COPYRIGHT

DISCLAIMER

This content is provided without warranty or representation of any kind. The use of the content is entirely at your own risk and Stellenbosch University (SU) will have no liability directly or indirectly as a result of this content.

The content must not be assumed to provide complete coverage of the particular study material. Content may be removed or changed without notice.

The video is of a recording with very limited post-recording editing. The video is intended for use only by SU students enrolled in the particular module.

forward together · saam vorentoe · masiye phambili

Computer Systems / Rekenaarstelsels 245

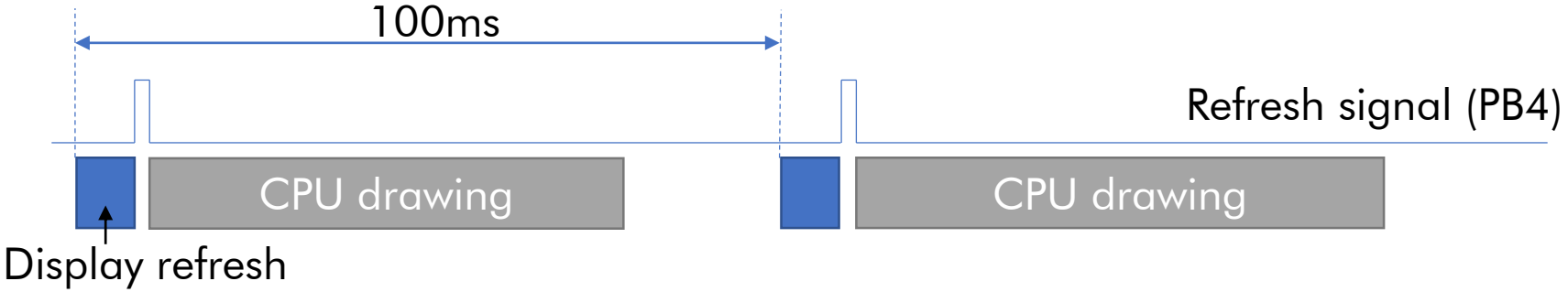
Lecture 27

Emulator Display Optimisation/
Emuleerder Vertooneenheid Optimerings

Dr Rensu Theart & Dr Lourens Visagie

Emulator Refresh signal
Emuleerder verfrissingssein

- The emulator “display controller” will refresh the display every 100ms (10Hz)
- Emulator will output a pulse on PB4 after completing a refresh of the display
- Use this signal to synchronise drawing to the frame buffer
  - hint: use an external interrupt – GPIO_EXTI4 – and in the interrupt handler, set a flag: refresh = 1;
  - In the main function: if (refresh) { …draw to screen… refresh = 0; }



Emulator Display Optimizations
Emuleerder vertoon optimerings

- Perform 32-bit read/write instead of bytes
  - LDRB (load byte) and LDR (load 4 bytes) takes the same amount of time to execute. But LDR loads 4x more data
  - (same with STRB and STR)
  - Make sprite width a multiple of 4. Instead of

```
for (int w = 0; w < 20; w++) {…}
```

do

```
for (int w = 0; w < 5; w++) {…}
```

← but with 32-bit read write operations

Emulator Display Optimizations
Emuleerder vertoon optimerings

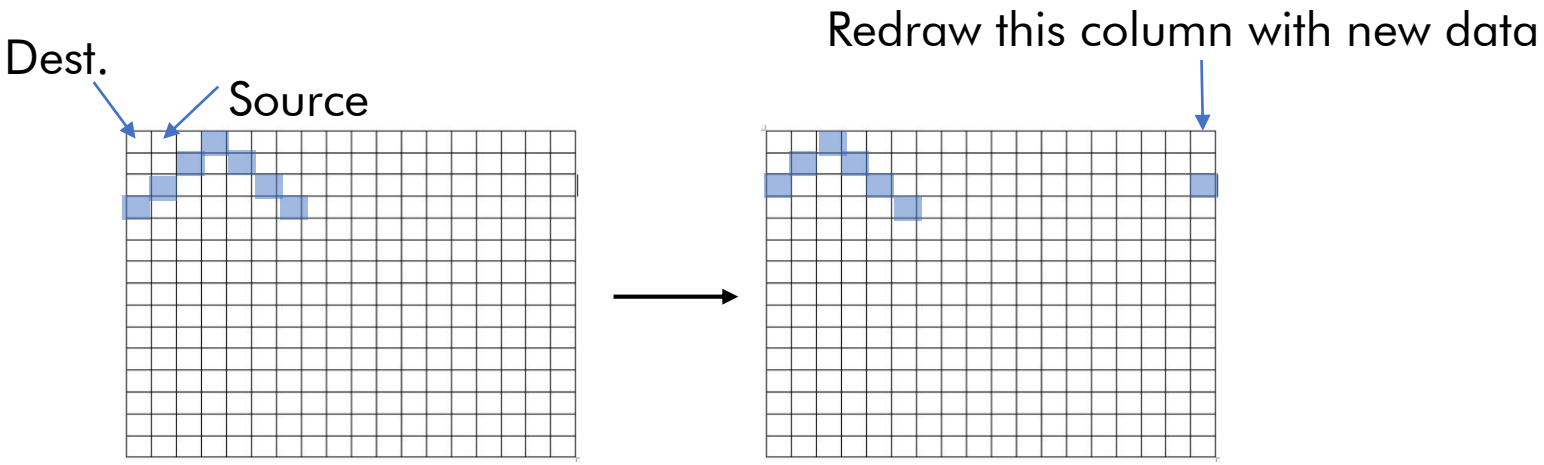
- Use memory-to-memory DMA for
  - Clear screen
  - Shift screen left
  - Shift screen up

- Clear screen:
  - Source address = address of value to clear screen with
  - Destination address = start of screen
  - Increment only destination address
  - Number of data = width*height

  - Increase performance further by using word (32-bit) transfers
  - Source address contains 4 pixels values with clear screen value (i.e. 0x0F0F0F0F)
  - Number of data = width*height/4

Emulator Display Optimizations
Emuleerder vertoon optimerings

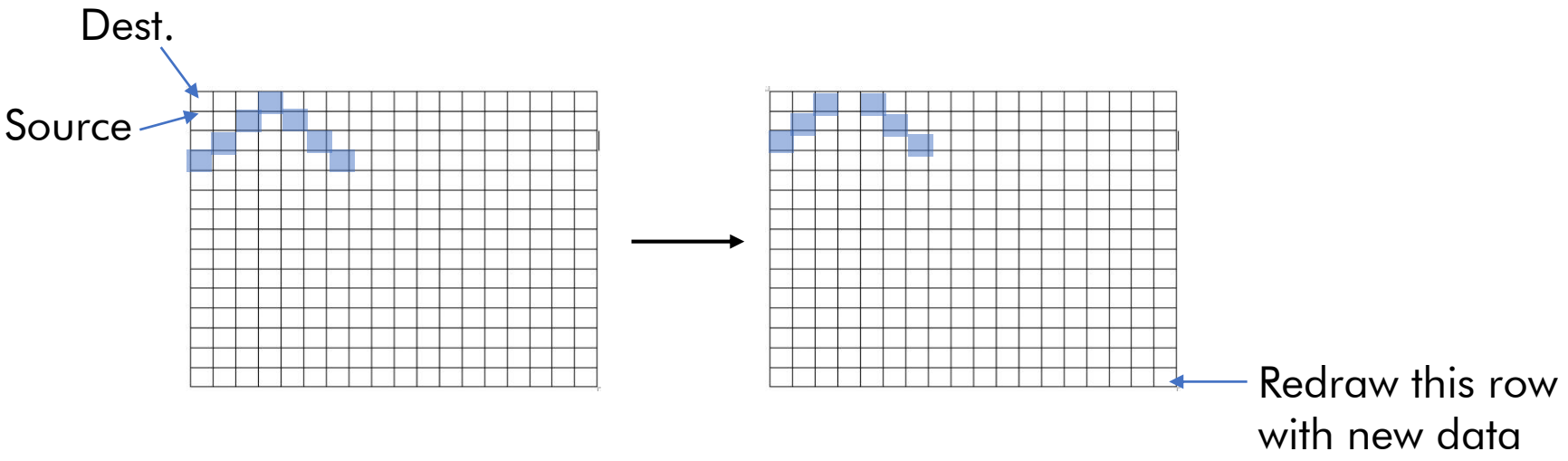
- Use memory-to-memory DMA to shift screen left:
  - Destination address = start of screen memory
  - Source address = start of screen memory + horizontal shift amount
  - DMA increment source and destination
  - Number of data = width*height – horizontal shift amount
  - Redraw the right-most columns



- Increase performance further by using 32-bit transfers
- Shift amount can only be multiples of 4
- Number of data = (width*height – horizontal shift amount)/4

Emulator Display Optimizations
Emuleerder vertoon optimerings

- Use memory-to-memory DMA to shift screen up:
  - Destination address = start of screen memory
  - Source address = start of screen memory + (vertical shift amount * width)
  - DMA increment source and destination
  - Number of data = width*(height – vertical shift amount)/4
  - (use 32-bit transfer)
  - Redraw the bottom rows



Emulator Display Optimizations
Emuleerder vertoon optimerings

- Using memory-to-memory DMA for display operations:
  - To know if the operation completed, use DMA interrupt:
  - Set ‘done = 1;’ flag in interrupt handler
  - Wait for done flag in main program