Part a

Explanation

```
if(dec_rs1 == exe_rd && dec_rs1_renb == 1'b1 && exe_rd_wenb == 1'b1)

  dec_stall = 1'b1;

  dec_load_use = exe_load;

  dec_csr_use = exe_csr;
```

In my design I check each DEC operand's address and see if they are the same as any of the EXE, MEM or WRB operands' addresses. If they match, and the write enable bit of that stage is 1, then the system will stall.

Evaluation

Cycle count = 1396
CPI = 1.9
Instruction Count = 620

Critical Path:

The Critical path is caused when branch prediction is carried out

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.210 ns	Worst Hold Slack (WHS):	0.041 ns	Worst Pulse Width Slack (WPWS):	2.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	9148	Total Number of Endpoints:	9148	Total Number of Endpoints:	3133

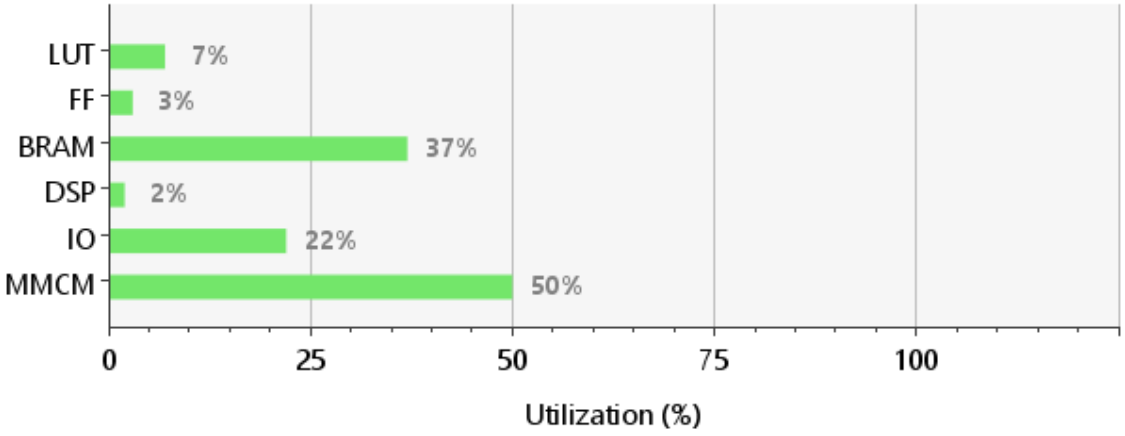
All user specified timing constraints are met.

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	..	Clock Uncertainty
Path 1	2.210	10	168	u_cpu/u_fet.../CLKARDCLK	u_cpu/u_fetc...ut_reg[30]/D	14.179	4.365	9.814	16.7	cpu_clk	cpu_clk		0.125

Time for Mandlebrot = 14.2 seconds

Name	Slice LUTs (53200)	Block RAM Tile (140)	DSPs (220)	Bonded IOB (125)	OLOGIC (125)	BUFGCTRL (32)	MMCME2_ADV (4)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)
top	3868	51.5	4	27	8	6	2	3045	561	93	1443	3868
u_cpu (cpu)	3533	3.5	4	0	0	0	0		554	93	1322	3533
u_control_unit (control_unit)	175	0	0	0	0	0	0		12	0	89	175
u_exec_unit (exec_unit)	1917	2	4	0	0	0	0		137	64	831	1917
u_alu (alu)	419	0	0	0	0	0	0		0	0	117	419
u_bypass_or_stall (bypass_or_stall)	16	0	0	0	0	0	0		0	0	6	16
u_dccm_ram (dccm_ram)	0	2	0	0	0	0	0		0	0	0	0
u_decoder (decoder)	55	0	0	0	0	0	0		0	0	21	55
u_divider (divider)	241	0	0	0	0	0	0		0	0	71	241
u_regfile (regfile)	681	0	0	0	0	0	0		128	64	447	681
u_store_queue (store_queue)	88	0	0	0	0	0	0		0	0	34	88
u_fetch_unit (fetch_unit)	1441	1.5	0	0	0	0	0		405	29	495	1441
u_cpu_clock_gen (cpu_clock_gen)	0	0	0	0	0	3	1		0	0	0	0
u_dvi_display (dvi_display)	304	48	0	0	8	0	0		0	0	122	304
u_resync (resync)	1	0	0	0	0	0	0		0	0	6	1
u_ssd_driver (ssd_driver)	28	0	0	0	0	0	0		7	0	13	28
u_video_clock_gen (video_clock_gen)	0	0	0	0	0	3	1		0	0	0	0

Resource	Utilization	Available	Utilization %
LUT	3868	53200	7.27
FF	3045	106400	2.86
BRAM	51.50	140	36.79
DSP	4	220	1.82
IO	27	125	21.60
MMCM	2	4	50.00



Part b

Explanation

```
depend1 = 1'b0;
depend2 = 1'b0;
```

I added these two registers to my design to allow me to tell which DEC operand has a dependency further down the pipeline.

```
  if(dec_rs1 == exe_rd && dec_rs1_renb == 1'b1 && exe_rd_wenb == 1'b1 && depend1 == 1'b0)
  begin
  // Set the load and csr registers
    dec_load_use = exe_load;
    dec_csr_use = exe_csr;
    depend1 = 1'b1;
    // If the operation is load or csr then stall
    if(exe_load == 1'b1 || exe_csr == 1'b1)
        begin
        dec_stall = 1'b1;
        end
    else
        dec_rs1_data = exe_result;
  end
```

At the EXE stage I only stall if there is a load or csr operation. Otherwise the exe_result value is passed back to the relevant DEC operand.

```
  if(dec_rs2 == mem_rd && dec_rs2_renb == 1'b1 && mem_rd_wenb == 1'b1 && depend2 == 1'b0)
  begin
    dec_rs2_data = mem_result;
    depend2 = 1'b1;
    if(dec_rs2 == exe_rd && dec_stall == 1'b1)
    // If the dec operand has a dependency in the EXE stage but also later down the
pipeline then there is no need to stall
        dec_stall = 1'b0;
  end
```

At the MEM and WRB stages there will never be a stall. If the DEC operand had a dependency at the EXE stage but also has one at the MEM or WRB stage, then the stall is stopped.

Evaluation

Cycle Count = 874
CPI = 1.1
Instruction Count = 620

1. Why has the CPI not reduced all the way to 1.0 in this optimized run?
The CPI is not 1.0 as there are still stalls for CSR and load operations.

2. Tabulate any remaining stalls that occur in this test
I counted a total of 40 load_use stalls, no csr stalls and two pipeline flushes.

3. If there are pipeline flushes, explain why they are there and estimate their performance impact for this test
Pipeline flushes occur when there is a branch, causing the next instruction to be incorrect. They would have a relatively big impact, as instructions were being carried out that need to be wiped and, after the flush, the EXE, MEM and WRB stage will not be doing anything.

Critical Path:
The critical path in this optimised design is caused by a multiply operation.

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.922 ns	Worst Hold Slack (WHS):	0.037 ns	Worst Pulse Width Slack (WPWS):	2.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	9148	Total Number of Endpoints:	9148	Total Number of Endpoints:	3133

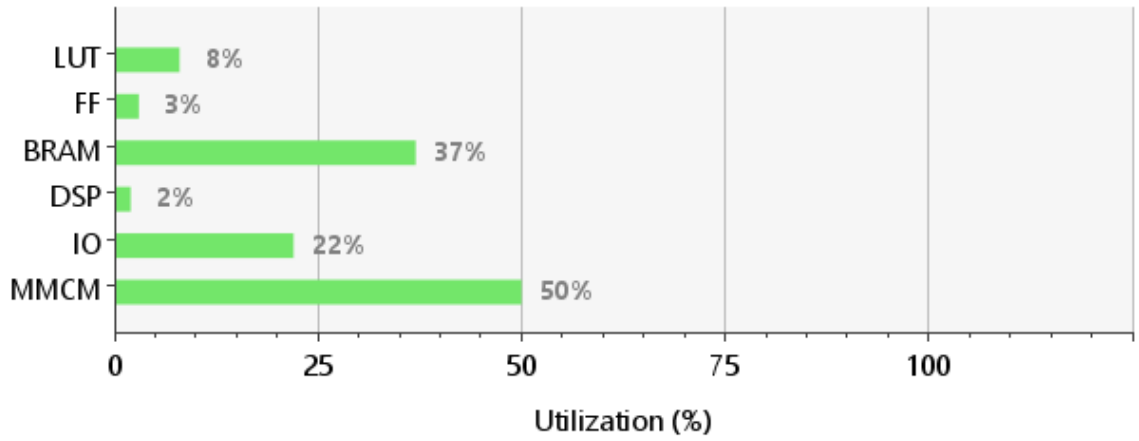
All user specified timing constraints are met.

Name	Slack ^1	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Clock Uncertainty
Path 1	0.922	12	58	u_cpu/u_exec..._r_reg[31]/C	u_cpu/u_exec..sult0/A[16]	15.280	8.063	7.217	16.7	cpu_clk	cpu_clk	0.125

Time for Mandlebrot = 8.5 seconds

Name	Slice LUTs (53200)	Block RAM Tile (140)	DSPs (220)	Bonded IOB (125)	OLOGIC (125)	BUFGCTRL (32)	MMCME2_ADV (4)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)
top	4011	51.5	4	27	8	6	2	3045	561	93	1306	4011
u_cpu (cpu)	3676	3.5	4	0	0	0	0		554	93	1199	3676
u_control_unit (control_unit)	174	0	0	0	0	0	0		12	0	93	174
u_exec_unit (exec_unit)	2059	2	4	0	0	0	0		137	64	784	2059
u_alu (alu)	419	0	0	0	0	0	0		0	0	117	419
u_bypass_or_stall (bypass_or_stall)	150	0	0	0	0	0	0		0	0	55	150
u_dccm_ram (dccm_ram)	0	2	0	0	0	0	0		0	0	0	0
u_decoder (decoder)	55	0	0	0	0	0	0		0	0	33	55
u_divider (divider)	244	0	0	0	0	0	0		0	0	89	244
u_regfile (regfile)	681	0	0	0	0	0	0		128	64	370	681
u_store_queue (store_queue)	89	0	0	0	0	0	0		0	0	41	89
u_fetch_unit (fetch_unit)	1443	1.5	0	0	0	0	0		405	29	487	1443
u_cpu_clock_gen (cpu_clock_gen)	0	0	0	0	0	3	1		0	0	0	0
u_dvi_display (dvi_display)	304	48	0	0	8	0	0		0	0	113	304
u_resync (resync)	1	0	0	0	0	0	0		0	0	5	1
u_ssd_driver (ssd_driver)	28	0	0	0	0	0	0		7	0	13	28
u_video_clock_gen (video_clock_gen)	0	0	0	0	0	3	1		0	0	0	0

Resource	Utilization	Available	Utilization %
LUT	4011	53200	7.54
FF	3045	106400	2.86
BRAM	51.50	140	36.79
DSP	4	220	1.82
IO	27	125	21.60
MMCM	2	4	50.00



Part c

Explanation

```
// If a multiply operation is carried out, send a signal to the stall_or_bypass to stall
case (exe_alu_opc_r)
  M32_OPC_MUL:
  begin
    m32_result  = m64_result[31:0];
    exe_mul_r = 1'b1;
  end
  M32_OPC_MULH,
  M32_OPC_MULHSU,
  M32_OPC_MULHU:
  begin
      m32_result  = m64_result[63:32];
      exe_mul_r = 1'b1;
  end
  M32_OPC_DIV,
  M32_OPC_DIVU,
  M32_OPC_REM,
  M32_OPC_REMU:   m32_result  = exe_div_result;
  default:        m32_result  = 32'hXXXX_XXXX;
endcase
```

To optimise my design, I decided to prevent multiply results being passed back to bypass, as it was causing significant slowdown of the pipeline. When a multiply operation was carried out, I send a stall signal back to bypass_or_stall. This increased my CPI as it introduced additional stalls. Due to this optimisation, multiply is no longer the operation that causes the critical path and it is now branch prediction again.

```
if(dec_rs1 == exe_rd && dec_rs1_renb == 1'b1 && exe_rd_wenb == 1'b1 && depend1 == 1'b0)
begin
// Set the load, csr and mul registers
  dec_load_use = exe_load;
  dec_csr_use = exe_csr;
  dec_mul_use = exe_mul;
  depend1 = 1'b1;
  // If the operation is load, csr or multiply then stall
  if(exe_load == 1'b1 || exe_csr == 1'b1 || exe_mul == 1'b1)
      begin
```

```
        dec_stall = 1'b1;
        end
    else
        dec_rs1_data = exe_result;
end
```

I added a condition to stall in the bypass_or_stall module if a multiply operation is being carried out.

Evaluation

Cycle Count = 946
CPI = 1.2
Instruction Count = 620

Timing report after adjusting Fmax:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.161 ns	Worst Hold Slack (WHS):	0.132 ns	Worst Pulse Width Slack (WPWS):	2.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	9148	Total Number of Endpoints:	9148	Total Number of Endpoints:	3133

All user specified timing constraints are met.

Name	Slack ^1	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Clock Uncertainty
Path 1	0.161	13	58	u_cpu/u_exec..._r_reg[31]/C	u_cpu/u_exec..._r_reg[24]/D	11.964	7.874	4.090	12.2	cpu_clk	cpu_clk	0.090

Final Fmax = 82MHz
Time for Mandlebrot = 6.5 seconds

Name	Slice LUTs (53200)	Block RAM Tile (140)	DSPs (220)	Bonded IOB (125)	OLOGIC (125)	BUFGCTRL (32)	MMCME2_ADV (4)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)
top	4009	51.5	4	27	8	6	2	3045	561	93	1394	4009
u_cpu (cpu)	3675	3.5	4	0	0	0	0		554	93	1276	3675
u_control_unit (control_unit)	174	0	0	0	0	0	0		12	0	87	174
u_exec_unit (exec_unit)	2059	2	4	0	0	0	0		137	64	794	2059
u_alu (alu)	418	0	0	0	0	0	0		0	0	113	418
u_bypass_or_stall (bypass_or_stall)	151	0	0	0	0	0	0		0	0	53	151
u_dccm_ram (dccm_ram)	0	2	0	0	0	0	0		0	0	0	0
u_decoder (decoder)	55	0	0	0	0	0	0		0	0	24	55
u_divider (divider)	244	0	0	0	0	0	0		0	0	79	244
u_regfile (regfile)	680	0	0	0	0	0	0		128	64	383	680
u_store_queue (store_queue)	89	0	0	0	0	0	0		0	0	38	89
u_fetch_unit (fetch_unit)	1442	1.5	0	0	0	0	0		405	29	488	1442
u_cpu_clock_gen (cpu_clock_gen)	0	0	0	0	0	3	1		0	0	0	0
u_dvi_display (dvi_display)	304	48	0	0	8	0	0		0	0	122	304
u_resync (resync)	1	0	0	0	0	0	0		0	0	5	1
u_ssd_driver (ssd_driver)	28	0	0	0	0	0	0		7	0	13	28
u_video_clock_gen (video_clock_gen)	0	0	0	0	0	3	1		0	0	0	0

Resource	Utilization	Available	Utilization %
LUT	4009	53200	7.54
FF	3045	106400	2.86
BRAM	51.50	140	36.79
DSP	4	220	1.82
IO	27	125	21.60
MMCM	2	4	50.00