**Timing Requirements Calculation:**

1 sec / 60 frames = 16.66 msec/frame

1 layer = 640\*480 pixels = 307200 pixels = 307200 words =

16.66 msec / 307200 words = 54.25 nsec / word

For 1 layer per stage pipeline, pixel fmin is 18.432 MHz

Background Process Per Pixel:

Read background at current loc into reg

Write reg to layer 1 at current loc

Layer Process Per Pixel:

Read the data of the current layer at that location into pixel register

Query all sprites to find who is there

If one exists:

Read it’s data at that location into second pixel register

Combine pixels with alpha blending into first pixel register

Write first pixel register to next layer location

If we pipeline, we can’t recoup that spare time (yes we can, pipeline by screen not by pixel and use average screen time / pixels in screen as frequency of pipeline elements)

* *Set limit of 30% screen for each layer (except background)*

Avg period = (0.3)(read + read + query + combine + write) + (0.7)(read + query + write)

Assume query/combine take same time as read/write aka 13.75 ns

Pipeline period = 49.5 ns

Pipeline freq = 20.2 MHz use 20 MHz to be safe

Margin of safety between design freq and required freq is 8.5%

Break up layer process per pixel into 5 stage pipeline to increase screen percent limit per layer?

**Memory Required:**

1 background layer, 4 sprite layers

Current estimate, 5 layers + final framebuffer + sprite storage + 5 “sections” for treadmill scrolling xy

Current sprite size idea, 64x64 pixels with 60 sprites per layer (including offscreen for animation)

Use pool of sprites architecture instead with dynamic layer assignment? Still 300 total though

Sprite memory size for asset storage: 64\*64\*60\*5 = 1228800 pixels

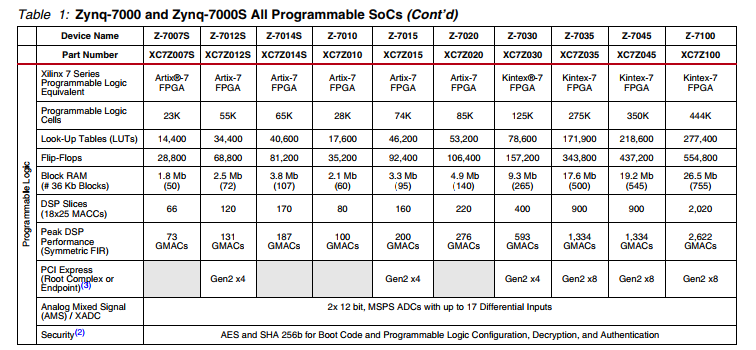
Layers and buffer size: 640\*480\*6 = 1843200 pixels

Assume 1 pixel is 1 word is 1 byte for now

Total Size Estimate: 2.93 MB

Margin of error 2x yields ~6 MB

**Memory Available:**



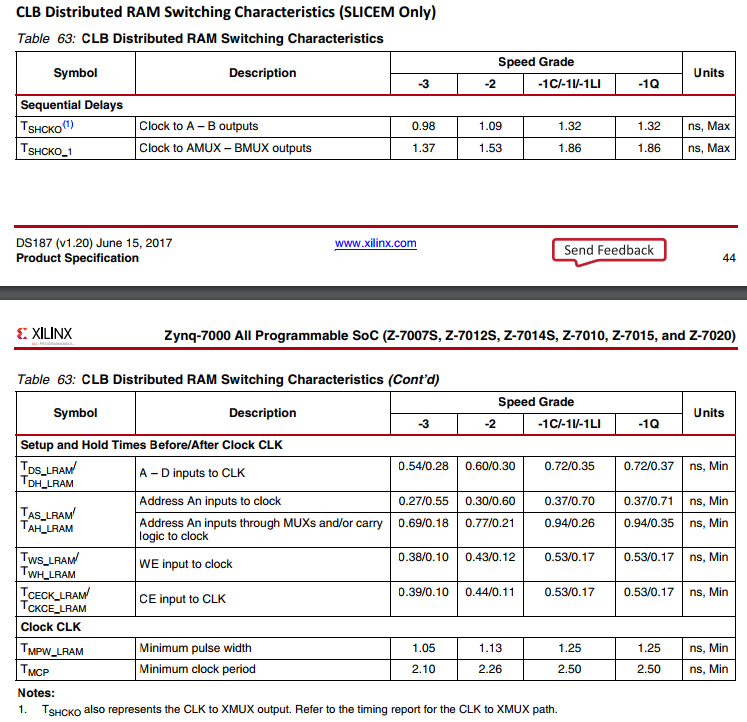
**Block Ram:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Maximum Frequency** | | | |
| **Symbol** | **Description** | **Frequency** | **Units** |
| FMAX\_BRAM\_WF\_NC | Block RAM (write first and no change modes) When not in SDP RF mode. | 388.2 | MHz |
| FMAX\_BRAM\_RF\_PERFORMA NCE | Block RAM (read first, performance mode) When in SDP RF mode but no address overlap between port A and port B. | 388.2 | MHz |
| FMAX\_BRAM\_RF\_DELAYED\_ WRITE | Block RAM (read first, delayed write mode) When in SDP RF mode and there is possibility of overlap between port A and port B addresses. | 339.67 | MHz |
| FMAX\_CAS\_WF\_NC | Block RAM cascade (write first, no change mode) When cascade but not in RF mode. | 345.78 | MHz |
| FMAX\_CAS\_RF\_PERFORMAN CE | Block RAM cascade (read first, performance mode) When in cascade with RF mode and no possibility of address overlap/one port is disabled. | 345.78 | MHz |
| FMAX\_CAS\_RF\_DELAYED\_W RITE | When in cascade RF mode and there is a possibility of address overlap between port A and port B. | 297.35 | MHz |
| FMAX\_FIFO | FIFO in all modes without ECC | 388.2 | MHz |
| FMAX\_ECC | Block RAM and FIFO in ECC configuration | 297.53 | MHz |
|  |  |  |  |
| **Average Fmax** |  | **348.83875** | **MHz** |
| **Size: 240 KB** |  |  |  |

Block Ram, while plenty fast enough to hit our 18.432 MHz target speed, is not large enough.

**Distributed Ram:**

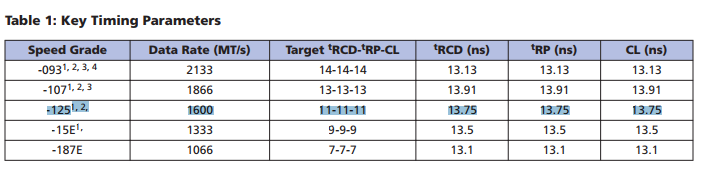
(Only like 200kB available, will find source for official proof later)

****

**DDR3 SDRAM:**

IC is MT41J128M16JT-125

(https://www.micron.com/resource-details/e39e9941-4d9e-426a-9aca-d65a25a01794)



Access time for a single read or write is 13.75 nsec?