N-Body Accelerator

Isaac Trost (wit2102) Kristian Nikolov (kdn2117) Robert Pendergrast (rlp2153) Moises Mata (mm6155) Adib Khondoker (aak2250)

Project Overview

Background: N-Body calculations, used to determine the chaotic motions between *N* interacting masses, are critical in the fields of astrophysics, orbital dynamics, and molecular dynamics.

Problem: N-Body calculations are computationally expensive and SLOW, requiring many dividers and multipliers

GOAL: Develop a computational accelerator that can perform **N<512** Body simulations

Result... We fought the good fight and we came out victorious

High-Level System Design



nbody.sv

SW_READ_WRITE	Waits for software writes via bus (X,Y,VX,VY,M), IDLE STATE
CALC_ACCEL	Iterates body pairs, computes acceleration with getAccl, updates velocity
Update_POS	Update positions using Leapfrog step

GO, READ	Control Handshake
DONE	High when sim step completes
addr[15:9]	Select reg/mem
addr[8:0]	Select body index
write/readdata	32bit avalon bus IO

getaccel.sv

Signal	Width	Purpose
x1/y1	64	Position body 1
x2/y2	64	Position body 2
m2	64	Mass body 2 (pre mult by G)
ax/ay	64	Acc on body 1 from body 2
Clk, rst	1	Clock / synchronous reset

AddSub	3 units	Subtract, add
Mult	6 Units	Square, cube, scale
InvSqrt	1 unit	Inv sqr root
shift_reg	4 units	Align pipeline timing

MultTime	11 cycles
AddTime	20 cycles
InvSqrtTime	27 Cycles

- Computes
 - $\circ \quad \mathbf{Q}_{\mathbf{X}} = \mathbf{M}_{2} \cdot (\mathbf{X}_{1} \mathbf{X}_{2}) / |\mathbf{r}|^{3}$
 - $\circ \quad a_{\gamma} = m_2 \cdot (y_1 y_2) / |r|^3$

Accelerator Logic

- SW Read/Write
- Calculate Acceleration and update velocity
- Update position



Marrie

Calculating Acceleration and Velocity









•









Update Position



Hardware Validation

State O: Read/Write (Input)

2	🖄 🗸 (Msgs											
E	🖬 🥠 /nbodyTb/dut/state	2'h0	0			į							
Ð	🖪 🔙 /nbodyTb/dut/readdata	32'h00000000	0000000)/									
Ð	🖪 👉 /nbodyTb/dut/write_vy_data	1.00000	Х	5.30499e-313	1.23516e-322	1.0000	(+O	t 10.0000	-2.35344e	185 (0.100000	2.02567e-3	21 <u> </u>	+0
Ð	🖪 🧇 /nbodyTb/dut/write_vx_data	1.00000	Х	5.30499e-313	1.23516e-322	1.00000	(+0	t 10.0000	-2.35344e	185 (0.100000	2.02567e-3	41 <u> </u>	+0
. I	🖪 🧇 /nbodyTb/dut/write_m_data	1.00000	Х	5.30499e-313	1.23516e-322	1.00000	+0	t 10.0000	-2.35344e	185 (0.100000	2.02567e-3	41 <u>;</u>	+0
l I	🖪 🧇 /nbodyTb/dut/write_x_data	1.00000	Х	5.30499e-313	1.23516e-322	1.00000	+0	10.0000	-2.35344e	185 (0.100000	2.02567e-3	2 1 (+0
G	🖪 🧇 /nbodyTb/dut/write_y_data	1.00000	Х	5.30499e-313	1.23516e-322	1.0000	(+0	10.0000	-2.35344e	185 (0.100000	2.02567e-3	1 1	+0
G	🖪 🧇 /nbodyTb/dut/m_read_addr	9'h000	000										
G	🖪 🍫 /nbodyTb/dut/v_read_addr	9'h000	000										
G		9'h000	000			1							
G		9'h000	000			1							
E I	🖪 🧇 /nbodyTb/dut/v_write_addr	9'h000	000										
G	🖪 👍 /nbodyTb/dut/writedata	32'h3ff00000	0000001	9	00000000	3ff00000	000000	00 4024000	0 <u>9999999</u> a	, 3fb99999	00000000		
L.	🖬 🎻 /nbodyTb/dut/addr	16'h8a00	8400		8800	8a00	(8c00	(8e00	1 9400	, 9600	(9800	L I	9a00

State 1: Begin acceleration computation

∕	Msgs		12																					
📆 🤣 /nbodyTb/dut/state	2'h0	0 <u>)</u> 1																			E			
📻 👍 /nbodyTb/dut/readdata	32'h0000000	0000000																			i se i r			
🕳 🤣 /nbodyTb/dut/write_vy_data	1.00000	<u> </u>																						
🕳 🤣 /nbodyTb/dut/write_vx_data	1.00000	<u> </u>							<u> 10.100000 (</u>)														
庄 🤣 /nbodyTb/dut/write_m_data	1.00000	<u> </u>																	ر کے					
🕳 🧇 /nbodyTb/dut/write_x_data	1.00000	<u> </u>																	ر کے					
😐 🧇 /nbodyTb/dut/write_y_data	1.00000	<u>) 2.12) +0</u>												1										
	9'h000	000																						<u>(001</u>
	9'h000	000																						
//indextstyle="color: blue;">	9'h000	000																						<u> (001</u>
//ibodyTb/dut/pos_input_2_addr	9'h000	000	<u>1001 10</u>	02 <u>1</u> 003	<u>1 004</u>	(005	1 006	<u>1007 100</u>	<u>8 1,009 1</u>	00a	1,00b 1,0)0d (00d <u>†</u> 00	<u>X 00f</u>	<u> 1</u> 010	<u>t 011</u>	<u>) 012</u>	<u>t 013</u>	<u>X 014</u>	<u> 1 015</u>	<u>X 016</u>	<u>∤</u> 017	<u> 1 018</u>	<u> 1 000 t</u>
	9'h000	000																						
🕳 🍫 /nbodyTb/dut/writedata	32'h3ff00000	0000001																						
🕒 🤣 /nbodyTb/dut/addr	16'h8a00	8000																						

State 2: Begin position update

		Msgs																											
		2h1	1			<u>1</u> 2															The second s						ر کے		
	📻 👍 /nbodyTb/dut/readdata	32'h00000000	00000000																		i de la compañía de				i de la compañía de la		i de la compañía de		
	🖅 🤣 /nbodyTb/dut/write_vy_data	3.97724	3.97724			(+0															a se				j		i and p		
	📻 🤣 /nbodyTb/dut/write_vx_data	2.67636	2.67636			<u>χ</u> +0															i and i				i i i		i an i f		
	🖅 - 🤣 /nbodyTb/dut/write_m_data	+0	+0																								_		
	🖅 - 🍫 /nbodyTb/dut/write_x_data	+0	+0			(1.93	1.18	. <u>(</u> -1.4)	2.6/6	36							<u> (</u> 2.93.	(-3.8	<u> 18.5</u>	(2.67	636				ر کے				
		+0	+0			(-1.3	1.29	. (0.55)	3.9772	24							<u> (8.62</u>		<u>) (0.55</u>	(3.97	/24						رهم		
		9'h008	01b			<u>(000 </u>																							
	/nbodyTb/dut/v_read_addr	9'h015	(007 (00	8 <u>1</u> 009	<u>(00a</u>	<u>(000 </u>	001	<u>, 002</u>	003	<u>1 004</u>	1005	<u>(</u> 006	<u> 1</u> 007	<u> (008</u>	<u>1 009</u>	(00a	<u>1 00b</u>	<u>t 00c</u>	<u>(</u> 00d	<u>t 00e</u>	<u>) 100 X</u>	<u> </u>	<u> (011</u>	<u>(012</u>	<u> (013</u>	<u>(014</u>	<u> (</u> 015	<u>X 016</u>	٦Į
	/nbodyTb/dut/pos_input_1_addr	9/h008	01b			<u>(000 </u>												(001	<u>(</u> 002	<u>(003</u>	<u>) 004</u>	<u>) 005</u>	<u> 1 006 </u>	<u>(</u> 007	<u> 1 008</u>	<u>(009</u>	<u> (00a</u>	<u> (00b</u>	ΞĮ.
	/nbodyTb/dut/pos_input_2_addr	9'h007	(<u>012) (01</u>	<u>3 (014</u>	<u>X 015</u>	<u>(000)</u>	001	<u>, 002</u>	003	<u>t 004</u>	<u> 1 005</u>	<u>(</u> 006	<u> 1 007</u>	<u> (008</u>	1 009	(00a	<u>X 00b</u>	<u>(00c</u>	<u>X 00d</u>	<u>(00e</u>	<u>) 100 y</u>	<u>) (010</u>	<u> (011</u>	<u> 1 012</u>	<u>(013</u>	<u>X 014</u>	<u> (015</u>	<u>X 016</u>	٦Ľ
	/nbodyTb/dut/v_write_addr	9'h00a	(<u>015 </u>	6 <u>1017</u>	<u>X 018</u>	<u> 1 000 j</u>																							
	🖅 🍫 /nbodyTb/dut/writedata	32'h0000001	00000001																		ہ ہے						ہ ہے		
	🛨 🍫 /nbodyTb/dut/addr	16°h9000	8000																										
L	• · · · · · · · · · · · · · · · · · · ·																												

State O: Reading output values

	<u> </u>	Msgs													
1	🕳 🛷 /nbodyTb/dut/state	2'h0	0												
	📻 👍 /nbodyTb/dut/readdata	32'h0878d75c	00000000		(0878d Sc	X 408351c6	t d25215c5	t c06cf111	(0878d75c	<u>X 4046a89b</u>	<u>)</u> f87a7f89	40432212	<u>(9d301def</u>	t c072e171	
1	📻 🤣 /nbodyTb/dut/write_vy_data	4.94066e-324	2.12200e-314		<u>(4.94066e-3</u>	24									
ł	🖶 🤣 /nbodyTb/dut/write_vx_data	4.94066e-324	2.12200e-314		(4.9406 <mark>6e-3</mark>	24									
	🖶 🤣 /nbodyTb/dut/write_m_data	4.94066e-324	2.12200e-314		(4.9406 <mark>6e-3</mark>	24									
1	📻 🤣 /nbodyTb/dut/write_x_data	4.94066e-324	2.12200e-314		(4.94066e-3	24									
	📻 🤣 /nbodyTb/dut/write_y_data	4.94066e-324	2.12200e-314		<u>(4.94066e-3</u>	24									
	/nbodyTb/dut/m_read_addr	9'h000	000		ļ.				<u> 1 001</u>				<u> (</u> 002		
ł		9'h000	000						<u> (001</u>				<u> (</u> 002		
	//ibodyTb/dut/pos_input_1_addr	9'h000	000						<u> 1 001</u>				<u>)</u> 002		
	//ibodyTb/dut/pos_input_2_addr	9'h000	000		ļ.				<u> 1 001</u>				<u> (</u> 002		
		9'h000	000						<u> (001</u>				<u> (</u> 002		
	🖶 🍲 /nbodyTb/dut/writedata	32'h0000000	00000001		<u> (000000</u> 00										
	🖶 🍫 /nbodyTb/dut/addr	16'ha200	8000	<u>X 8200</u>	<u>(a200</u>	<u>(</u> a400	<u>) a600</u>	<u>(</u> a800	<u>(</u> a201	<u>(</u> a401	<u> (</u> a601	<u>X</u> a801	<u>(</u> a202	<u>, a402</u>	Ľέ
	/nbodyTb/dut/read_sw	1'h1													
	/nbodyTb/dut/go	1'h1													
	/nbodyTb/dut/done	1'h1													
- 8	K Star Star Star Star Star Star Star Star														

Testbench Final Results

Time=	62575: Te
For body nbodyTb	0
X = 618.221696	
Y = -231.533425	
For body nbodyTb	1
X = 45.317231	
Y = 38.266204	
For body nbodyTb	2
X = -302.090238	
Y = 108.575161	
Time=	122835: Te
For body nbodyTb	0
X = 1529.887500	
Y = -583.881837	
For body nbodyTb	1
X = 112.031482	
Y = 109.574936	
For body nbodyTb	2
X = -775.501705	
Y = 268.047184	
** Note: \$finish	: nbodyTh
Time: 123095 ps	Iteration

Software

Accelerator Software Interface

32-bit Memory-Mapped Interface with **16 Bit addresses**:

OOOOOOOOOOOOOOOO-> 9 Bits for Body Number 0-511ParameterBody Number

Parameter Mappings:

Ox4C - VY_LOW Ox4D - VY_UPPER Read Addresses:

0x44 - X_LOW	Ox48 - M_LOW	0x51 - Read_X_LOW
0x45 - X_UPPER	Ox49 - M_UPPER	0x52 - Read_X_UPPER
0x46 - Y_LOW	Ox4A - VX_LOW	0x53 - Read_Y_LOW
0x47 - Y_UPPER	Ox4B - VX_UPPER	0x54 - Read_Y_UPPER

EX: #define X_ADDR_LOW(base, body) (base) + ((body<<2) + (0x44 << 11))

Accelerator Driver

WRITE POSITIONS	WRITE PARAMETERS	READ POSITIONS
typedef struct {	typedef struct {	typedef struct{
double x, y, vx, vy,	<pre>int N;</pre>	double x, y;
m ;	int gap;	int n;
int n;	<pre>} nbody_sim_config_t;</pre>	<pre>} body_pos_t;</pre>
} body t;		

NOTE - Positions and parameters are written and read as upper 32 and lower 32 bits separately:

```
memcpy(&x_bits, &body_parameters->x, sizeof(uint64_t));
iowrite32(x_bits[0], X_ADDR_LOW(dev.virtbase, i));
iowrite32(x_bits[1], X_ADDR_HIGH(dev.virtbase, i));
```

Control Flow



User Logic

- Accelerator reads selected input CSV from userspace.
- Simulation parameters determined by user
 - Gap & Iterations
- After acceleration, positions for each iteration is written to output csv file

[screen 0: ttyUSB0] File Edit View Search Terminal Help Iteration 130 complete! Iteration 140 complete! Iteration 160 complete! Iteration 160 complete! Iteration 170 complete! Iteration 180 complete! Iteration 190 complete! Results saved to nbody_results.csv N-Body Userspace program terminating root@del-soc:-/EmbeddedFinalProject/software/final# ./nbody

Welcome to the N-Body Simulation Final Project!

Please select a setting for how many bodies you would like to simulate: 1 - 32 Bodies 2 - 64 Bodies 3 - 128 Bodies 4 - 512 Bodies

But wait! There's more

Display Hardware Interface

Memory mapped VGA Frame Buffer Module:

32 bit words accessed through 15 bit memory addresses, which maps to 32 pixels on the screen

- The read address passed on to memory is calculated as so below, and uses both vcount and hcount with an appropriate offset, to ensure the address is passed to memory in time



Display Software Interface

Kernel Module with specific functions to facilitate N-Body animations

IOCTLS

WRITE_PROPERTIES - Copies the current state for all bodies in the timestep from user to kernel space. Writes each body as a circle into the framebuffer, which is written to memory at the end.

CLEAR_SCREEN - Clears both the virtual framebuffer and the corresponding area in memory

FILL_SCREEN - Writes a pattern directly to framebuffer memory. Used in debugging to display a

filled screen, checker pattern.



Display Software Interface

Userspace program that allows playback from CSV with adjustable play speed

```
static void convert_coordinates(float
nbody_x, float nbody_y,short
*display x, short *display y) {
```

```
*display_x = (short)((nbody_x +
500.0) / 1000.0 * (DISPLAY_WIDTH));
    *display_y = (short)((nbody_y +
500.0) / 1000.0 * (DISPLAY_HEIGHT));
```

}

After this coordinate conversion, values are sent to the driver through the ioctl call!



