

Virtual Gate

V7.0

© 2017 MStar Semiconductor, Inc. All rights reserved.

MStar Semiconductor makes no representations or warranties including, for example but not limited to, warranties of merchantability, fitness for a particular purpose, infringement of any intellectual property right or the accuracy or completeness of this document, and reserves the right to make changes without further notice to any products herein to improve reliability, function or design. No responsibility is assumed by MStar Semiconductor arising out of the application or user of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

MStar is a trademark of MStar Semiconductor, Inc. Other trademarks or names herein are only for identification purposes only and owned by their respective owners.

REVISION HISTORY

Revision No.	Description	Date
1.0	• Created.	10/04/2016
2.0	• System information, return value and VgSet modification	10/28/2016
3.0	• Add ROI illustration	11/18/2016
4.0	• Add Iva_VG_Release introduction	11/24/2016
5.0	• Change the code size and DRAM usage	12/16/2016
6.0	• Add API Introduction and change flowchart	03/21/2017
7.0	• Modify the structure and introduction	04/13/2017
8.0	• Modify the return value	04/26/2017

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1. Purpose	1
2. API REFERENCE	2
2.1. API Overview	2
2.2. API Lists	3
<i>MI_VG_Init</i>	3
<i>MI_VG_Uninit</i>	3
<i>MI_VG_SetScene</i>	4
<i>MI_VG_GetScene</i>	4
<i>MI_VG_SetLineNumber</i>	5
<i>MI_VG_GetLineNumber</i>	6
<i>MI_VG_SetLineAndDir</i>	6
<i>MI_VG_GetLineAndDir</i>	7
<i>MI_VG_SetObjectSizeThd</i>	8
<i>MI_VG_GetObjectSizeThd</i>	9
<i>MI_VG_Run</i>	10
<i>Mi_VG_GetResult</i>	10
<i>Mi_VG_GetDebugInfo</i>	11
3. DATA TYPE	12
3.1. Overview	12
3.2. Struct Lists	12
<i>MI_VG_RET</i>	12
<i>MI_VG_HANDLE</i>	13
<i>MI_VG_Point_t</i>	14
<i>MI_VG_Line_t</i>	14
<i>MI_VgSet_t</i>	15
<i>MI_VgResult_t</i>	16
<i>MI_VgDebug_t</i>	17
4. VIRTUAL GATE FLOW	19
5. SYSTEM INFORMATION	20
5.1. Code Size Information	20
5.2. DRAM Usage Information (Working Buffer)	20
5.3. CPU MIPS/Clock Cycles Estimation	20

LIST OF FIGURES

Figure 1 Code Size	20
Figure 1 Code Size	20

1. INTRODUCTION

1.1. Purpose

Virtual Gate is an algorithm for detect the object whether through the virtual gate or not. This algorithm can used to surveillance, IP camera and security monitoring....

2. API REFERENCE

2.1. API Overview

- [MI_VG_Init](#) : Initialize virtual gate library.
- [MI_VG_Uninit](#) : Release virtual gate library and free buffer.
- [MI_VG_SetScene](#) : Set parameter for scene.
- [MI_VG_GetScene](#) : Get the parameter of scene.
- [MI_VG_SetLineNumber](#) : Set the number for virtual line.
- [MI_VG_GetLineNumber](#) : Get the number of virtual line.
- [MI_VG_SetLineAndDir](#) : Set the coordinate information for virtual line.
- [MI_VG_GetLineAndDir](#) : Get the coordinate information of virtual line.
- [MI_VG_SetObjSizeThd](#) : Set threshold for detect object.
- [MI_VG_GetObjSizeThd](#) : Get the threshold of detect object.
- [MI_VG_Run](#) : Run virtual gate library.
- [MI_VG_GetResult](#) : Get the result for alarm.
- [MI_VG_GetDebugInfo](#) : Get the debug information for analyze.

2.2. API Lists

MI_VG_Init

Purpose

Initialize virtual gate library.

Function Prototype

```
MI_VG_HANDLE MI_VG_Init(MI_VgSet_t* vg_user_info, uint16_t width, uint16_t height);
```

Arguments

Name	Description
vg_user_info	The structure of MI_VgSet_t
Width	Width of source image
Height	Height of source image

Return value

Name	Description
MI_VG_HANDLE	Not 0 : Success
NULL	0 : Fail

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

- vg_user_info must allocate buffer and set parameter at first.
- Suggest image resolution is 320 x180.

MI_VG_Uninit

Purpose

Release virtual gate library and free buffer.

Function Prototype

```
void MI_VG_Uninit(MI_VG_HANDLE vg_handle);
```

Arguments

Name	Description
vg_handle	Pointer to handle (It cannot be null)

Return value

None

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

None

MI_VG_SetScene

Purpose

Set parameter for scene.

Function Prototype

```
MI_VG_RET MI_VG_SetScene(MI_VgSet_t * vg_user_info, int8_t scene);
```

Arguments

Name	Description
vg_user_info	The structure of MI_VgSet_t
scene	value of indoor

Return value

Name	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INVALID_USER_INFO_POINTER	Invalid user information pointer
MI_VG_RET_INVALID_ENVIRONMENT_STATE	Invalid environment state

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

Scene value must be 0 or 1. (0 = outdoor ; 1 = indoor)

MI_VG_GetScene

Purpose

Get the parameter of scene.

Function Prototype

```
MI_VG_RET MI_VG_GetScene(MI_VgSet_t * vg_user_info, int8_t* scene);
```


Arguments

Name	Description
vg_user_info	The structure of MI_VgSet_t
scene	Unsigned char pointer (Use to get the value of indoor)

Return value

Value	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INVALID_USER_INFO_POINTER	Invalid user information pointer
MI_VG_RET_INVALID_ENVIRONMENT_POINTER	Invalid environment pointer
MI_VG_RET_INVALID_ENVIRONMENT_STATE	Invalid environment state

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

- scene pointer must be allocated buffer at first.

MI_VG_SetLineNumber

Purpose

Set the number for virtual gate.

Function Prototype

```
MI_VG_RET MI_VG_SetLineNumber(MI_VgSet_t * vg_user_info, uint16_t lineno);
```

Arguments

Name	Description
vg_user_info	The structure of MI_VgSet_t
lineno	value of line_number

Return value

Name	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INVALID_USER_INFO_POINTER	Invalid user information pointer
MI_VG_RET_INVALID_LINE_NUMBER	Invalid line number

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

lineno value must be 1 or 2.

MI_VG_GetLineNumber

Purpose

Get the number of virtual line.

Function Prototype

```
MI_VG_RET MI_VG_GetLineNumber(MI_VgSet_t * vg_user_info, uint16_t* lineno);
```

Arguments

Name	Description
vg_user_info	The structure of MI_VgSet_t
lineno	Unsigned short pointer (Use to get the value of line_number)

Return value

Value	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INVALID_USER_INFO_POINTER	Invalid user information pointer
MI_VG_RET_INVALID_LINE_POINTER	Invalid line pointer
MI_VG_RET_INVALID_LINE_NUMBER	Invalid line number

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

- lineno pointer must be allocated buffer at first.

MI_VG_SetLineAndDir

Purpose

Set the coordinate information for virtual line.

Function Prototype

```
MI_VG_RET MI_VG_SetLineAndDir(MI_VgSet_t * vg_user_info, MI_VgLine_t * line_coordinate,
uint16_t lineno);
```

Arguments

Name	Description
vg_user_info	The structure of MI_VgSet_t
line_coordinate	Virtual line information (Coordinate of first point · coordinate of second point · coordinate of first direction point · and coordinate of second direction point)
lineno	The index of the selected line

Return value

Name	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INVALID_USER_INFO_POINTER	Invalid user information pointer
MI_VG_RET_INVALID_COORDINATE_POINTER	Invalid coordinate pointer
MI_VG_RET_INVALID_LINE_NUMBER	Invalid line number
MI_VG_RET_INVALID_FIRST_LINE_INFO	Invalid first line information
MI_VG_RET_INVALID_SECOND_LINE_INFO	Invalid second line information

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

- lineno value must be 1 or 2.
- Line 1 and line 2 can not cross.
- The length of virtual gate is suggested smaller than the half of width of input image.

MI_VG_GetLineAndDir

Purpose

Get the coordinate information of virtual line.

Function Prototype

```
MI_VG_RET MI_VG_GetLineAndDir(MI_VgSet_t * vg_user_info, MI_VgLine_t * line_coordinate,
uint16_t lineno);
```

Arguments

Name	Description
vg_user_info	The structure of MI_VgSet_t
line_coordinate	Virtual line information (Coordinate of first point , coordinate of second point , coordinate of first direction point , and coordinate of second direction point)
lineno	The index of the selected line

Return value

Name	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INVALID_USER_INFO_POINTER	Invalid user information pointer
MI_VG_RET_INVALID_COORDINATE_POINTER	Invalid coordinate pointer
MI_VG_RET_INVALID_LINE_NUMBER	Invalid line number
MI_VG_RET_INVALID_FIRST_LINE_INFO	Invalid first line information
MI_VG_RET_INVALID_SECOND_LINE_INFO	Invalid second line information

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

- line_coordinate pointer must be allocated buffer at first.

MI_VG_SetObjectSizeThd

Purpose

Set threshold for detect object.

Function Prototype

```
MI_VG_RET MI_VG_SetObjSizeThd(MI_VgSet_t * vg_user_info, uint16_t size_thd);
```

Arguments

Name	Description
vg_user_info	The structure of MI_VgSet_t
size_thd	value of size_thd

Return value

Name	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INVALID_USER_INFO_POINTER	Invalid user information pointer
MI_VG_RET_INVALID_THRESHOLD	Invalid object threshold

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

size_thd value must be 0 ~ 100.

MI_VG_GetObjectSizeThd

Purpose

Get the threshold of detect object.

Function Prototype

```
MI_VG_RET MI_VG_GetObjSizeThd(MI_VgSet_t * vg_user_info, uint16_t* size_thd);
```

Arguments

Name	Description
vg_user_info	The structure of MI_VgSet_t
size_thd	Unsigned short pointer (Use to get the value of size_thd)

Return value

Value	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INVALID_USER_INFO_POINTER	Invalid user information pointer
MI_VG_RET_INVALID_THRESHOLD_POINTER	Invalid threshold pointer
MI_VG_RET_INVALID_THRESHOLD	Invalid object threshold

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

- size_thd pointer must be allocated buffer at first.

MI_VG_Run

Purpose

Run virtual gate library.

Function Prototype

```
MI_VG_RET MI_VG_Run(MI_VG_HANDLE vg_handle, uint8_t* _ucMask);
```

Arguments

Name	Description
vg_handle	Pointer to VG handle (It cannot be null).
_ucMask	Preview buffer address.

Return value

Name	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INVALID_HANDLE	Invalid VG handle
MI_VG_RET_INVALID_INPUT_POINTER	Invalid input pointer
MI_VG_RET_OPERATE_ERROR	VG operate error

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

- For reduce cpu usage, fame rate suggest to be 10 ~15

Mi_VG_GetResult

Purpose

Get the result for alarm.

Function Prototype

```
MI_VG_RET MI_VG_GetResult(MI_VG_HANDLE vg_handle, VgResult *cross_alarm);
```

Arguments

Name	Description
vg_handle	Pointer to VG handle (It cannot be null).
cross_alarm	The structure of VgResult

Return value

Name	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INVALID_HANDLE	Invalid VG handle
MI_VG_RET_INVALID_ALARM_POINTER	Invalid alarm pointer
MI_VG_RET_OPERATE_ERROR	VG operate error

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

- cross_alarm pointer must be allocated buffer at first.

*Mi_VG_GetDebugInfo***Purpose**

Get the debug information for analyze.

Function Prototype

MI_VG_RET MI_VG_GetDebugInfo(MI_VG_HANDLE vg_handle, VgDebug *debug_info);

Arguments

Name	Description
vg_handle	Pointer to VG handle (It cannot be null).
debug_info	The structure of VgDebug.

Return value

Name	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INVALID_HANDLE	Invalid VG handle
MI_VG_RET_INVALID_DEBUG_POINTER	Invalid debug pointer
MI_VG_RET_OPERATE_ERROR	VG operate error

Requirement

Header files: mi_vg.h

Library files: libVG_LINUX.a 、libVG_LINUX.so

Note

- debug_info pointer must be allocated buffer at first.

3. DATA TYPE

3.1. Overview

MI_VG_RET	The structure of return value
MI_VG_HANDLE	Pointer to VG handle (It cannot be null)
MI_VG_Point_t	Define the coordinate of virtual gate
MI_VG_Line_t	The structure of virtual gate information
MI_VgSet_t	The structure of virtual gate parameter
MI_VgResult_t	The structure of result
MI_VgDebug_t	The structure of virtual gate debug information

3.2. Struct Lists

MI_VG_RET

Description

The structure of return value.

Syntax

```
typedef enum _MI_RET_E
{
    MI_VG_RET_SUCCESS                = 0x00000000,
    MI_VG_RET_INIT_ERROR             = 0x10000301,
    MI_VG_RET_IC_CHECK_ERROR         = 0x10000302,
    MI_VG_RET_INVALID_HANDLE         = 0x10000303,
    MI_VG_RET_INVALID_USER_INFO_POINTER = 0x10000304,
    MI_VG_RET_INVALID_ENVIRONMENT_STATE = 0x10000305,
    MI_VG_RET_INVALID_ENVIRONMENT_POINTER = 0x10000306,
    MI_VG_RET_INVALID_LINE_NUMBER     = 0x10000307,
    MI_VG_RET_INVALID_LINE_POINTER    = 0x10000308,
    MI_VG_RET_INVALID_COORDINATE_POINTER = 0x10000309,
    MI_VG_RET_INVALID_FIRST_LINE_INFO = 0x1000030A,
    MI_VG_RET_INVALID_SECOND_LINE_INFO = 0x1000030B,
    MI_VG_RET_INVALID_THRESHOLD       = 0x1000030C,
    MI_VG_RET_INVALID_THRESHOLD_POINTER = 0x1000030D,
    MI_VG_RET_INVALID_INPUT_POINTER   = 0x1000030E,
    MI_VG_RET_OPERATE_ERROR           = 0x1000030F,
```



```

MI_VG_RET_INVALID_ALARM_POINTER      = 0x10000310,
MI_VG_RET_INVALID_DEBUG_POINTER      = 0x10000311,
} MI_RET;

```

Member

Member	Description
MI_VG_RET_SUCCESS	Success
MI_VG_RET_INIT_ERROR	VG init error
MI_VG_RET_IC_CHECK_ERROR	Platform check error
MI_VG_RET_INVALID_HANDLE	Invalid VG handle
MI_VG_RET_INVALID_USER_INFO_POINTER	Invalid user information pointer
MI_VG_RET_INVALID_ENVIRONMENT_STATE	Invalid environment state
MI_VG_RET_INVALID_ENVIRONMENT_POINTER	Invalid environment pointer
MI_VG_RET_INVALID_LINE_NUMBER	Invalid line number
MI_VG_RET_INVALID_LINE_POINTER	Invalid line pointer
MI_VG_RET_INVALID_COORDINATE_POINTER	Invalid coordinate pointer
MI_VG_RET_INVALID_FIRST_LINE_INFO	Invalid first line information
MI_VG_RET_INVALID_SECOND_LINE_INFO	Invalid second line information
MI_VG_RET_INVALID_THRESHOLD	Invalid object threshold
MI_VG_RET_INVALID_THRESHOLD_POINTER	Invalid threshold pointer
MI_VG_RET_INVALID_INPUT_POINTER	Invalid input pointer
MI_VG_RET_OPERATE_ERROR	VG operate error
MI_VG_RET_INVALID_ALARM_POINTER	Invalid alarm pointer
MI_VG_RET_INVALID_DEBUG_POINTER	Invalid debug pointer

Note

None.

MI_VG_HANDLE

Description

Pointer to VG handle (It cannot be null).

Syntax

```
typedef void* MI_VG_HANDLE;
```

Member

None

Note

None.

MI_VG_Point_t

Description

Define the coordinate of virtual gate.

Syntax

```
typedef struct _MI_VG_Point_t
{
    int32_t x;
    int32_t y;
} MI_VG_Point_t;
```

Member

Member	Description
x	Coordinate of x
y	Coordinate of y

Note

None.

MI_VG_Line_t

Description

The structure of virtual gate information.

Syntax

```
typedef struct _MI_VG_Line_t
{
    VG_Point_t px;
    VG_Point_t py;
    VG_Point_t pdx;
    VG_Point_t pdy;
} MI_VG_Line_t;
```

Member

Member	Description
Px	First coordinate point
Py	Second coordinate point
Pdx	First direction point
Pdy	Second direction point

Note

None.

MI_VgSet_t

Description

Define the parameter of virtual gate.

Syntax

```
typedef struct _MI_VgSet_t
{
    uint16_t object_size_thd;
    uint16_t line_number;
    uint8_t indoor;
    uint16_t fx;
    uint16_t fy;
    uint16_t sx;
    uint16_t sy;
    uint16_t fdx;
    uint16_t fdy;
    uint16_t sdx;
    uint16_t sdy;
    uint16_t sfx;
    uint16_t sfy;
    uint16_t ssx;
    uint16_t ssy;
    uint16_t sfdx;
    uint16_t sfdy;
    uint16_t ssdx;
    uint16_t ssdy;
}MI_VgSet_t;
```

Member

Member	Description
object_size_thd	Threshold of the object size (percentage of preview image size).
Line_number	Virtual gate number.
indoor	Environment state (0 = outdoor ; 1 = indoor).
fx	The x coordinate of first point for first virtual gate.
fy	The y coordinate of first point for first virtual gate.
sx	The x coordinate of second point for first virtual gate.
sy	The y coordinate of second point for first virtual gate.

Member	Description
fdx	The x coordinate of first direction point for first virtual gate.
fdy	The y coordinate of first direction point for first virtual gate.
sdx	The x coordinate of second direction point for first virtual gate.
sdv	The y coordinate of second direction point for first virtual gate.
sfx	The x coordinate of first point for second virtual gate.
sfy	The y coordinate of first point for second virtual gate.
ssx	The x coordinate of second point for second virtual gate.
ssy	The y coordinate of second point for second virtual gate.
sfdx	The x coordinate of first direction point for second virtual gate.
sfdy	The y coordinate of first direction point for second virtual gate.
ssdx	The x coordinate of second direction point for second virtual gate.
ssdy	The y coordinate of second direction point for second virtual gate.

Note

- If the object size smaller than object_size_thd, will be removed. Range = 0 ~ 100 (default = 5).
- line_number = 1 or 2.
- indoor = 0 or 1 (0 = outdoor ; 1 = indoor).
- The direction is decided by the direction point, if the object from first direction point side to second direction point side, alarm will be triggered.
- If the coordinate of first direction point and the second direction are same, mean no direction, object cross to the virtual gate, alarm will be triggeren.
- If the first direction point and the second direction point are in the same side, mean no direction, bject cross to the virtual gate, alarm will be triggeren.
- If the first direction point and the second direction point are on the virtual gate, mean no direction, bject cross to the virtual gate, alarm will be triggeren.
- The length of virtual gate is suggested smaller than the half of width of input image.

MI_VgResult_t

Description

Define the result of Virtual Gate.

Syntax

```
typedef struct _MI_VgResult_t
{
    uint16_t alarm1;
    uint16_t alarm2;
}MI_VgResult_t;
```

Member

Member	Description
alarm1	Alarm situation for first virtual gate, 0 = normal, 1 = alarm.
alarm2	Alarm situation for second virtual gate, 0 = normal, 1 = alarm.

Note

None.

MI_VgDebug_t

Description

Define the VG debug information.

Syntax

```
typedef struct _MI_VgDebug_t
{
    uint16_t background_state;
    uint32_t version;
    uint32_t debug_object_size;
    uint32_t debug_state;
    uint16_t debug_fsp_x;
    uint16_t debug_fsp_y;
    uint16_t debug_fep_x;
    uint16_t debug_fep_y;
    uint16_t debug_fx;
    uint16_t debug_fy;
    uint16_t debug_sx;
    uint16_t debug_sy;
    uint16_t debug_fdx;
    uint16_t debug_fdy;
    uint16_t debug_sdx;
    uint16_t debug_sdy;
    uint16_t debug_ssp_x;
    uint16_t debug_ssp_y;
    uint16_t debug_sep_x;
    uint16_t debug_sep_y;
    uint16_t debug_sfx;
    uint16_t debug_sfy;
    uint16_t debug_ssx;
    uint16_t debug_ssy;
    uint16_t debug_sfdx;
    uint16_t debug_sfdy;
    uint16_t debug_ssd_x;
    uint16_t debug_ssd_y;
}MI_VgDebug_t;
```

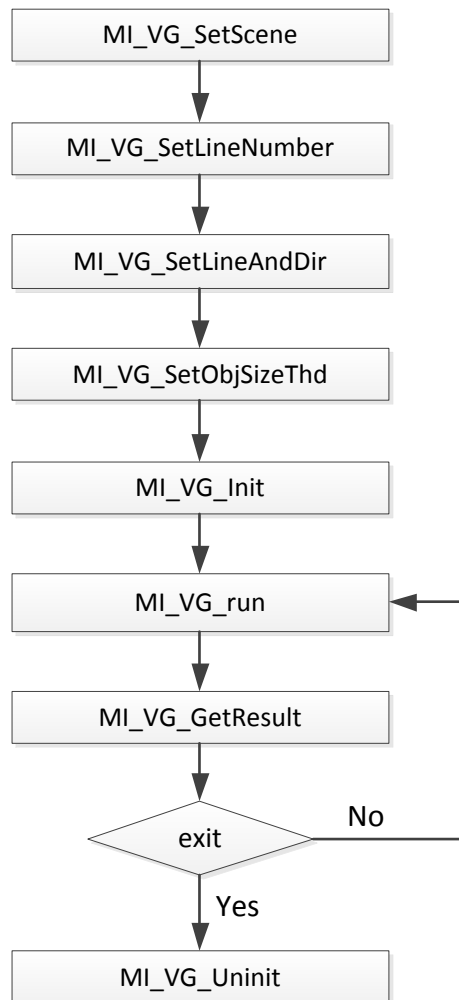
Member

Member	Description
background_state	State of background training. (0 : success ; 1 : fail)
version	Library version.
debug_object_size	Threshold of the object size
debug_state	Debug mode open or close. (0 : close ; 1 : open)
debug_fsp_x	The x coordinate of start point for first interested region.
debug_fsp_y	The y coordinate of start point for first interested region.
debug_fep_x	The x coordinate of end point for first interested region.
debug_fep_y	The y coordinate of end point for first interested region.
debug_fx	The x coordinate of first point for first virtual gate.
debug_fy	The y coordinate of first point for first virtual gate.
debug_sx	The x coordinate of second point for first virtual gate.
debug_sy	The y coordinate of second point for first virtual gate.
debug_fdx	The x coordinate of first direction point for first virtual gate.
debug_fdy	The y coordinate of first direction point for first virtual gate.
debug_sdx	The x coordinate of second direction point for first virtual gate.
debug_sdy	The y coordinate of second direction point for first virtual gate.
debug_ssp_x	The x coordinate of start point for second interested region.
debug_ssp_y	The y coordinate of start point for second interested region.
debug_sep_x	The x coordinate of end point for second interested region.
debug_sep_y	The y coordinate of end point for second interested region.
debug_sfx	The x coordinate of first point for second virtual gate.
debug_sfy	The y coordinate of first point for second virtual gate.
debug_ssx	The x coordinate of second point for second virtual gate.
debug_ssy	The y coordinate of second point for second virtual gate.
debug_sfdx	The x coordinate of first direction point for second virtual gate.
debug_sfdy	The y coordinate of first direction point for second virtual gate.
debug_ssd_x	The x coordinate of second direction point for second virtual gate.
debug_ssd_y	The y coordinate of second direction point for second virtual gate.

Note

None

4. VIRTUAL GATE FLOW



5. SYSTEM INFORMATION

5.1. Code Size Information








	Code	RO Data	RW Data	ZI Data	Debug	
	36192	4554	8284	59536	114144	Grand Totals
	=====					
	Total RO	Size(Code + RO Data)			40746	(39.79kB)
	Total RW	Size(RW Data + ZI Data)			67820	(66.23kB)
	Total ROM	Size(Code + RO Data + RW Data)			49030	(47.88kB)
	=====					

Figure 1 Code Size

5.2. DRAM Usage Information (Working Buffer)

For input image 320 x 180, working buffer is about ~1.65MB

ROI = $\sim(1/3)(320 \times 180)$, depend on the length of virtual gate.

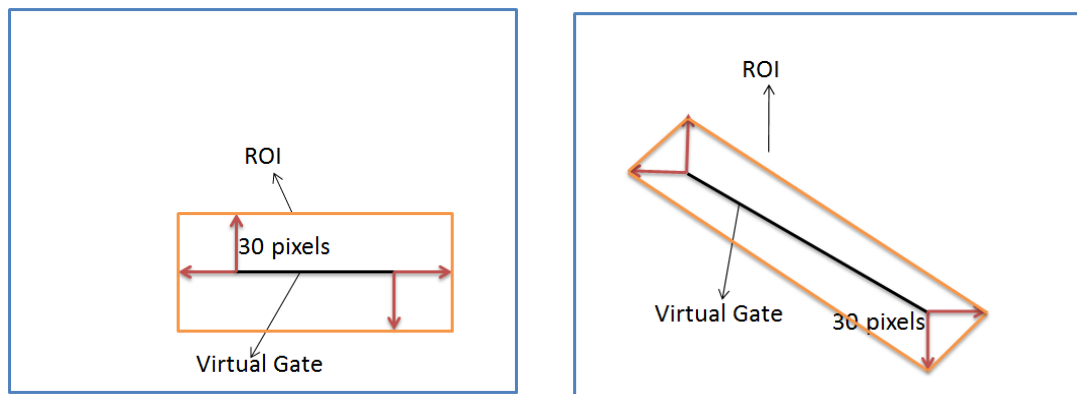


Figure 2 Example

5.3. CPU MIPS/Clock Cycles Estimation

For process one input image 320 x 180

Instructions : ~20.3M

Cycles : ~26.4M