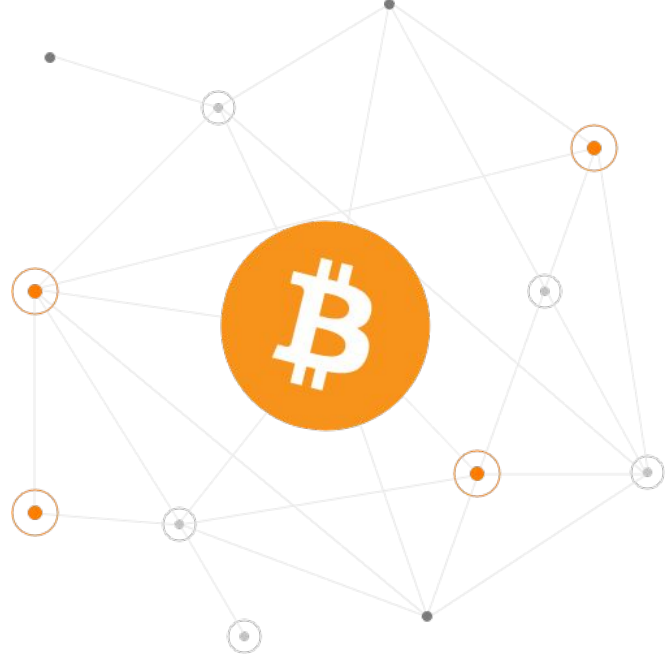
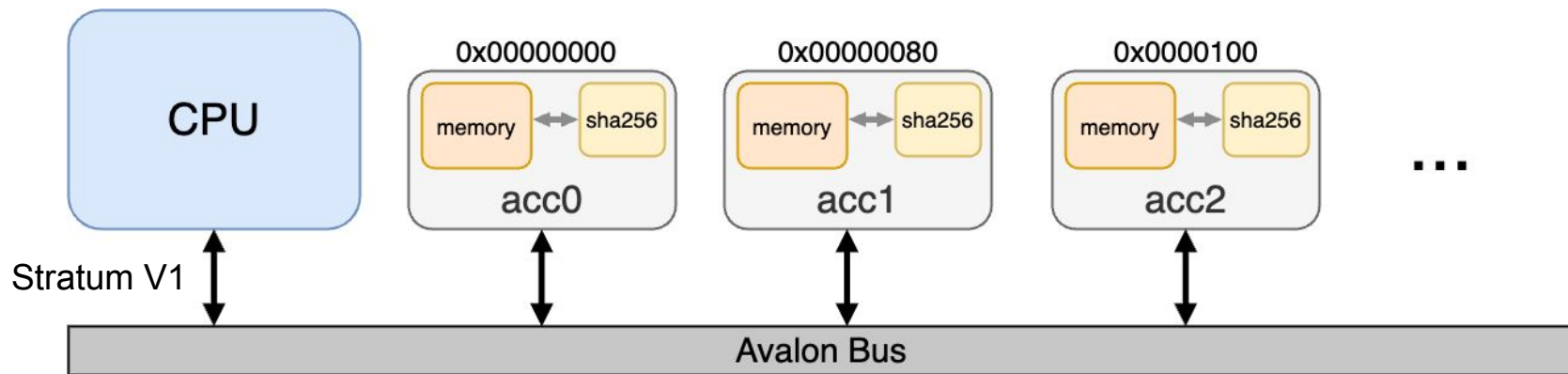


GoldMiner

Jules Comte
Mingyang Song
Zhe Mo
Tianyu Qin
Xueji Zhao



Overview of Project Design



Message Extracted from Mining Pool

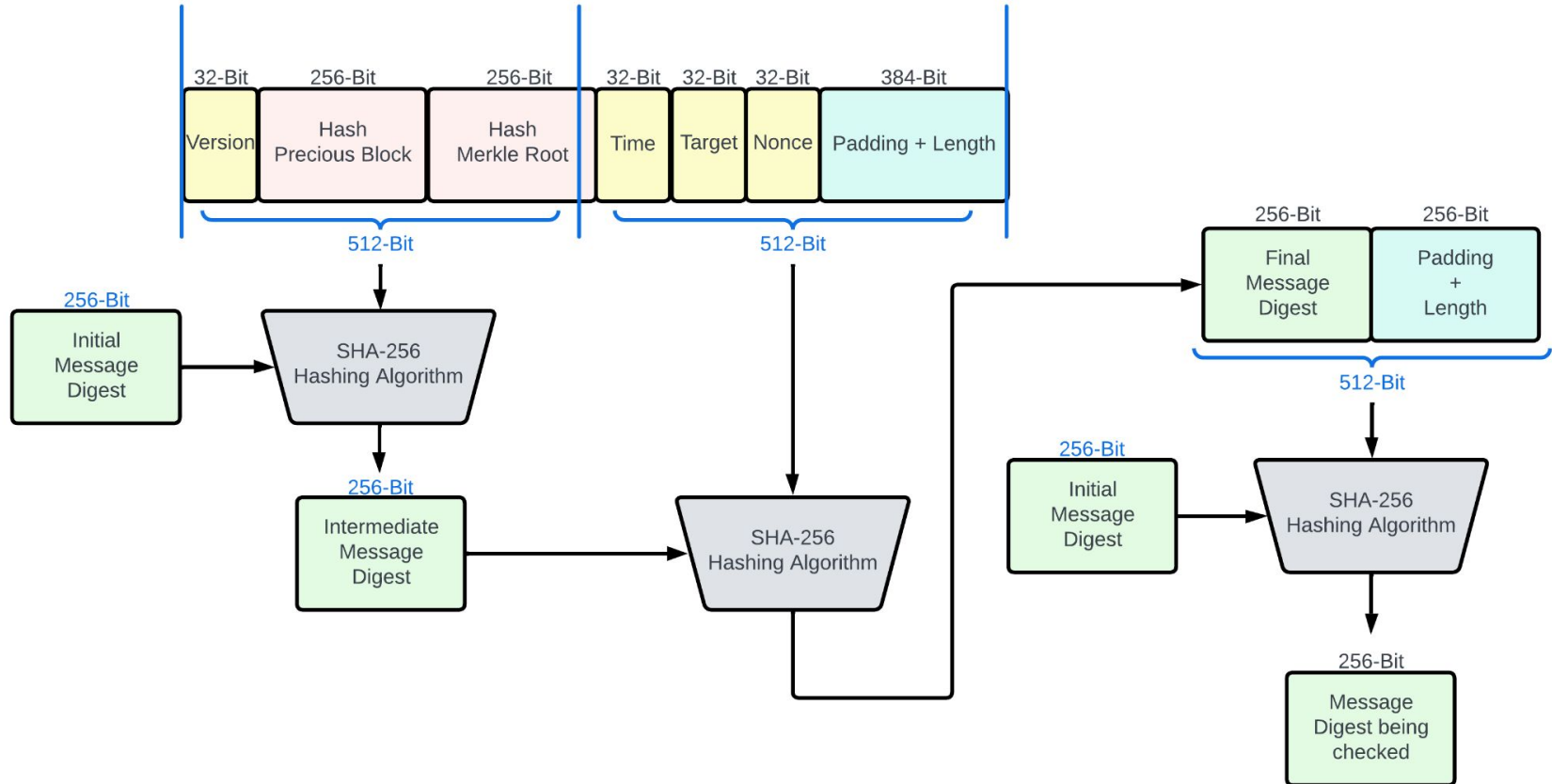
- Extranonce1
- Extranonce2 size

- id
- methods
- params
 - job id
 - ...
 - Coinbase Transaction Part 1
 - Coinbase Transaction Part 2

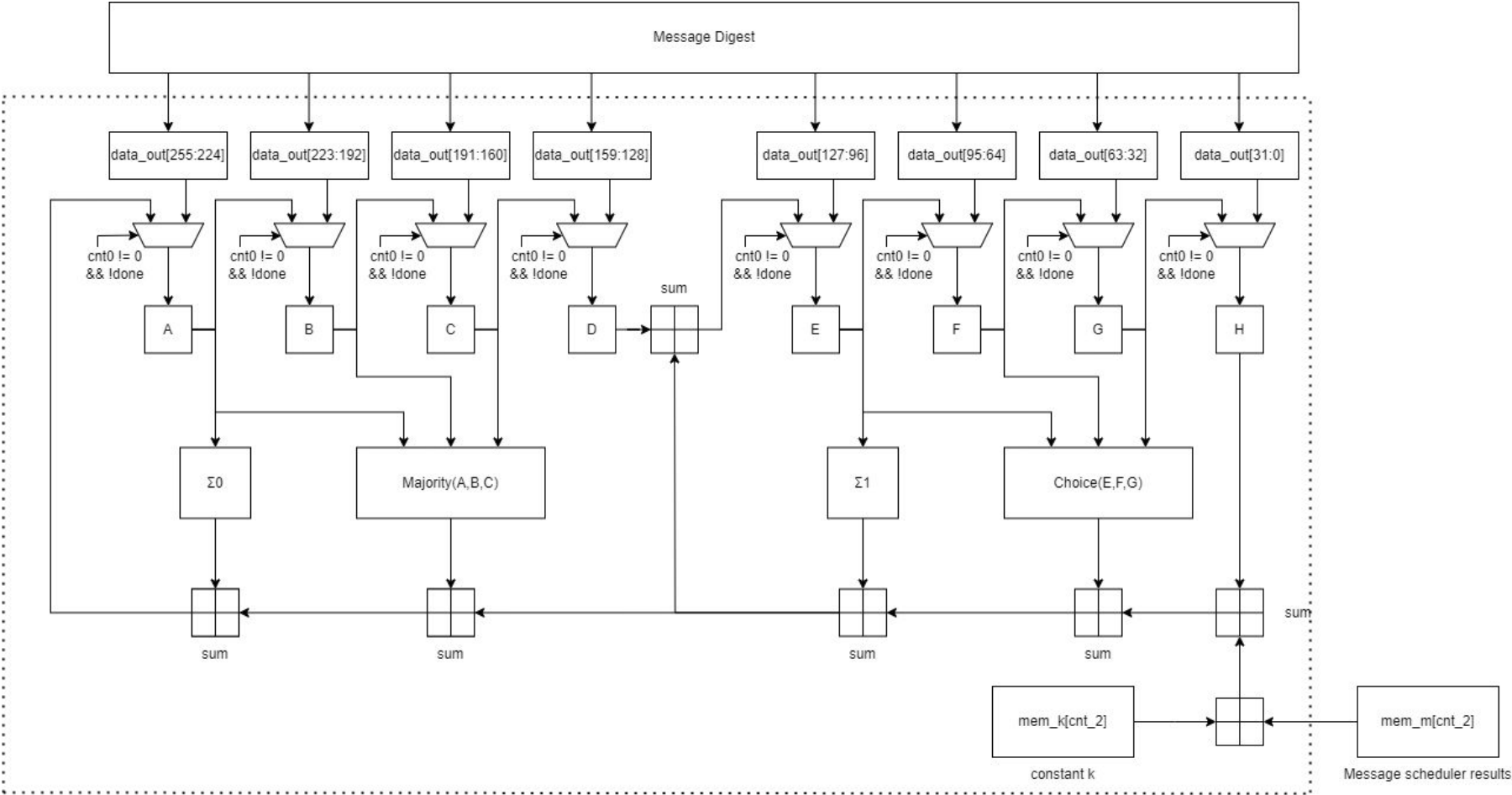
```

00  pub(crate) fn new_target(&mut self, target: Uint256) {
01      //
02      //
03      //
04      //
05      //
06      //
07      //
08      //
09      //
10      //
11      //
12      //
13      //
14      //
15      //
16      //
17      //
18      //
19      //
20      //
21      //
22      //
23      //
24      //
25      //
26      //
27      //
28      //
29      //
30      //
31      //
32      //
33      //
34      //
35      //
36      //
37      //
38      //
39      //
40      //
41      //
42      //
43      //
44      //
45      //
46      //
47      //
48      //
49      //
50      //
51      //
52      //
53      //
54      //
55      //
56      //
57      //
58      //
59      //
60      //
61      //
62      //
63      //
64      //
65      //
66      //
67      //
68      //
69      //
70      //
71      //
72      //
73      //
74      //
75      //
76      //
77      //
78      //
79      //
80      //
81      //
82      //
83      //
84      //
85      //
86      //
87      //
88      //
89      //
90      //
91      //
92      //
93      //
94      //
95      //
96      //
97      //
98      //
99      //
100     //
101     //
102     //
103     //
104     //
105     //
106     //
107     //
108     //
109     //
110     //
111     //
112     //
113     //
114     //
115     //
116     //
117     //
118     //
119     //
120     //
121     //
122     //
123     //
124     //
125     //
126     //
127     //
128     //
129     //
130     //
131     //
132     //
133     //
134     //
135     //
136     //
137     //
138     //
139     //
140     //
141     //
142     //
143     //
144     //
145     //
146     //
147     //
148     //
149     //
150     //
151     //
152     //
153     //
154     //
155     //
156     //
157     //
158     //
159     //
160     //
161     //
162     //
163     //
164     //
165     //
166     //
167     //
168     //
169     //
170     //
171     //
172     //
173     //
174     //
175     //
176     //
177     //
178     //
179     //
180     //
181     //
182     //
183     //
184     //
185     //
186     //
187     //
188     //
189     //
190     //
191     //
192     //
193     //
194     //
195     //
196     //
197     //
198     //
199     //
200     //
201     //
202     //
203     //
204     //
205     //
206     //
207     //
208     //
209     //
210     //
211     //
212     //
213     //
214     //
215     //
216     //
217     //
218     //
219     //
220     //
221     //
222     //
223     //
224     //
225     //
226     //
227     //
228     //
229     //
230     //
231     //
232     //
233     //
234     //
235     //
236     //
237     //
238     //
239     //
240     //
241     //
242     //
243     //
244     //
245     //
246     //
247     //
248     //
249     //
250     //
251     //
252     //
253     //
254     //
255     //
256     //
257     //
258     //
259     //
260     //
261     //
262     //
263     //
264     //
265     //
266     //
267     //
268     //
269     //
270     //
271     //
272     //
273     //
274     //
275     //
276     //
277     //
278     //
279     //
280     //
281     //
282     //
283     //
284     //
285     //
286     //
287     //
288     //
289     //
290     //
291     //
292     //
293     //
294     //
295     //
296     //
297     //
298     //
299     //
300     //
301     //
302     //
303     //
304     //
305     //
306     //
307     //
308     //
309     //
310     //
311     //
312     //
313     //
314     //
315     //
316     //
317     //
318     //
319     //
320     //
321     //
322     //
323     //
324     //
325     //
326     //
327     //
328     //
329     //
330     //
331     //
332     //
333     //
334     //
335     //
336     //
337     //
338     //
339     //
340     //
341     //
342     //
343     //
344     //
345     //
346     //
347     //
348     //
349     //
350     //
351     //
352     //
353     //
354     //
355     //
356     //
357     //
358     //
359     //
360     //
361     //
362     //
363     //
364     //
365     //
366     //
367     //
368     //
369     //
370     //
371     //
372     //
373     //
374     //
375     //
376     //
377     //
378     //
379     //
380     //
381     //
382     //
383     //
384     //
385     //
386     //
387     //
388     //
389     //
390     //
391     //
392     //
393     //
394     //
395     //
396     //
397     //
398     //
399     //
400     //
401     //
402     //
403     //
404     //
405     //
406     //
407     //
408     //
409     //
410     //
411     //
412     //
413     //
414     //
415     //
416     //
417     //
418     //
419     //
420     //
421     //
422     //
423     //
424     //
425     //
426     //
427     //
428     //
429     //
430     //
431     //
432     //
433     //
434     //
435     //
436     //
437     //
438     //
439     //
440     //
441     //
442     //
443     //
444     //
445     //
446     //
447     //
448     //
449     //
450     //
451     //
452     //
453     //
454     //
455     //
456     //
457     //
458     //
459     //
460     //
461     //
462     //
463     //
464     //
465     //
466     //
467     //
468     //
469     //
470     //
471     //
472     //
473     //
474     //
475     //
476     //
477     //
478     //
479     //
480     //
481     //
482     //
483     //
484     //
485     //
486     //
487     //
488     //
489     //
490     //
491     //
492     //
493     //
494     //
495     //
496     //
497     //
498     //
499     //
500     //
501     //
502     //
503     //
504     //
505     //
506     //
507     //
508     //
509     //
510     //
511     //
512     //
513     //
514     //
515     //
516     //
517     //
518     //
519     //
520     //
521     //
522     //
523     //
524     //
525     //
526     //
527     //
528     //
529     //
530     //
531     //
532     //
533     //
534     //
535     //
536     //
537     //
538     //
539     //
540     //
541     //
542     //
543     //
544     //
545     //
546     //
547     //
548     //
549     //
550     //
551     //
552     //
553     //
554     //
555     //
556     //
557     //
558     //
559     //
560     //
561     //
562     //
563     //
564     //
565     //
566     //
567     //
568     //
569     //
570     //
571     //
572     //
573     //
574     //
575     //
576     //
577     //
578     //
579     //
580     //
581     //
582     //
583     //
584     //
585     //
586     //
587     //
588     //
589     //
590     //
591     //
592     //
593     //
594     //
595     //
596     //
597     //
598     //
599     //
600     //
601     //
602     //
603     //
604     //
605     //
606     //
607     //
608     //
609     //
610     //
611     //
612     //
613     //
614     //
615     //
616     //
617     //
618     //
619     //
620     //
621     //
622     //
623     //
624     //
625     //
626     //
627     //
628     //
629     //
630     //
631     //
632     //
633     //
634     //
635     //
636     //
637     //
638     //
639     //
640     //
641     //
642     //
643     //
644     //
645     //
646     //
647     //
648     //
649     //
650     //
651     //
652     //
653     //
654     //
655     //
656     //
657     //
658     //
659     //
660     //
661     //
662     //
663     //
664     //
665     //
666     //
667     //
668     //
669     //
670     //
671     //
672     //
673     //
674     //
675     //
676     //
677     //
678     //
679     //
680     //
681     //
682     //
683     //
684     //
685     //
686     //
687     //
688     //
689     //
690     //
691     //
692     //
693     //
694     //
695    
```

Overview of SHA-256 Process



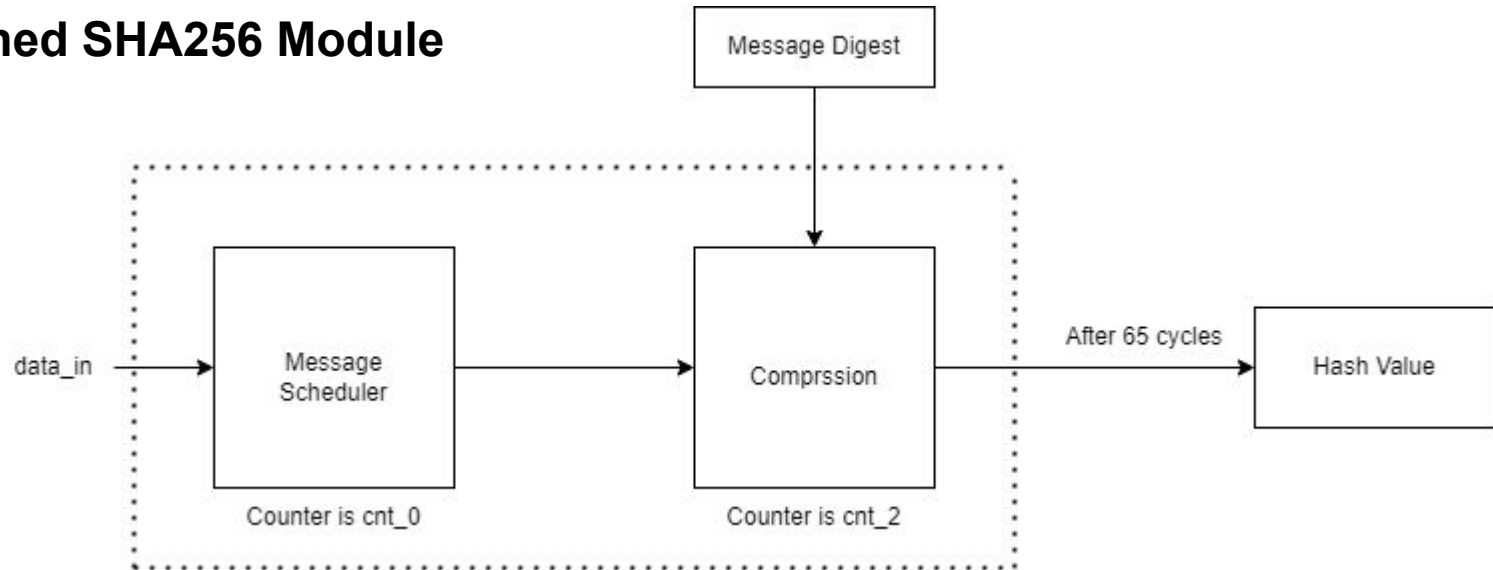
Compression



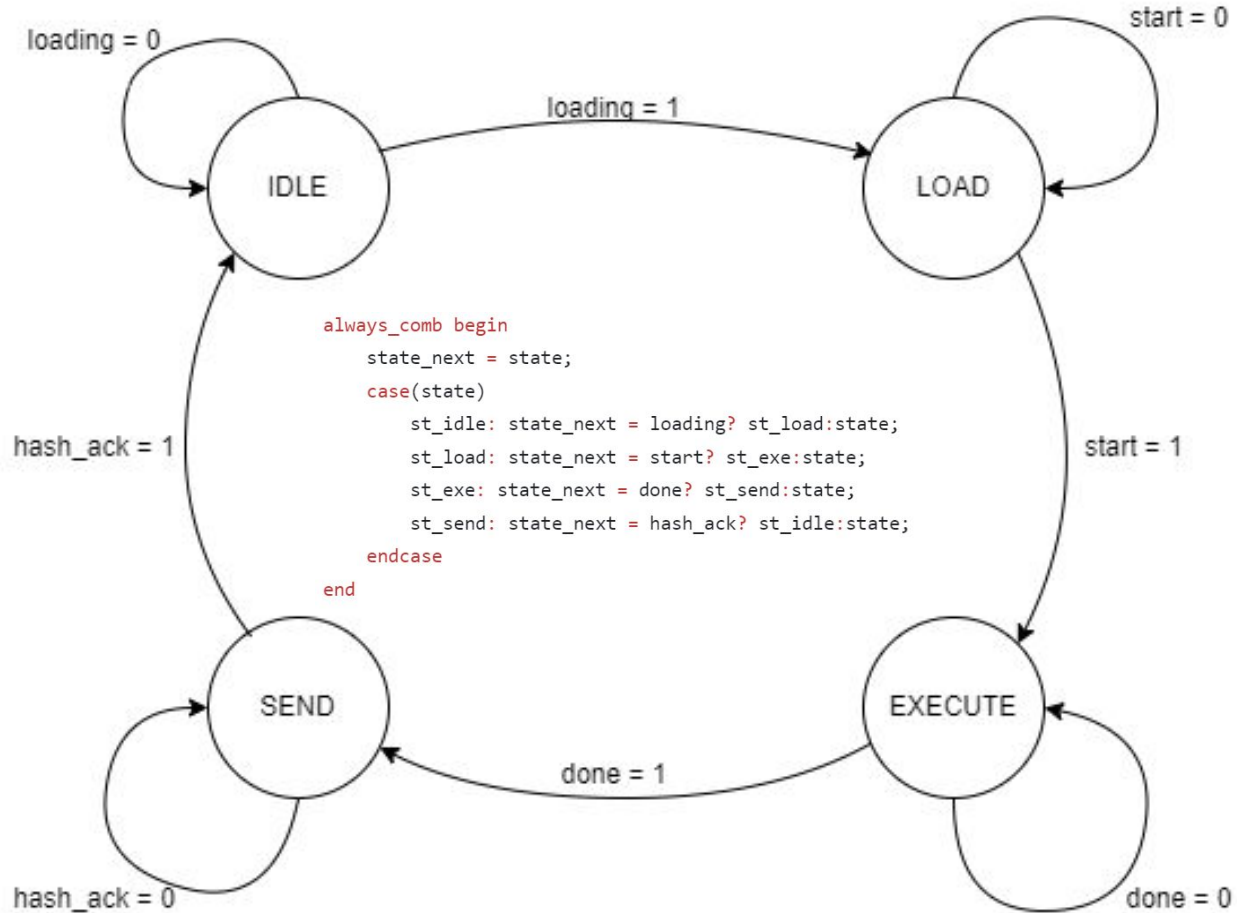
Message Scheduler

```
if(cnt_0 < 16){  
    mem_m[cnt_0] reads from input data  
}  
else if(cnt_0 < 64){  
    mem_m[cnt_0] = sig1(mem_m[cnt_0-2])+mem_m[cnt_0-7]+sig0(mem_m[cnt_0-5])+mem_m[cnt_0-16]  
}
```

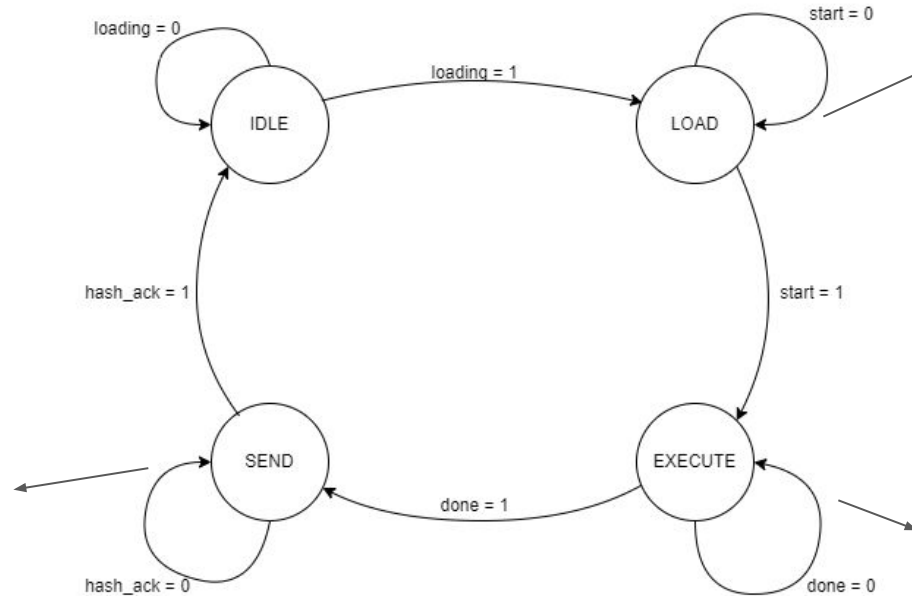
Combined SHA256 Module



Hardware Finite State Machine



Hardware Finite State Machine



```

always_ff @ (posedge clk)
  if (reset)
    data_out <= 0;
  else
    if (state == st_send && reading)
      case(address)
        0: data_out <= hashvalue[31:0];
        1: data_out <= hashvalue[63:32];
        2: data_out <= hashvalue[95:64];
        3: data_out <= hashvalue[127:96];
        4: data_out <= hashvalue[159:128];
        5: data_out <= hashvalue[191:160];
        6: data_out <= hashvalue[223:192];
        7: data_out <= hashvalue[255:224];
        17: data_out <= 32'h11111111;
        default: data_out <= 2;
      endcase
    else
      data_out <= 32'h0f0ff0f0;
  end
  
```

```

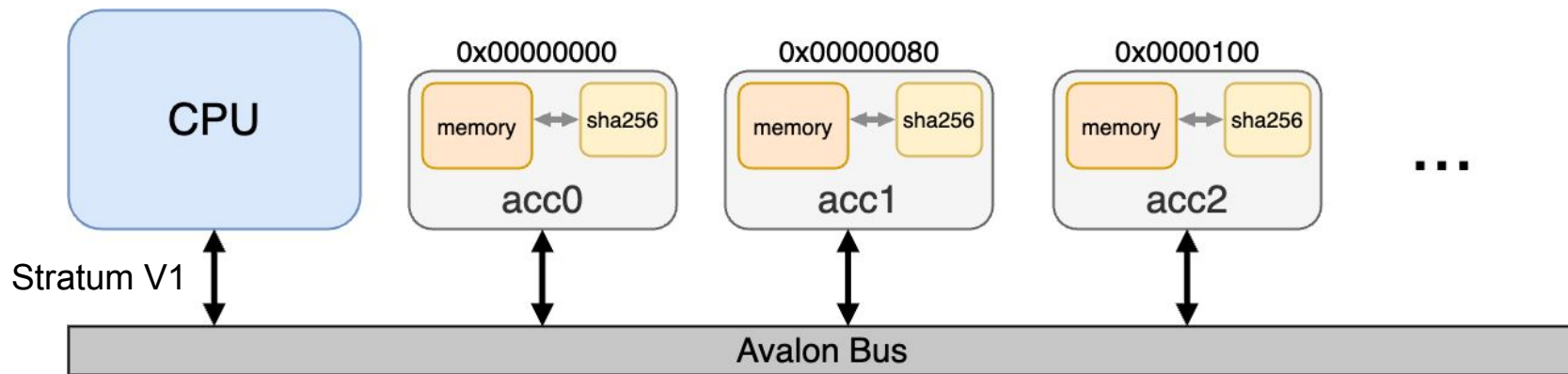
always_ff @ (posedge clk) begin
  if (reset) buffer <= 0;
  else
    if (loading)
      case(address)
        0: buffer[31:0] <= writedata;
        1: buffer[63:32] <= writedata;
        2: buffer[95:64] <= writedata;
        3: buffer[127:96] <= writedata;
        4: buffer[159:128] <= writedata;
        5: buffer[191:160] <= writedata;
        6: buffer[223:192] <= writedata;
        7: buffer[255:224] <= writedata;
        8: buffer[287:256] <= writedata;
        9: buffer[319:288] <= writedata;
        10: buffer[351:320] <= writedata;
        11: buffer[383:352] <= writedata;
        12: buffer[415:384] <= writedata;
        13: buffer[447:416] <= writedata;
        14: buffer[479:448] <= writedata;
        15: buffer[511:480] <= writedata;
      endcase
    end
  end
  
```

```

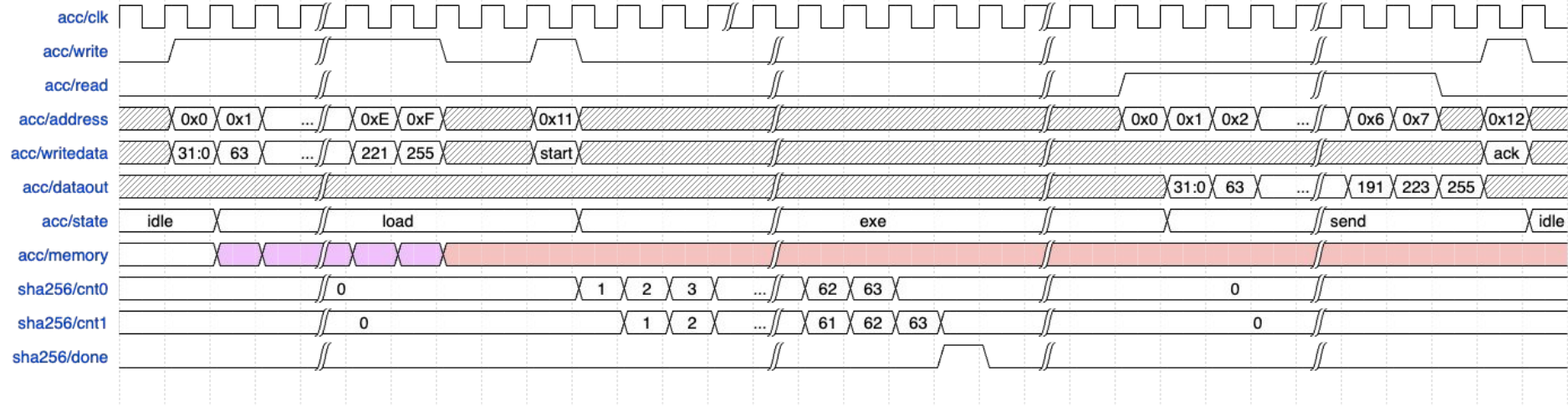
/**** Module ports map ****/
sha256_module sha256_module_0(
  .clk(clk),
  .reset(acc_reset),
  .start(start),
  .data_in(buffer),
  .data_out(hashvalue),
  .done(done)
);

/*****
  *****/
  
```

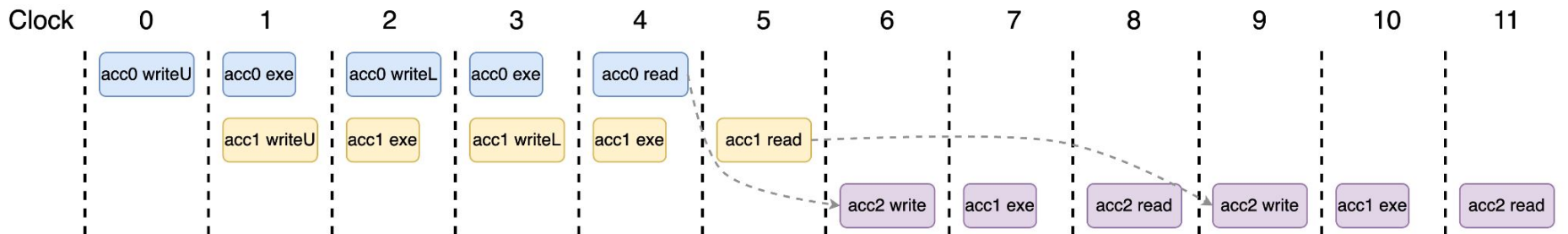

Overview of Project Design



Hardware Timing Diagram



Hardware Pipeline



Speedup: Approximately 25%

HW vs SW

85% Speedup

```
root@del-soc:~/sw# ./a.out s
Software SHA256
(SOFTWARE)Time elapsed: 0.867471 seconds
----- OUTPUT OF software block0 -----
hash_output.r0: f260764c
hash_output.r1: 2ae540d6
hash_output.r2: 437a88a6
hash_output.r3: d9001e2e
hash_output.r4: 8fa8f8a8
hash_output.r5: bd650000
hash_output.r6: 0
hash_output.r7: 0
----- OUTPUT OF software block1 -----
hash_output.r0: ceb1e4e8
hash_output.r1: 3bcfa266
hash_output.r2: 97eef0d3
hash_output.r3: 55633391
hash_output.r4: 1061f23f
hash_output.r5: 7b77dc15
hash_output.r6: 0
hash_output.r7: 0
root@del-soc:~/sw# ./a.out h
Hardware SHA256
(HARDWARE)Time elapsed: 0.468272 seconds
----- OUTPUT OF ACC0 -----
hash_output0.r0: f260764c
hash_output0.r1: 2ae540d6
hash_output0.r2: 437a88a6
hash_output0.r3: d9001e2e
hash_output0.r4: 8fa8f8a8
hash_output0.r5: bd650000
hash_output0.r6: 0
hash_output0.r7: 0
----- OUTPUT OF ACC1 -----
hash_output1.r0: ceb1e4e8
hash_output1.r1: 3bcfa266
hash_output1.r2: 97eef0d3
hash_output1.r3: 55633391
hash_output1.r4: 1061f23f
hash_output1.r5: 7b77dc15
hash_output1.r6: 0
hash_output1.r7: 0
```

```

90 sha256_transform:
91     @ args = 0, pretend = 0, frame = 320
92     @ frame_needed = 1, uses_anonymous_args = 0
93     push    {r7, lr}
94     sub sp, sp, #320
95     add r7, sp, #0
96     adds    r3, r7, #4
97     str r0, [r3]
98     mov r3, r7
99     str r1, [r3]
100    movw    r3, #::lower16:__stack_chk_guard
101    movt    r3, #::upper16:__stack_chk_guard
102    ldr r3, [r3]
103    str r3, [r7, #316]
104    add r3, r7, #44
105    movs    r2, #0
106    str r2, [r3]
107    add r3, r7, #48
108    movs    r2, #0
109    str r2, [r3]
110    b       .L2
111 .L3:
112    mov r2, r7
113    add r3, r7, #48
114    ldr r2, [r2]
115    ldr r3, [r3]
116    add r3, r3, r2
117    ldrb    r3, [r3]    @ zero_extendqisi2
118    mov r1, r3
119    add r3, r7, #48
120    ldr r3, [r3]
121    adds    r3, r3, #1
122    mov r2, r7
123    ldr r2, [r2]
124    add r3, r3, r2
125    ldrb    r3, [r3]    @ zero_extendqisi2
126    lsls    r3, r3, #8
127    orr r2, r1, r3
128    add r3, r7, #48
129    ldr r3, [r3]
130    adds    r3, r3, #2
131    mov r1, r7
132    ldr r1, [r1]
133    add r3, r3, r1

```

Multi-Device Project

Project goal: 3 independent sha256 devices all managed by a single driver.

Connections	Name	Description	Export	Clock	Base	End	IRQ
	<input type="checkbox"/> clk_0	Clock Source					
	clk_in	Clock Input	clk	exported			
	clk_in_reset	Reset Input	reset				
	clk	Clock Output	<i>Double-click to</i>	clk_0			
	clk_reset	Reset Output	<i>Double-click to</i>				
	<input type="checkbox"/> hps_0	Arria V/Cyclone V Hard Proce...					
	h2f_user1_clock	Clock Output	<i>Double-click to</i>	hps_0_h2...			
	memory	Conduit	hps_ddr3				
	hps_io	Conduit	hps				
	h2f_reset	Reset Output	<i>Double-click to</i>				
	h2f_axi_clock	Clock Input	<i>Double-click to</i>	clk_0			
	h2f_axi_master	AXI Master	<i>Double-click to</i>	[h2f_axi_...			
	f2h_axi_clock	Clock Input	<i>Double-click to</i>	clk_0			
	f2h_axi_slave	AXI Slave	<i>Double-click to</i>	[f2h_axi_...			
	h2f_lw_axi_clock	Clock Input	<i>Double-click to</i>	clk_0			
	h2f_lw_axi_master	AXI Master	<i>Double-click to</i>	[h2f_lw_a...			
	f2h_irq0	Interrupt Receiver	<i>Double-click to</i>		IRQ 0	IRQ 31	←×
	f2h_irq1	Interrupt Receiver	<i>Double-click to</i>		IRQ 0	IRQ 31	←×
	<input type="checkbox"/> vga_ball_0	VGA Ball					
	clock	Clock Input	<i>Double-click to</i>	clk_0			
	reset	Reset Input	<i>Double-click to</i>	[clock]			
	avalon_slave_0	Avalon Memory Mapped Slave	<i>Double-click to</i>	[clock]	<input type="checkbox"/> 0x0000_0000	0x0000_007f	
	<input type="checkbox"/> vga_ball_1	VGA Ball					
	clock	Clock Input	<i>Double-click to</i>	clk_0			
	reset	Reset Input	<i>Double-click to</i>	[clock]			
	avalon_slave_0	Avalon Memory Mapped Slave	<i>Double-click to</i>	[clock]	<input type="checkbox"/> 0x0000_0100	0x0000_017f	
	<input type="checkbox"/> vga_ball_2	VGA Ball					
	clock	Clock Input	<i>Double-click to</i>	clk_0			
	reset	Reset Input	<i>Double-click to</i>	[clock]			
	avalon_slave_0	Avalon Memory Mapped Slave	<i>Double-click to</i>	[clock]	<input type="checkbox"/> 0x0000_0200	0x0000_027f	

```
module vga_ball(input logic          clk,  
                input logic          reset,  
                input logic [31:0]   writedata,  
                output logic [31:0]   readdata,  
                input logic          write,  
                input logic          read,  
                input                chipselect,  
                input logic [4:0]    address);
```


Multi-Device Project

Registers:

All the interface registers are 4-bytes wide.

address [0-15]: write-only, for the software to send the 64 byte input to the hardware.

address [0-7]: read-only, for the software to read the 32 byte hash.

address [15]: read-only, 1 when the hash is ready to be read from address [0-7].

address [31]: write-only, for the software to send a reset signal.

```
/* ioctls and their arguments */  
#define WRITE_INPUT_0 _IOW(SHA256_MAGIC, 1, sha256_arg *)  
#define WRITE_INPUT_1 _IOW(SHA256_MAGIC, 4, sha256_arg *)  
#define WRITE_INPUT_2 _IOW(SHA256_MAGIC, 7, sha256_arg *)  
  
#define READ_DONE_0 _IOW(SHA256_MAGIC, 2, sha256_arg *)  
#define READ_DONE_1 _IOW(SHA256_MAGIC, 5, sha256_arg *)  
#define READ_DONE_2 _IOW(SHA256_MAGIC, 8, sha256_arg *)  
  
#define READ_HASH_0 _IOW(SHA256_MAGIC, 3, sha256_arg *)  
#define READ_HASH_1 _IOW(SHA256_MAGIC, 6, sha256_arg *)  
#define READ_HASH_2 _IOW(SHA256_MAGIC, 9, sha256_arg *)  
  
#define RESET_0 _IOW(SHA256_MAGIC, 10, sha256_arg *)  
#define RESET_1 _IOW(SHA256_MAGIC, 11, sha256_arg *)  
#define RESET_2 _IOW(SHA256_MAGIC, 12, sha256_arg *)  
#endif
```

```
/*  
 * Information about our device  
 */  
struct sha256_dev {  
    struct resource res; /* Resource: our registers */  
    void __iomem *virtbase; /* Where registers can be accessed in memory */  
} dev[3];
```

```
static int __init sha256_probe(struct platform_device *pdev)
{
    int ret;
    char *driver_names[] = {
        DRIVER_NAME_0,
        DRIVER_NAME_1,
        DRIVER_NAME_2,
    };

    /* Register ourselves as a misc device: creates /dev/sha256 */
    ret = misc_register(&sha256_misc_device[init_device_count]);

    /* Get the address of our registers from the device tree */
    ret = of_address_to_resource(pdev->dev.of_node, 0, &dev[init_device_count].res);
    if (ret) {
        ret = -ENOENT;
        goto out_deregister;
    }

    /* Make sure we can use these registers */
    if (request_mem_region(dev[init_device_count].res.start, resource_size(&dev[init_device_count].res),
        driver_names[init_device_count]) == NULL) {
        ret = -EBUSY;
        goto out_deregister;
    }
}
```

```
    write_input(raw_fd, input, index);

    let mut done: c_uint = 0;
    loop {
        read_done(raw_fd, &mut done, index);
        if done == 1 {
            break;
        }
    }

    pointer += 64;
}

let mut hash: sha256_hash = sha256_hash::default();
read_hash(raw_fd, &mut hash, index);
```

Multi-Device Testbench

Steps of the test:

1. Get a random block header from mempool.space.
2. Calculate its hash using the reference rust-bitcoin library.
3. Calculate the same hash using the three hardware devices.
4. Calculate the same hash using our own software implementation of SHA256.
5. Check all three of the hardware output against the golden output.

```
// GET HEADER BYTES
let header_bytes = get_block_header(height)?;

// GET GOLDEN HASH
let mut cursor = Cursor::new(header_bytes);
let header = Header::consensus_decode(&mut cursor)?;
let start = Instant::now();
let gold_hash = header.block_hash().to_raw_hash();
let end = Instant::now();
let duration = end - start;
*gold_time = *gold_time + duration;
```

```
start = Instant::now();  
// The i parameter here sets which device to use  
hash = sha256_hw::get_hash(&sha256_hw::get_hash(&header_bytes, i), i);  
end = Instant::now();  
duration = end - start;  
*hw_time = *hw_time + duration;
```



```
start = Instant::now();  
hash = sha256_sw::get_hash(&sha256_sw::get_hash(&header_bytes));  
end = Instant::now();  
duration = end - start;  
*sw_time = *sw_time + duration;
```

```
//CHECK
```

```
if gold_hash.as_byte_array() == hash.as_slice() {  
    print!("PASS  ");  
    println!("{:?}", duration);  
} else {  
    println!("FAIL  ");  
    println!("{:?}", duration);  
}
```

Multi-Device Testbench Result

BLOCK	390397		
GOLDEN	0d3e24c943a9683d8c1353e21f62fa8f0293d166948c38000000000000000000		26.85μs
HW[0]	0d3e24c943a9683d8c1353e21f62fa8f0293d166948c38000000000000000000	PASS	114.73μs
HW[1]	0d3e24c943a9683d8c1353e21f62fa8f0293d166948c38000000000000000000	PASS	74.571μs
HW[2]	0d3e24c943a9683d8c1353e21f62fa8f0293d166948c38000000000000000000	PASS	73.02μs
SW	0d3e24c943a9683d8c1353e21f62fa8f0293d166948c38000000000000000000	PASS	16.48μs
Average gold time: 25.754μs			
Average hw time : 87.114μs			
Average sw time : 16.528μs			

Average time calculated based on 25 tests

THANK YOU VERY MUCH FOR LISTENNNINNNNGGG!!!