddr*	Copies a rectangle of dimensions (width, height) from address *saddr to daddr.* For each row copied the *saddr* and *daddr* is incremented by *sinc* and *dinc*. All addresses and increment values have to be even values to align to a pixel. The current mode is used between the source and the destination pixels.

		---- word 4 ---- *[31:0] = daddr*	
GE_CMD_PLOTHLINE	1	[31:24] = 3 [23:12] = y [11:0] = x	Draws a horizontal line from the current plot coordinate to (x,current y). The current mode and colour is used. The current plot coordinate is then set to (x,y).
GE_CMD_PLOTVLINE	1	[31:24] = 4 [23:12] = y [11:0] = x	Draws a vertical line from the current plot coordinate to (current x,y). The current mode and colour is used. The current plot coordinate is then set to (x,y).
GE_CMD_PLOTLINE	1	[31:24] = 5 [23:12] = y [11:0] = x	Draws a line from the current plot coordinate to the pixel at x,y. The current mode and colour is used. The current plot coordinate is then set to (x,y).
GE_CMD_SETCOLMOD	1	[31:24] = 6 [23:20] = ignored [19:12] = mode [11:0] = colour	Sets the current colour and mode
GE_CMD_SETBASEWIDTH	2	---- word 1 ---- [31:24] = 7 [23:16] = ignored [15:0] = width ---- word 2 ---- [31:0] = baseaddr	Sets the base address for drawing operations (except BitBlt) and the width of the destination canvas.
GE_CMD_SETALPHA	1	[31:24] = 7 [23:5] = ignored [4:0] = alpha	Sets the current alpha value for use with the alpha blending mode. The alpha is an unsigned value in 1.4 format.

Table 1 – Graphics Commands

5.4 Drawing Modes

The following table lists the modes available for drawing and BitBlt commands.

Mode	Code	Meaning for BitBlt commands	Meaning for non-BitBlt commands
GE_MODE_DIRECT	0x00	Pixels from the source is written to the destination.	Pixels at the destination is overwritten with the current colour.
GE_MODE_READONLY	0x10	Currently unused	Currently unused
GE_MODE_AND	0x20	Pixels written are the logical AND between the pixel at the destination and the current colour.	Pixels written are the logical AND between the pixel at the destination and the source.
GE_MODE_OR	0x30	Pixels written are the logical OR between the pixel at the destination and the current colour.	Pixels written are the logical OR between the pixel at the destination and the source.

GE_MODE_XOR	0x40	Pixels written are the logical XOR between the pixel at the destination and the current colour.	Pixels written are the logical XOR between the pixel at the destination and the source.
GE_MODE_KEY	0x50	Not an allowed mode	The pixels at the destination are only overwritten with the source when the source pixels do not match the configured value of the C_COLOR_KEY parameter. Alpha blending is also active in this mode.
GE_MODE_ALPHA	0x60	Not an allowed mode	The pixels at the destination and source is blended using the current alpha value. See section 1.5 for how the blending is performed.

5.5 Pixel format

To reduce the size and complexity of the the core, the pixel format is fixed to a 16 bits per pixel format with the colour components packed as shown below:

Bit	Colour
[15:11]	Red component
[10:5]	Green component
[4:0]	Blue component

.

6. Resource Utilization

The core resources used in the spartan-6, virtex-5, and spartan-3 series are detailed in the table below:

Device family	Resource	
Spartan-6	Slice LUTs	964
	Slice Registers	420
	BRAM18	1
	DSP48A1	5
Virtex-5	Slice LUTs	1024
	Slice Registers	419
	BRAM18	1
	DSP48E	5
Spartan-3a	Slices	674
	BRAM (18kbit)	1
	MUL18X18	5

Virtex-6	Slice LUTs	942
	Slice Registers	420
	BRAM18	1
	DSP48E	5
Kintex 7	Slice LUTs	942
	Slice Registers	420
	BRAM18	1
	DSP48E1	5

7. Revision history

Date	Version	Revision
November 1st, 2011	0.9	Pre release
November 22nd, 2011	1.0	First release

8. Notice of Disclaimer