Phase 2 Final Report

2018-04-05

Jonathan Saunders & Drew Harshaw
10193165 & 10186593

1.0 Functional Simulation Runs

1.1 Assembly Code used to test datapath

```
.org 0x0

ldi R1, 200                ;load 200 = 0xC8                                    R1 should have 0xC8

ld R2, R1, 1       ; c8 +1 = c9 go to address c9 to load 57 = 0x39                                    R2 should have 0x39

addi R3, R2, 3     ; 57 + 3 = 60 = 0x3C                                R3 should have 0x3C

ori R4, R2, 6      ; 110 OR 11 1001 = 0011 1111 = 0x3F R4 should have 0x3F

andi R5, R2, 15 ; 1111 AND 11 1001 = 1001 = 0x9                        R5 should have 0x9

brzr R0, BRANCH_1 ; R0 is still 0 therefore branch by 4 R0 should be 0

.org 0xA

BRANCH_1: brnz R5, BRANCH_2 ; R5 is 0x9 therefore branch by 4

.org 0xF

BRANCH_2: brpl R4, BRANCH_3 ; R4 is 0x3F therefore branch by 4

.org 0x14

BRANCH_3: brmi R4, 0x4 ; R4 is 0x3F therefore do NOT branch by 4

jr R3      ; jump to 0x3C

.org 0x3C

jal R4     ; jump to 0x3F

.org 0x3F

mul R1, R2 ; 57 * 200 = 11 400 = 0x2C88

mfhi R6 ; high is 0

mflo R7 ; low is 2C88

out R7   ; send datapath_to_io should be 0x2C88

in R8      ; get data from io port (0xffff)

st 502, R1          ; store into 502

st 303(R1), R0 ; store into 503
```

st 304(R1), R2 ; store into 504

st 305(R1), R3 ; store into 505

st 306(R1), R4 ; store into 506

st 307(R1), R5 ; store into 507

st 308(R1), R6 ; store into 508

st 309(R1), R7 ; store into 509

st 310(R1), R8 ; store into 510

ldi R1 with 3c

ld R1 with values from address 1f7 multiplied by value from r0 (which is 0)

addi r2, nothing with value of 1f8 (c sign extended from mdr)

.org 0xC8

ThisPos: 0xc8 .word ; 200 in decimal

NextPos: 0x39 .word ; 57 in decimal

EmptyPos: 0x0 .word

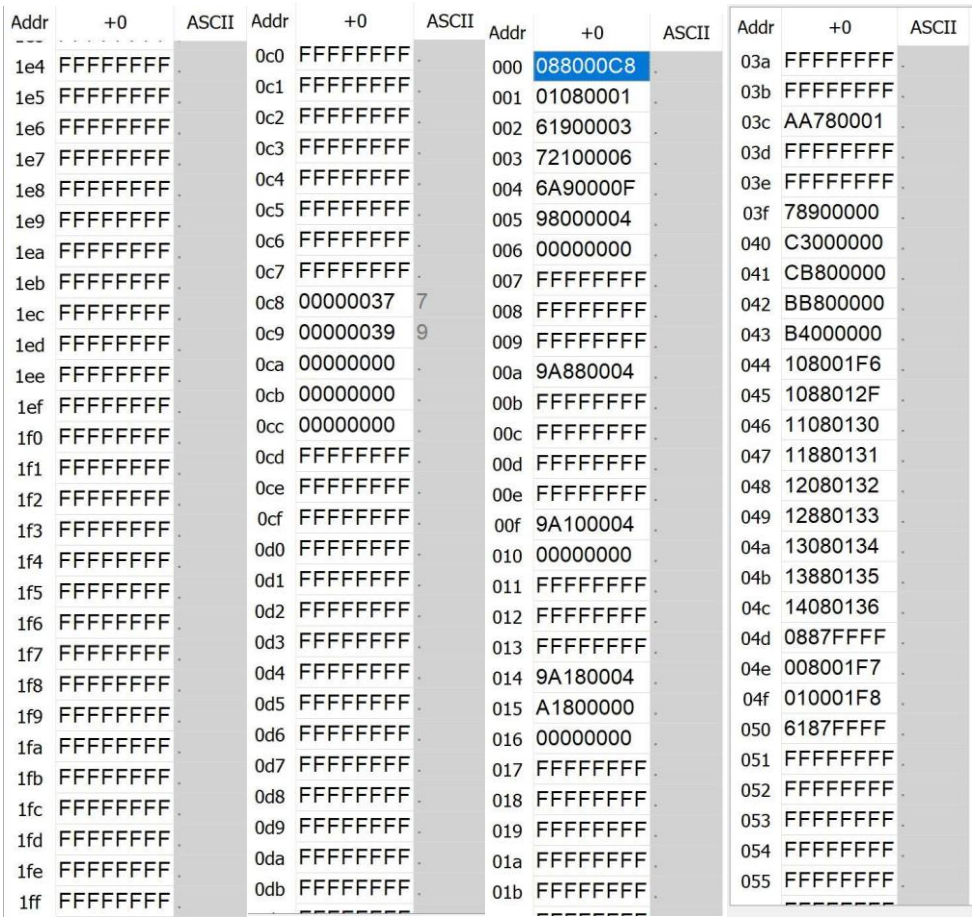
negative: 0xffff ffff .word ; -1 in decimal

negative2: 0xffff fffe .word ; -2 in decimal

1.2 WaveForms

On next page

2.0 Contents of memory



Addr	+0	ASCII
1e4	FFFFFFFF	
1e5	FFFFFFFF	
1e6	FFFFFFFF	
1e7	FFFFFFFF	
1e8	FFFFFFFF	
1e9	FFFFFFFF	
1ea	FFFFFFFF	
1eb	FFFFFFFF	
1ec	FFFFFFFF	
1ed	FFFFFFFF	
1ee	FFFFFFFF	
1ef	FFFFFFFF	
1f0	FFFFFFFF	
1f1	FFFFFFFF	
1f2	FFFFFFFF	
1f3	FFFFFFFF	
1f4	FFFFFFFF	
1f5	FFFFFFFF	
1f6	FFFFFFFF	
1f7	FFFFFFFF	
1f8	FFFFFFFF	
1f9	FFFFFFFF	
1fa	FFFFFFFF	
1fb	FFFFFFFF	
1fc	FFFFFFFF	
1fd	FFFFFFFF	
1fe	FFFFFFFF	
1ff	FFFFFFFF	

Addr	+0	ASCII
0c0	FFFFFFFF	
0c1	FFFFFFFF	
0c2	FFFFFFFF	
0c3	FFFFFFFF	
0c4	FFFFFFFF	
0c5	FFFFFFFF	
0c6	FFFFFFFF	
0c7	FFFFFFFF	
0c8	00000037	7
0c9	00000039	9
0ca	00000000	
0cb	00000000	
0cc	00000000	
0cd	FFFFFFFF	
0ce	FFFFFFFF	
0cf	FFFFFFFF	
0d0	FFFFFFFF	
0d1	FFFFFFFF	
0d2	FFFFFFFF	
0d3	FFFFFFFF	
0d4	FFFFFFFF	
0d5	FFFFFFFF	
0d6	FFFFFFFF	
0d7	FFFFFFFF	
0d8	FFFFFFFF	
0d9	FFFFFFFF	
0da	FFFFFFFF	
0db	FFFFFFFF	

Addr	+0	ASCII
000	088000C8	
001	01080001	
002	61900003	
003	72100006	
004	6A90000F	
005	98000004	
006	00000000	
007	FFFFFFFF	
008	FFFFFFFF	
009	FFFFFFFF	
00a	9A880004	
00b	FFFFFFFF	
00c	FFFFFFFF	
00d	FFFFFFFF	
00e	FFFFFFFF	
00f	9A100004	
010	00000000	
011	FFFFFFFF	
012	FFFFFFFF	
013	FFFFFFFF	
014	9A180004	
015	A1800000	
016	00000000	
017	FFFFFFFF	
018	FFFFFFFF	
019	FFFFFFFF	
01a	FFFFFFFF	
01b	FFFFFFFF	

Addr	+0	ASCII
03a	FFFFFFFF	
03b	FFFFFFFF	
03c	AA780001	
03d	FFFFFFFF	
03e	FFFFFFFF	
03f	78900000	
040	C3000000	
041	CB800000	
042	BB800000	
043	B4000000	
044	108001F6	
045	1088012F	
046	11080130	
047	11880131	
048	12080132	
049	12880133	
04a	13080134	
04b	13880135	
04c	14080136	
04d	0887FFFF	
04e	008001F7	
04f	010001F8	
050	6187FFFF	
051	FFFFFFFF	
052	FFFFFFFF	
053	FFFFFFFF	
054	FFFFFFFF	
055	FFFFFFFF	

3.0 VHDL Code including datapath testbench

On next page