Notes about tiling

Junming Liu

Outline

- Why Tilling
- Background
- Tiled (Fenced) Regions

Outline

- Why Tilling
- Background
- Tiled (Fenced) Regions

why tilling

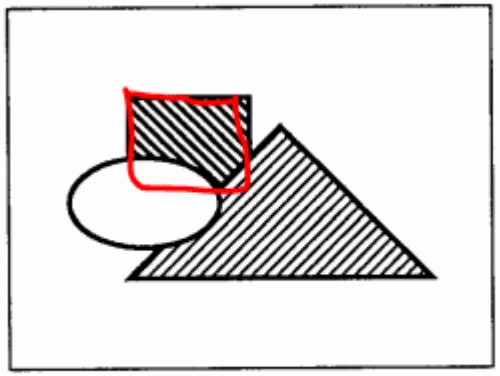


Figure 1: A Row Ordered Array in Memory

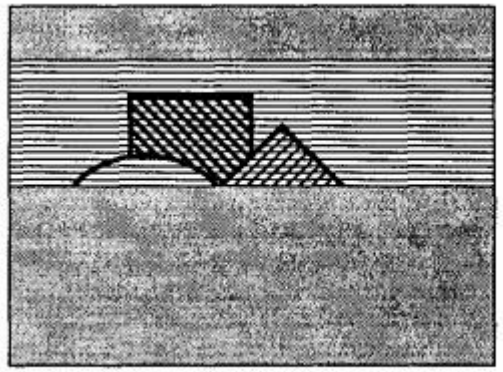


Figure 4: Non-tiled Working Set for Rendering Square

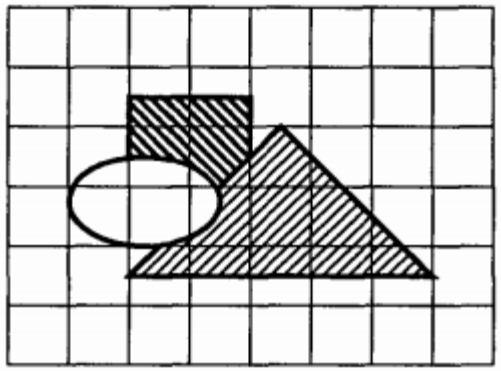


Figure 3: Mapping of Array to Tiles

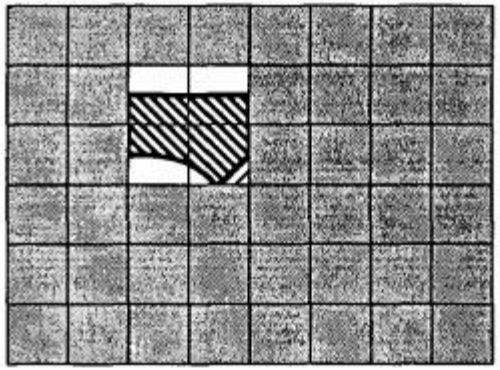


Figure 5: Tiled Working Set for Rendering Square

why tilling

- When we want to update contents in Figure1 red region
- If don’t use tilling, the result will be Figure 4

- If use tilling, the memory layout would be like Figure 3
  - The small rectangle is a 4KB page
- The result will be Figure 5

Notes

- When dealing with memory operands (e.g., graphics surfaces) that are inherently rectangular in nature, certain functions within the graphics device support the storage/access of the operands using tiled memory formats in order to increase performance.

- divide the enclosing region into an array of smaller rectangular regions, called memory *tiles*.

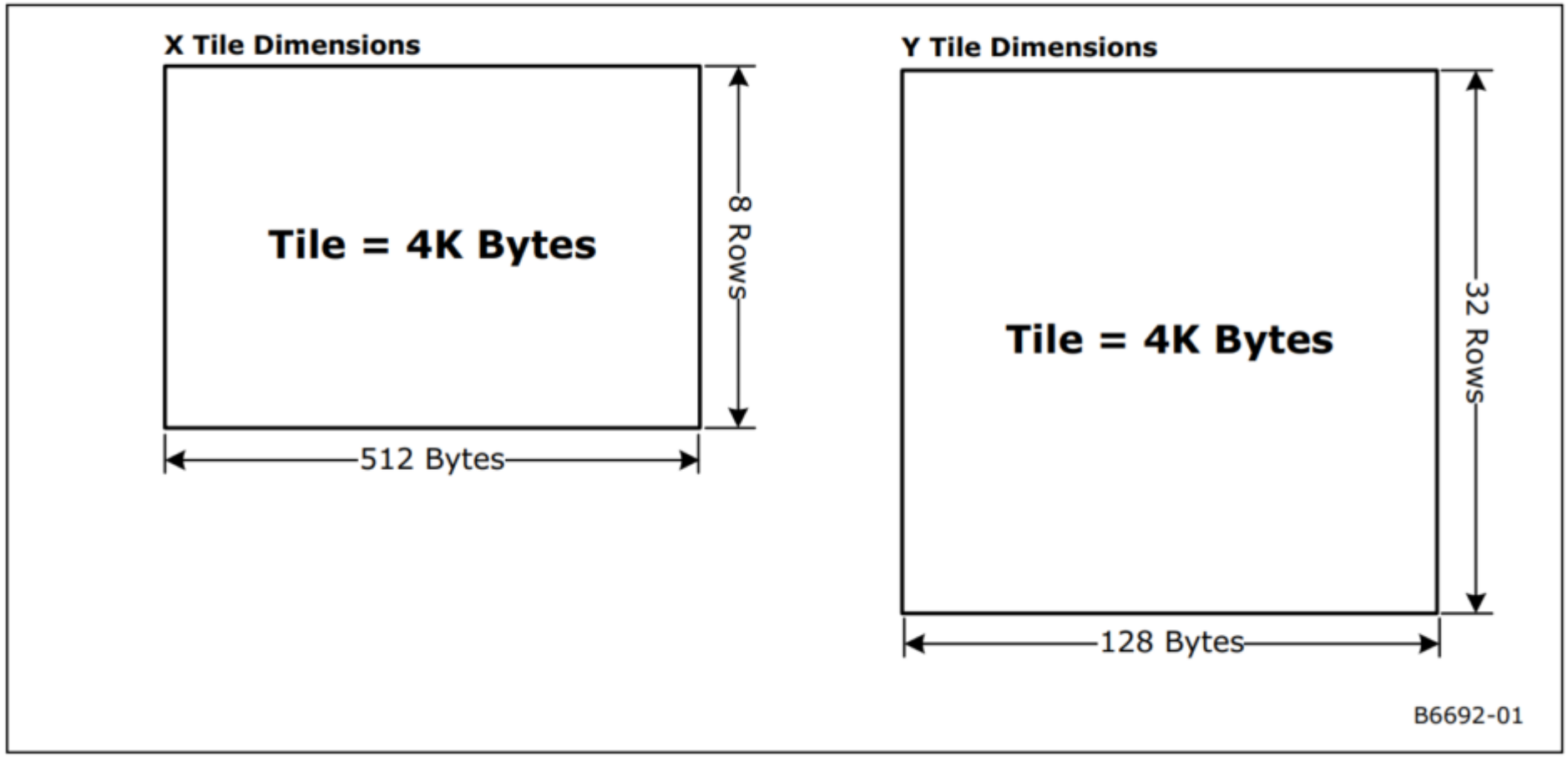
- Surface elements falling within a given tile will all be stored in the same physical memory page, thus eliminating page-crossing penalties for 2D subregion accesses within a tile and thereby increasing performance.

Outline

- Why Tilling
- Background
- Tiled (Fenced) Regions

Tiles

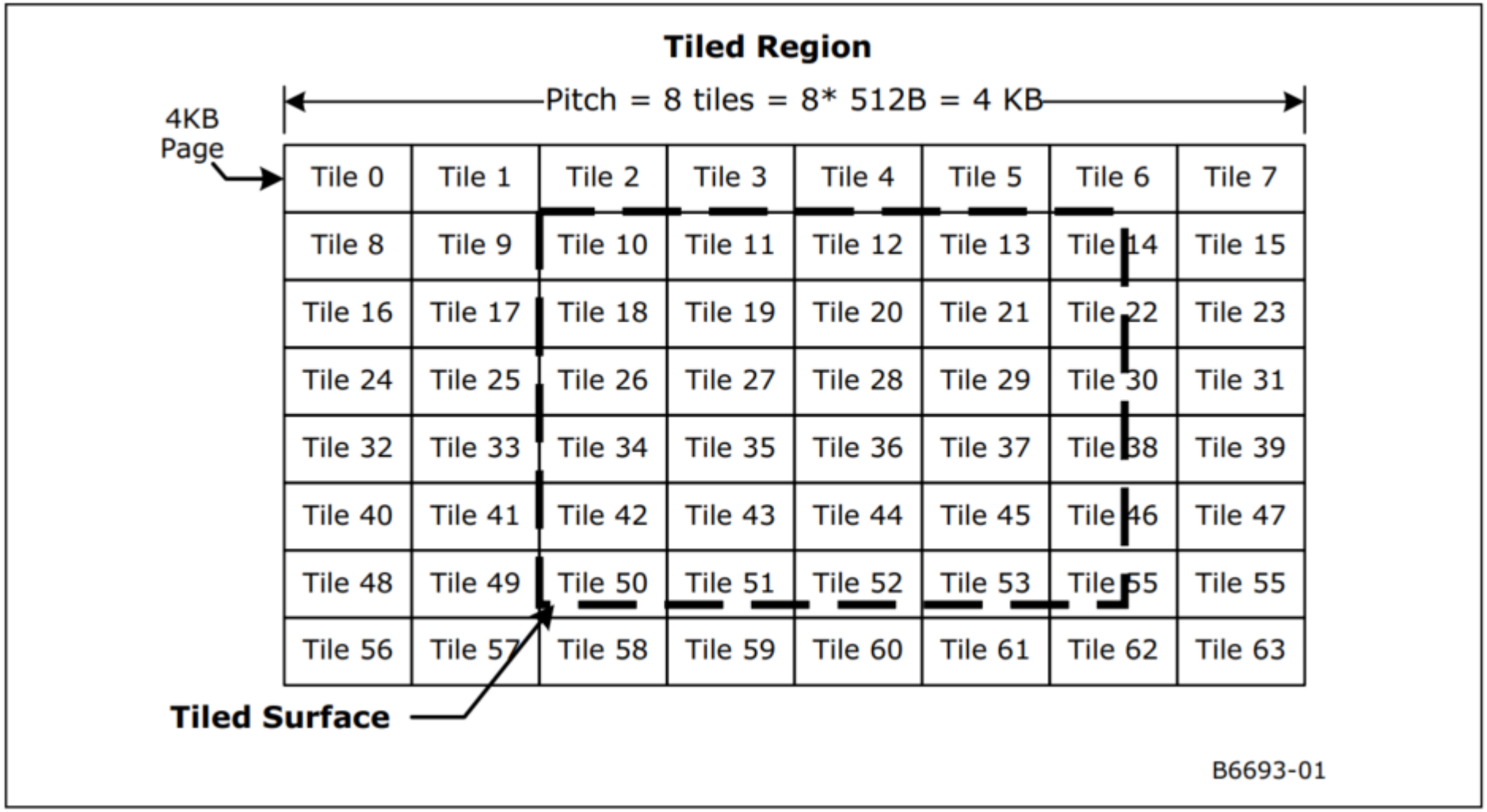
Figure 3-4. Memory Tile Dimensions



Tiles

- Tiles have a fixed 4KB size
- They are either 8 rows high by 512 bytes wide or 32 rows high by 128 bytes wide
- The pitch of a tiled enclosing region must be an integral number of tile widths.

Tiled Surface Layout



Outline

- Why Tilling
- Background
- Tiled (Fenced) Regions

BSP description

- The graphics device performs address translation from linear space to tiled space for a CPU access to graphics memory using the fence registers. Note that the fence registers are used only for CPU accesses to gfx memory. Graphics rendering/display pipelines use Per Surface Tiling (PST) parameters to access tiled gfx memory.
- The intent of tiling is to locate graphics data that are close (in X and Y surface axes) in one physical memory page while still locating some amount of line oriented data sequentially in memory for display efficiency. All 3D rendering is done such that the QWords of any one span are all located in the same memory page, improving rendering performance. Applications view surfaces as linear, hence when the cpu access a surface that is tiled, the gfx hardware must perform linear to tiled address conversion and access the correct physical memory location(s) to get the data.
- Tiled memory is supported for rendering and display surfaces located in graphics memory. A tiled memory surface is a surface that has a width and height that are subsets of the tiled region's pitch and height. The device maintains the constants required by the memory interface to perform the address translations. Each tiled region can have a different pitch and size. The CPU-memory interface needs the surface pitch and tile height to perform the address translation. It uses the GMAddr (PCI-BAR) offset address to compare with the fence start and end address, to determine if the rendering surface is tiled. The tiled address is generated based on the tile orientation determined from the matching fence register. Fence ranges are at least 4 KB aligned. Note that the fence registers are used only for CPU accesses to graphics memory.
- A Tile represents 4 KB of memory. Tile height is 8 rows for X major tiles and 32 rows for Y major tiles. Tile Pitch is 512Bs for X major tiles and 128Bs for Y major tiles. The surface pitch is programmed in 128B units such that the pitch is an integer multiple of "tile pitch".
- Engine restrictions on tile surface usage are detailed in Surface Placement Restrictions (Memory Interface Functions). Note that X major tiles can be used for Sampler, Color, Depth, motion compensation references and motion compensation destination, Display, Overlay, GDI Blt source and destination surfaces. Y major tiles can be used for Sampler, depth, color and motion compensation assuming they do not need to be displayed. GDI Blit operations, overlay and display cannot used Tiled Y orientations.
- A "PST" graphics surface that will also be accessed via fence needs its base address to be tile row aligned.
- Hardware handles the flushing of any pending cycles when software changes the fence upper/lower bounds.
- Fence Table Registers occupy the address range specified above. Each Fence Table Register has the following format.
- FENCE registers are not reset by a graphics reset. They will maintain their values unless a full reset is performed.

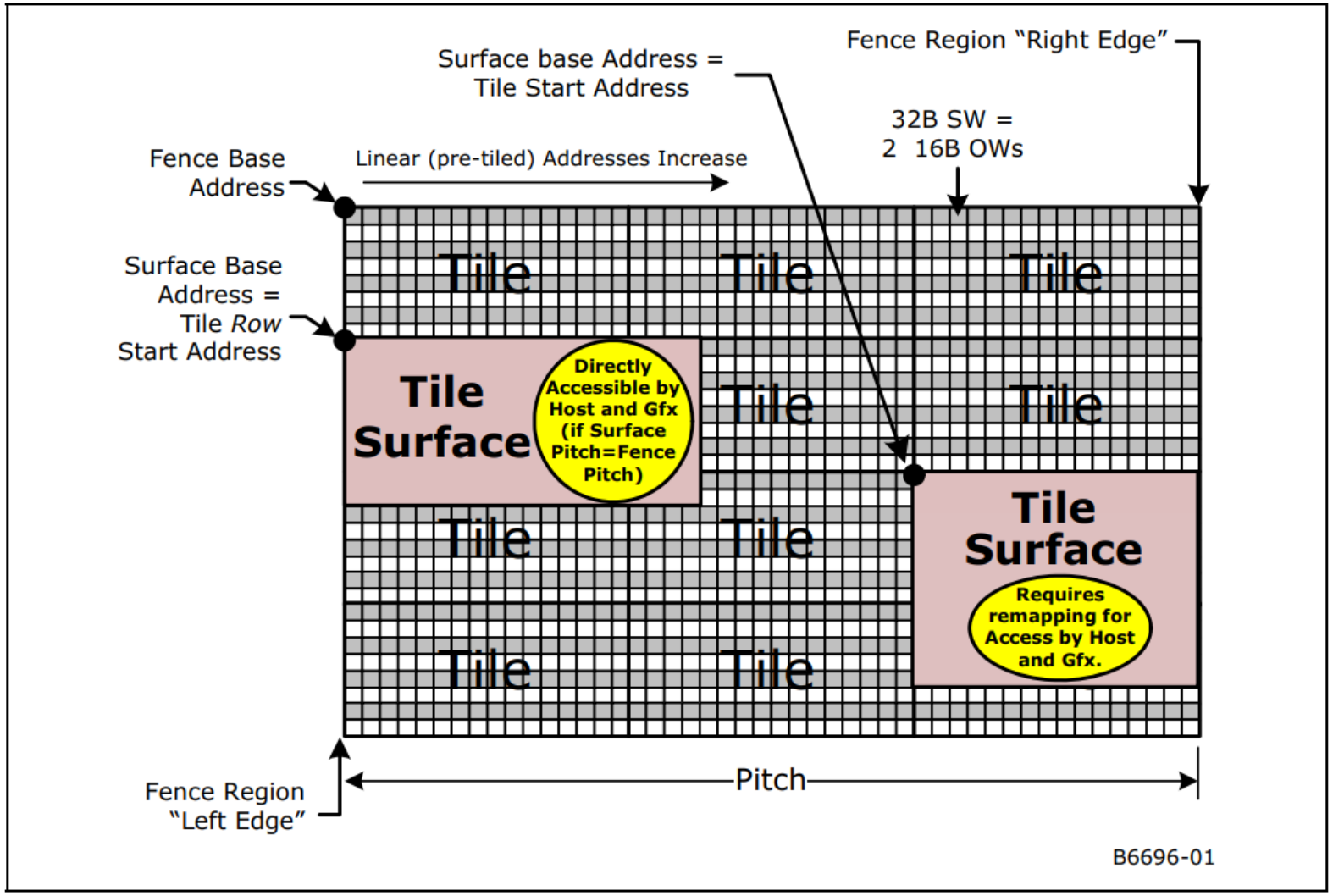
BSP conclusion

- gfx hardware: convert linear address to tiled address

The only mechanism to support the access of surfaces in tiled format by the host or external graphics client is to place them within "fenced" tiled regions within Graphics Memory. A fenced region is a block of Graphics Memory specified using one of the sixteen FENCE device registers. (See *Memory Interface Registers* for details). Surfaces contained within a fenced region are considered tiled from an external access point of view. Note that fences cannot be used to untile surfaces in the PGM_Address space since external devices cannot access PGM_Address space. Even if these surfaces (or any surfaces accessed by an internal graphics client) fall within a region covered by an enabled fence register, that enable will be effectively masked during the internal graphics client access. Only the explicit surface parameters described in the next section can be used to tile surfaces being accessed by the internal graphics clients.

Each FENCE register (if its Fence Valid bit is set) defines a Graphics Memory region ranging from 4KB to the aperture size. The

Tiled Surface Placement



Detail

- It uses the GMAddr (PCI-BAR) offset address to compare with the fence start and end address, to determine if the rendering surface is tiled.

DWord	Bit	Description
0..15	63:44	**Fence Upper Bound** Project: All Address: GraphicsAddress[31:12] Bits 31:12 of the ending Graphics Address of the fence region. Fence regions must be aligned to a 4KB page. This address represents the last 4KB page of the fence region (Upper Bound is included in the fence region). Graphics Address is the offset within GMADR space.
	41:32	**Fence Pitch** Project: All Default Value: 0h DefaultVaueDesc Format: U10-1 Width in 128 bytes This field specifies the width (pitch) of the fence region in multiple of "tile width". For Tile X this field must be programmed to a multiple of 512B ("003" is the minimum value) and for Tile Y this field must be programmed to a multiple of 128B ("000" is the minimum value). 000h = 128B 001h = 256B ... 3FFh = 128KB
	31:12	**Fence Lower Bound** Project: All Address: GraphicsAddress[31:12] Bits 31:12 of the starting Graphics Address of the fence region. Fence regions must be aligned to 4KB. This address represents the first 4KB page of the fence region (Lowe Bound is included in the fence region). Graphics Address is the offset within GMADR space.
	11:2	**Reserved** Project: All Format: MBZ

Conclusion

- 首先完成tiled region的物理page分配
- 将tiled memory 映射到ggtt的logical address
  - Fence regions must be aligned to a 4KB page.
- fence register中记录了fence region的开始地址与结束地址(这些地址都是在ggtt中的偏移量)

Reference

- Newman, Gary. "Memory management support for tiled array organization." *ACM SIGARCH Computer Architecture News* 20.4 (1992): 22-30.
- https://01.org/sites/default/files/documentation/ilk_ihd_os_vol1_part2r2_0.pdf