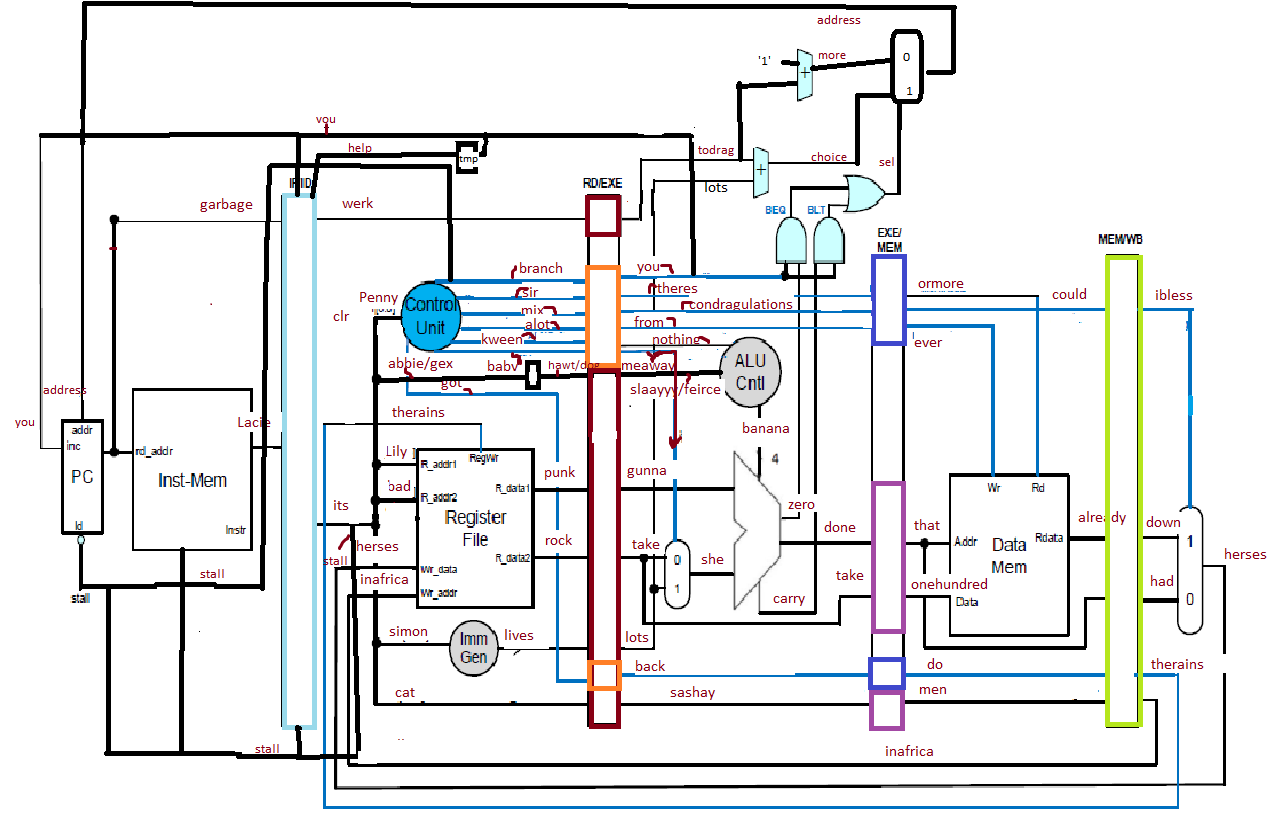
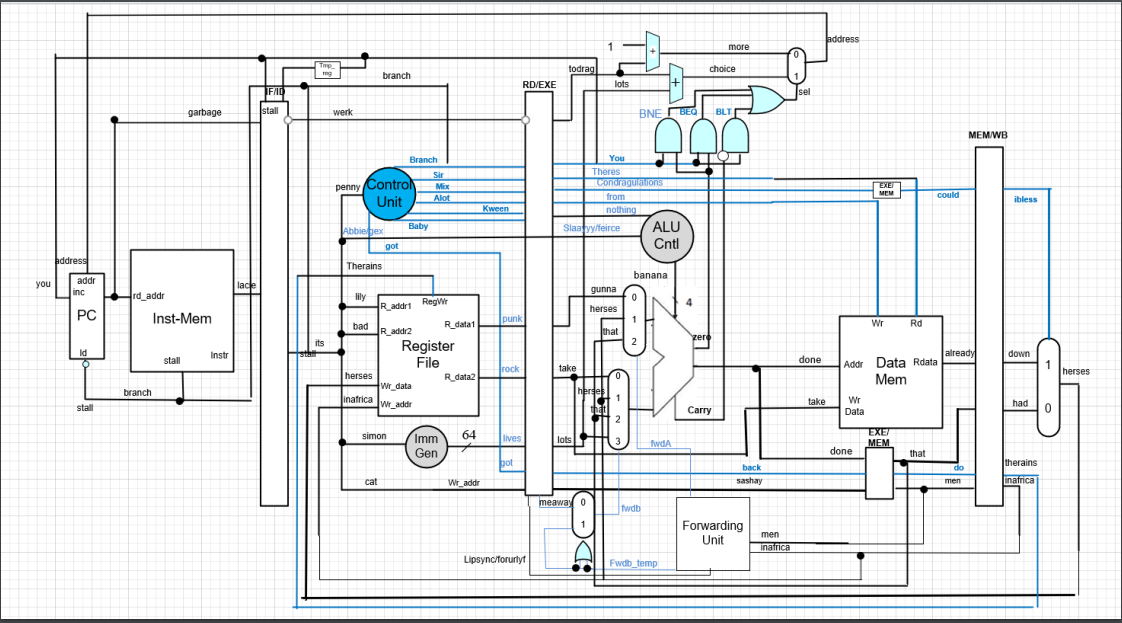
Appendix B.

*Datapath Diagrams*

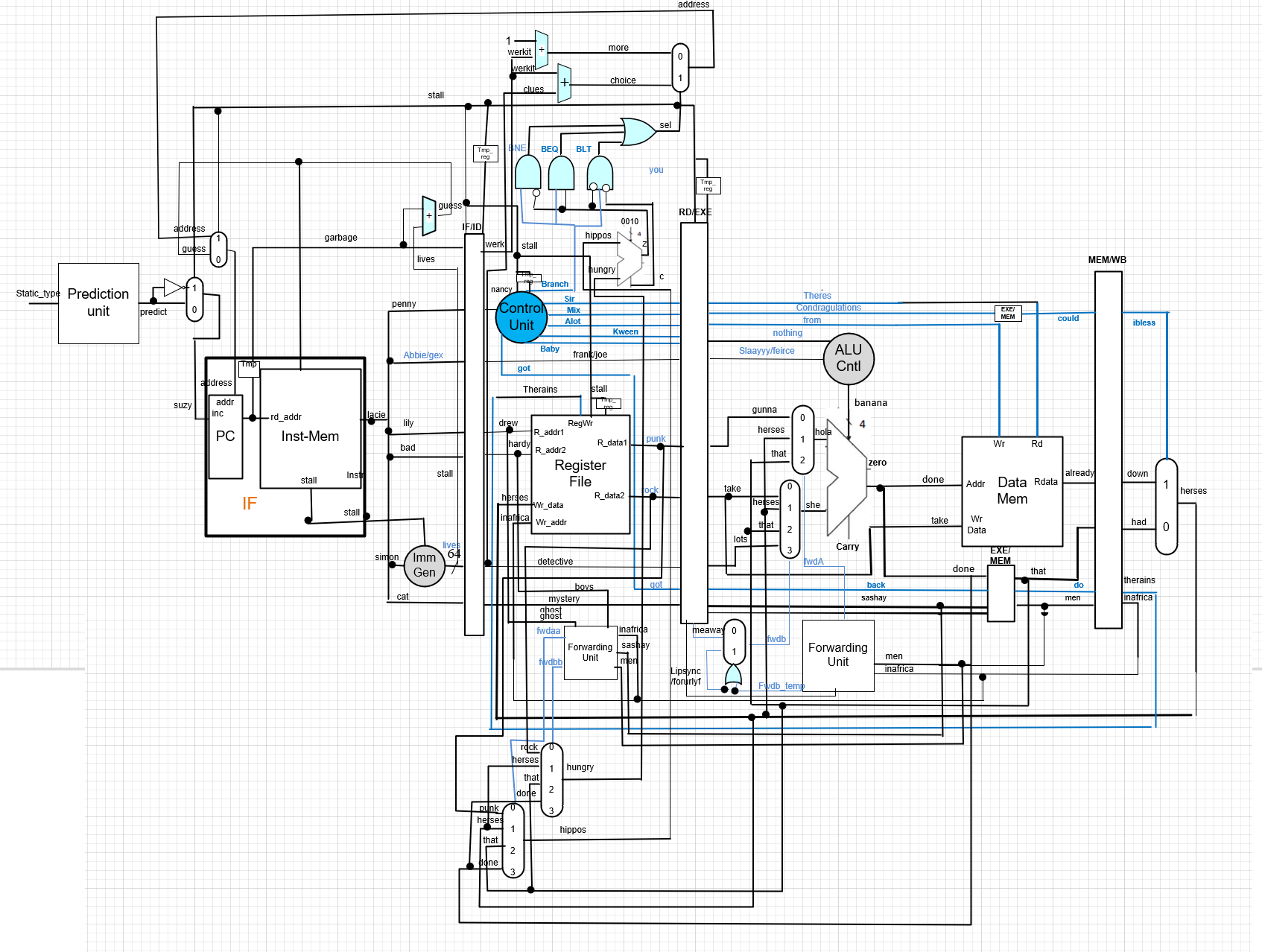
Fully piplined, no forwarding no prediction



Forwarding

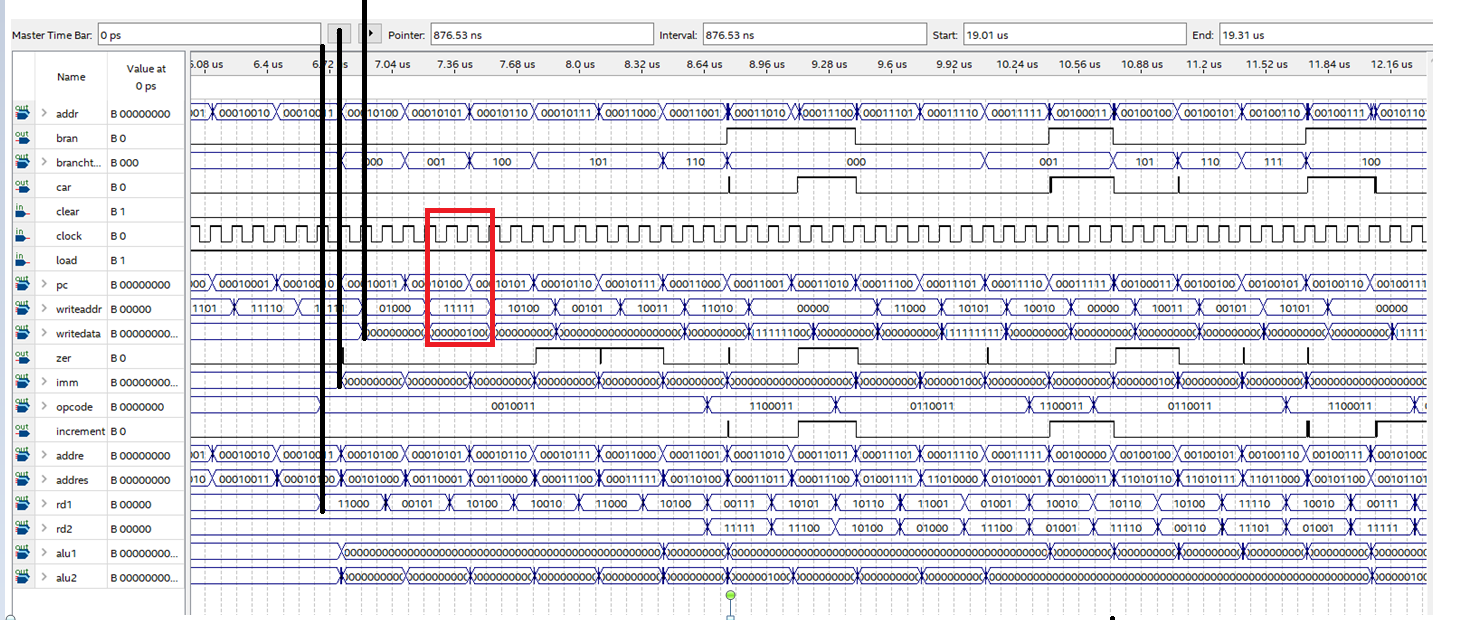


Branch prediction

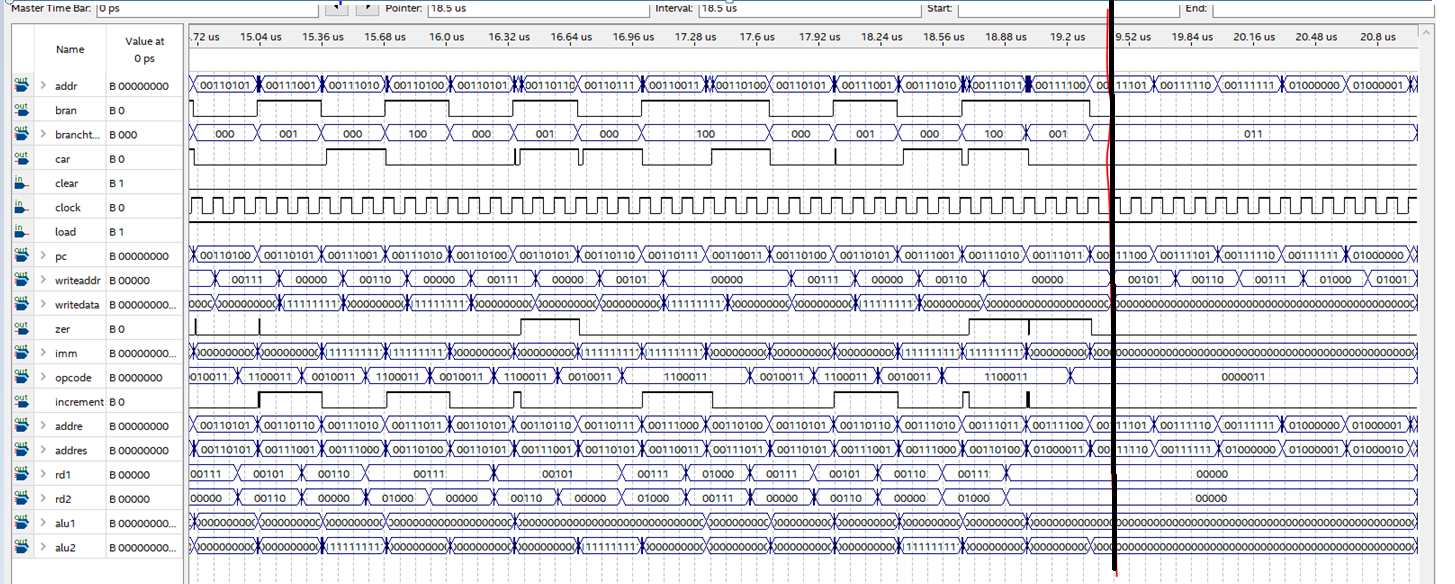


*Experiment results*

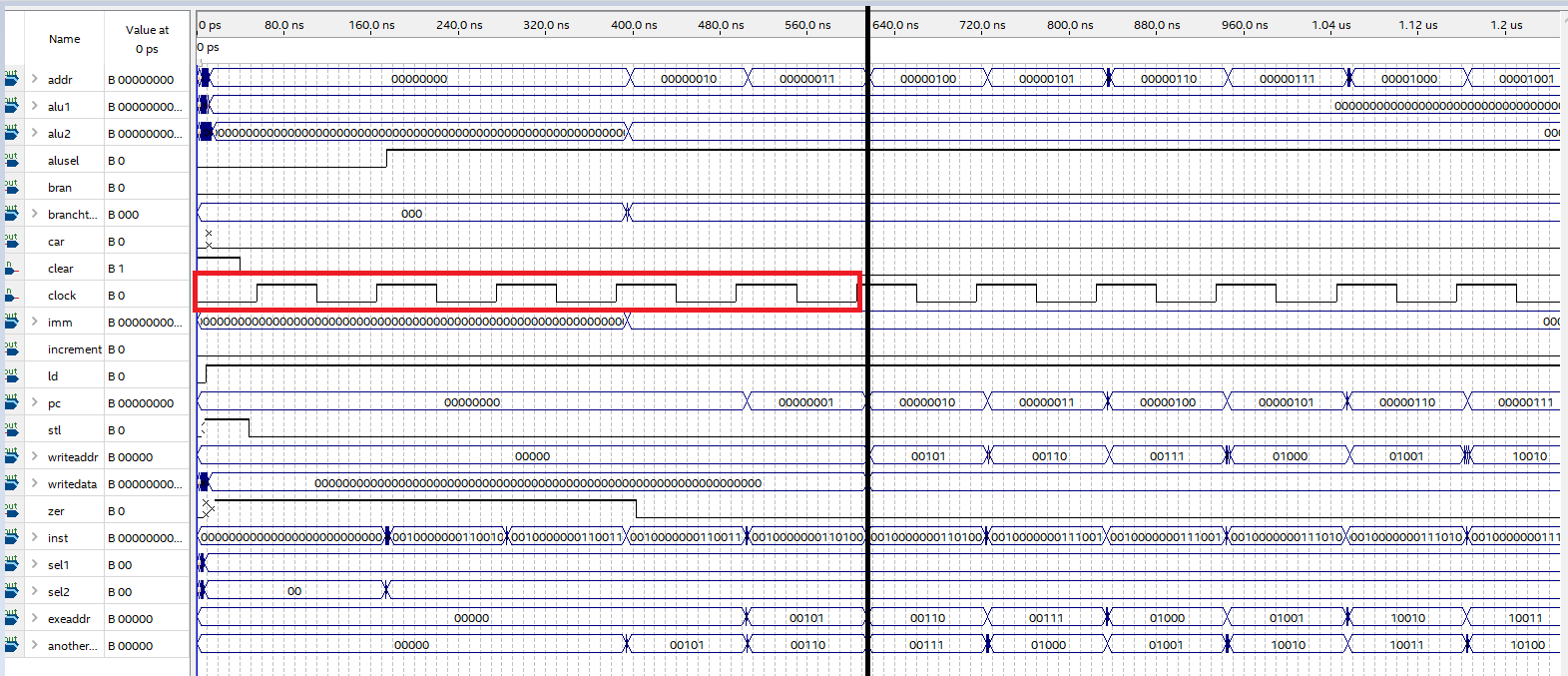
Lab 5 datapath, red box is one instruction clack lines show movement through datapath.



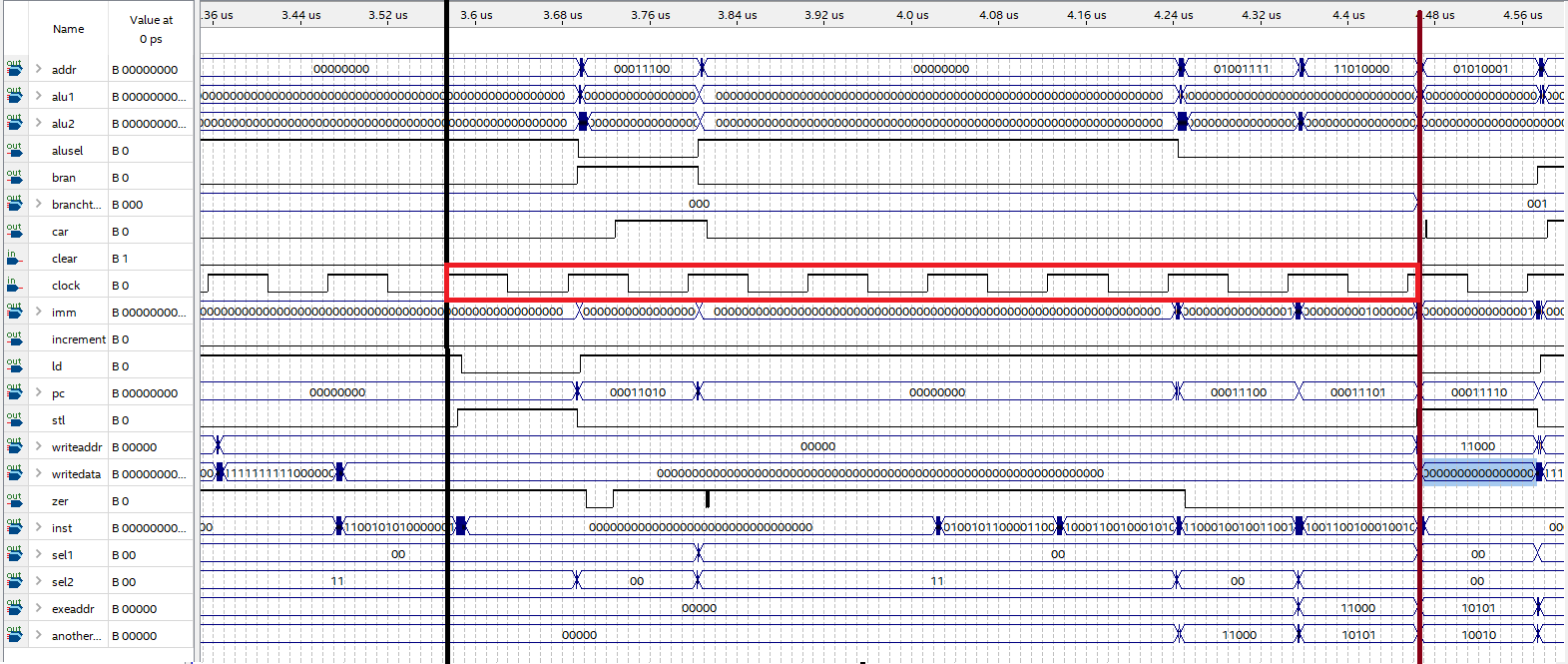
Lab 5 end time for instruction set



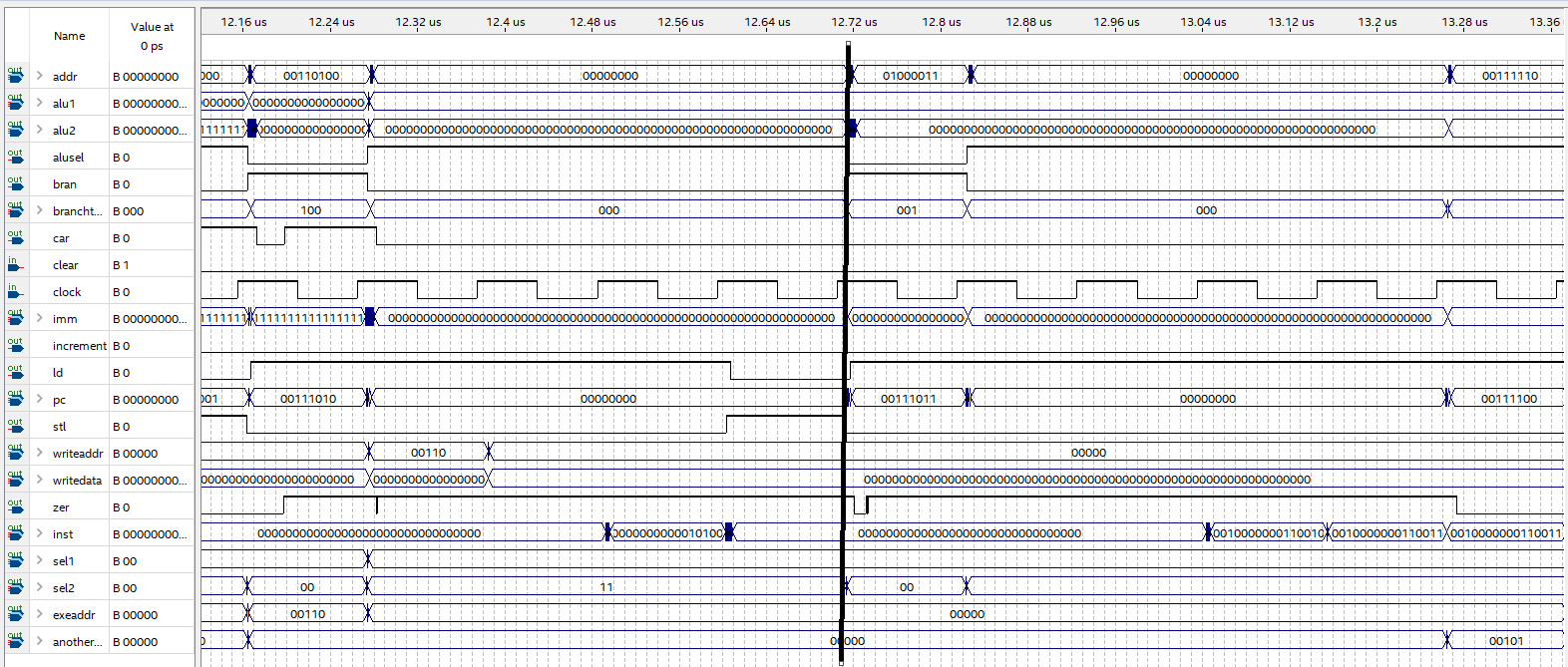
Datapath with data forwarding, red box best case time to complete one instruction



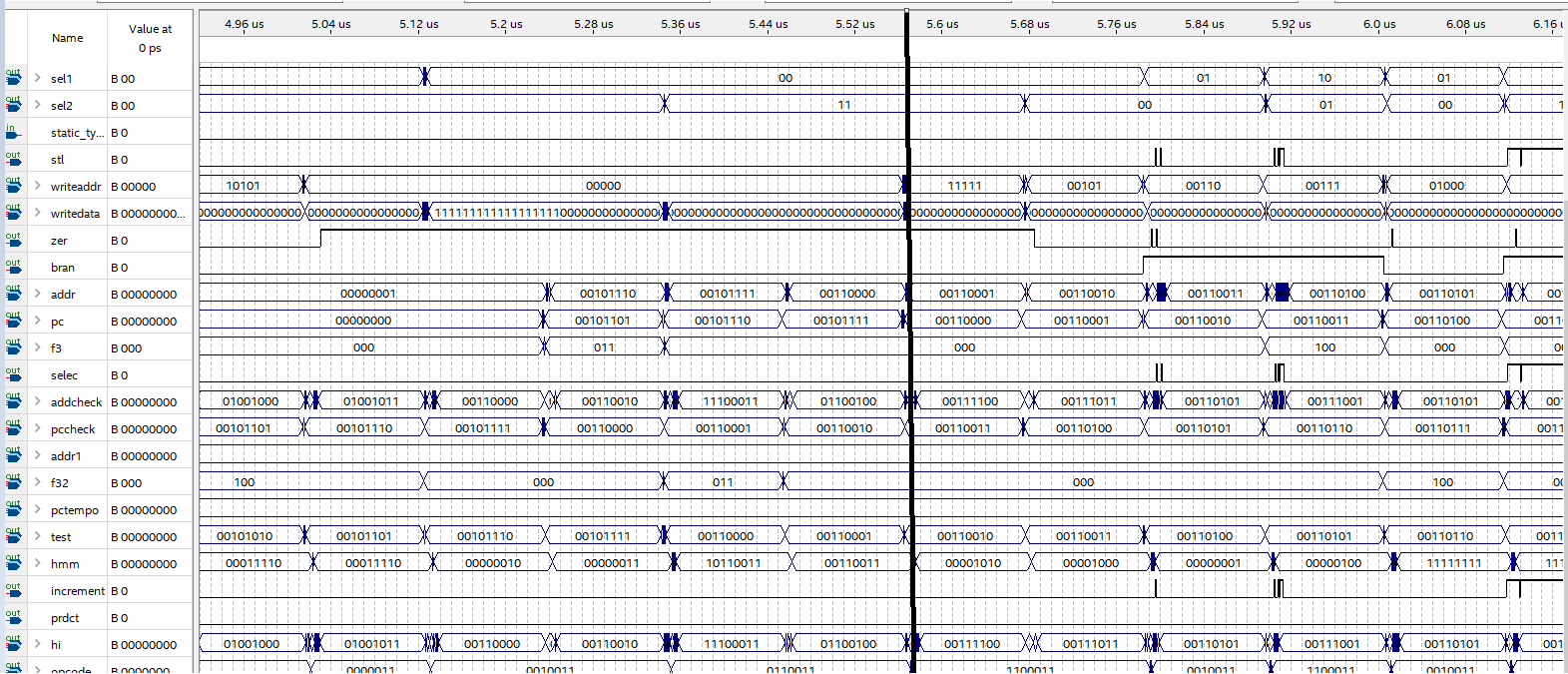
Datapath with data forwarding worst case time to finish one instruction



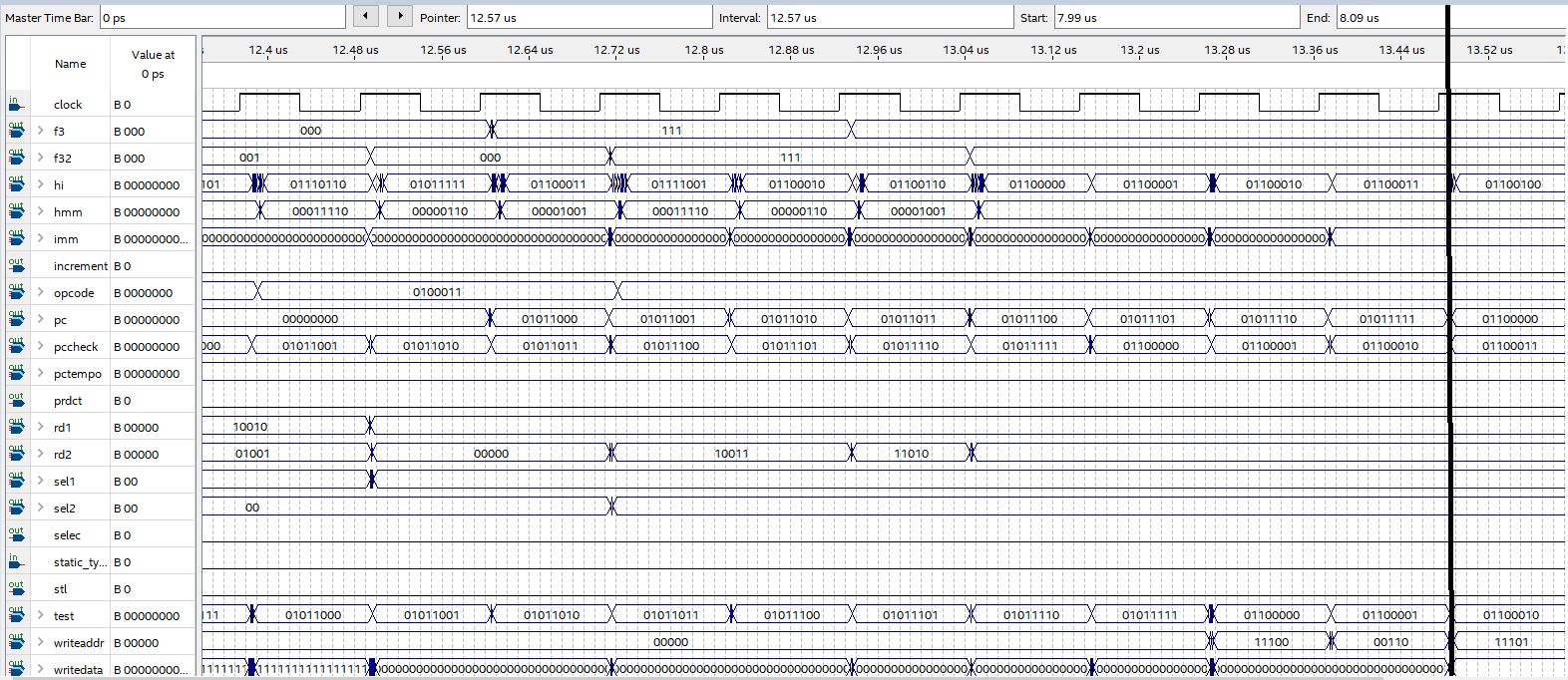
Datapath with data forwarding end time for instruction set



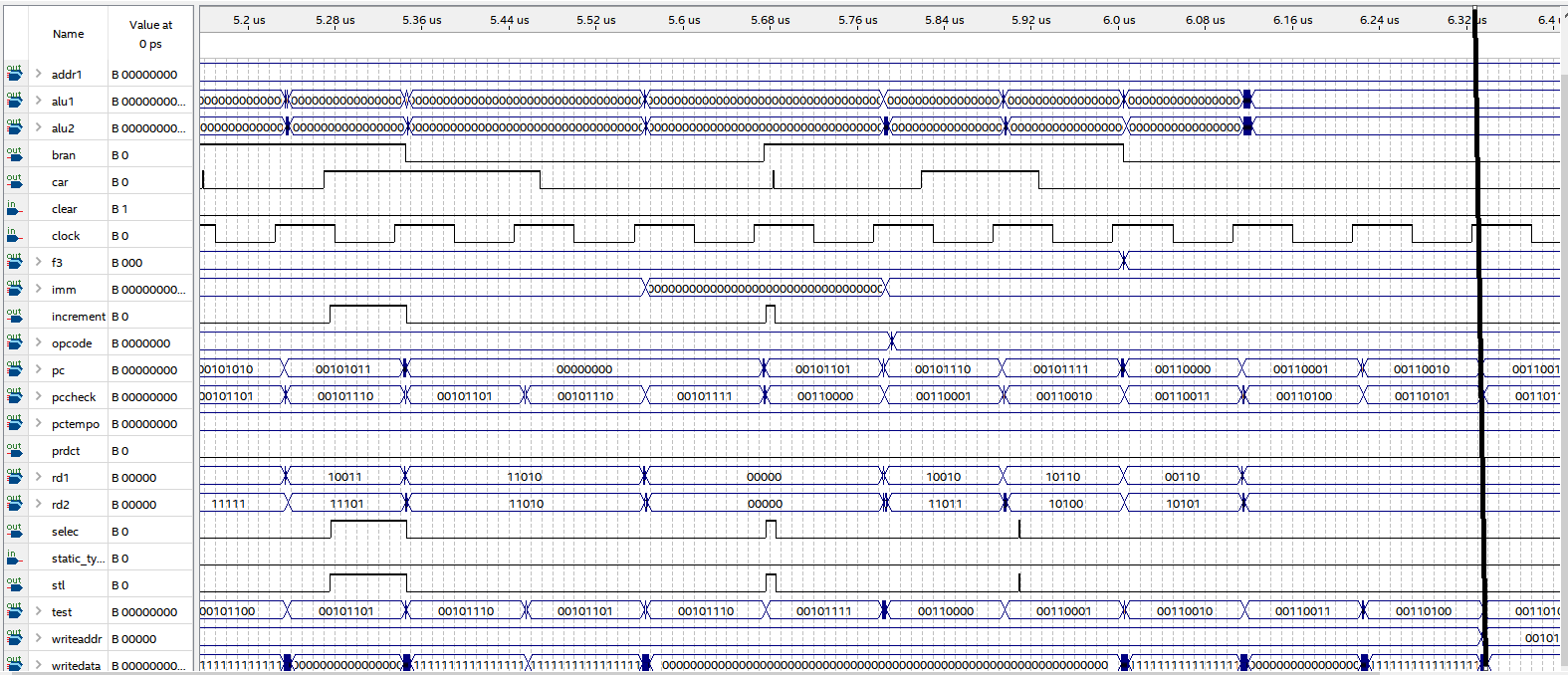
Never taken prediction end time: moderate

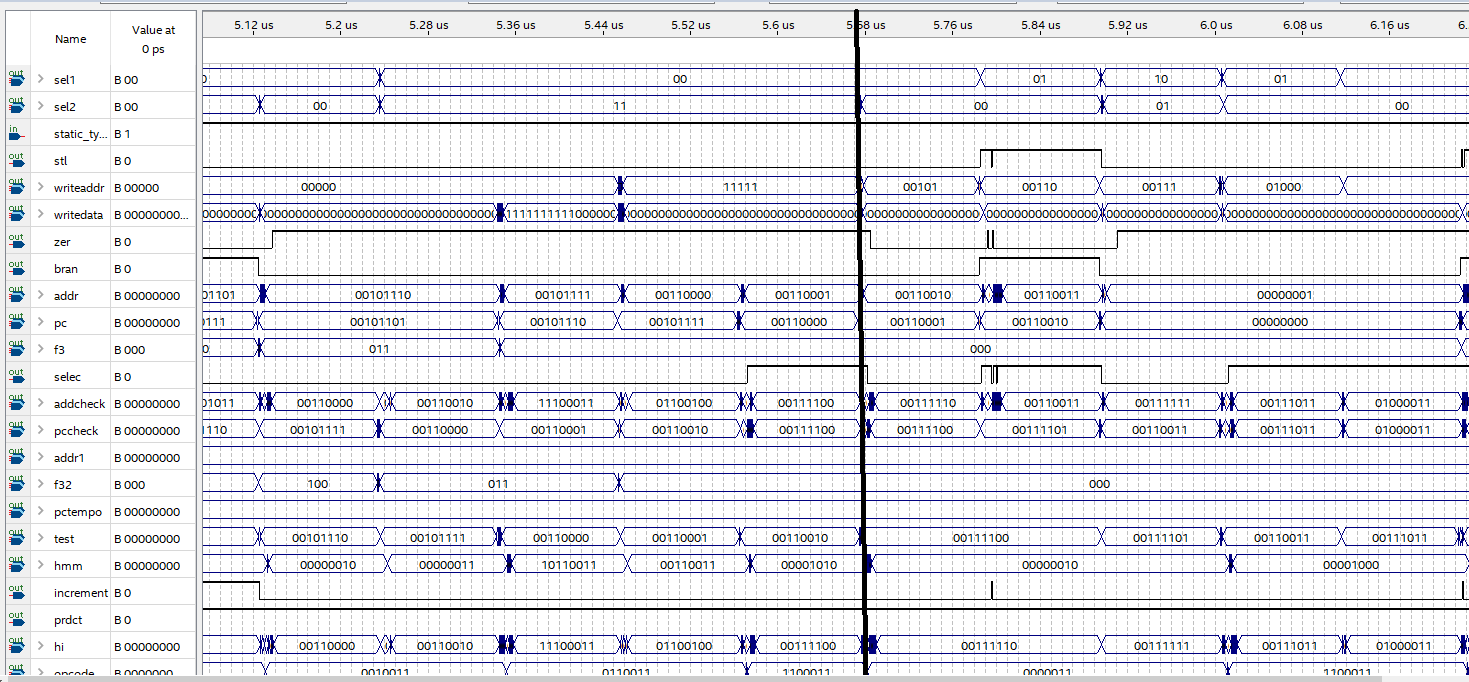


Never taken prediction end time: many branches

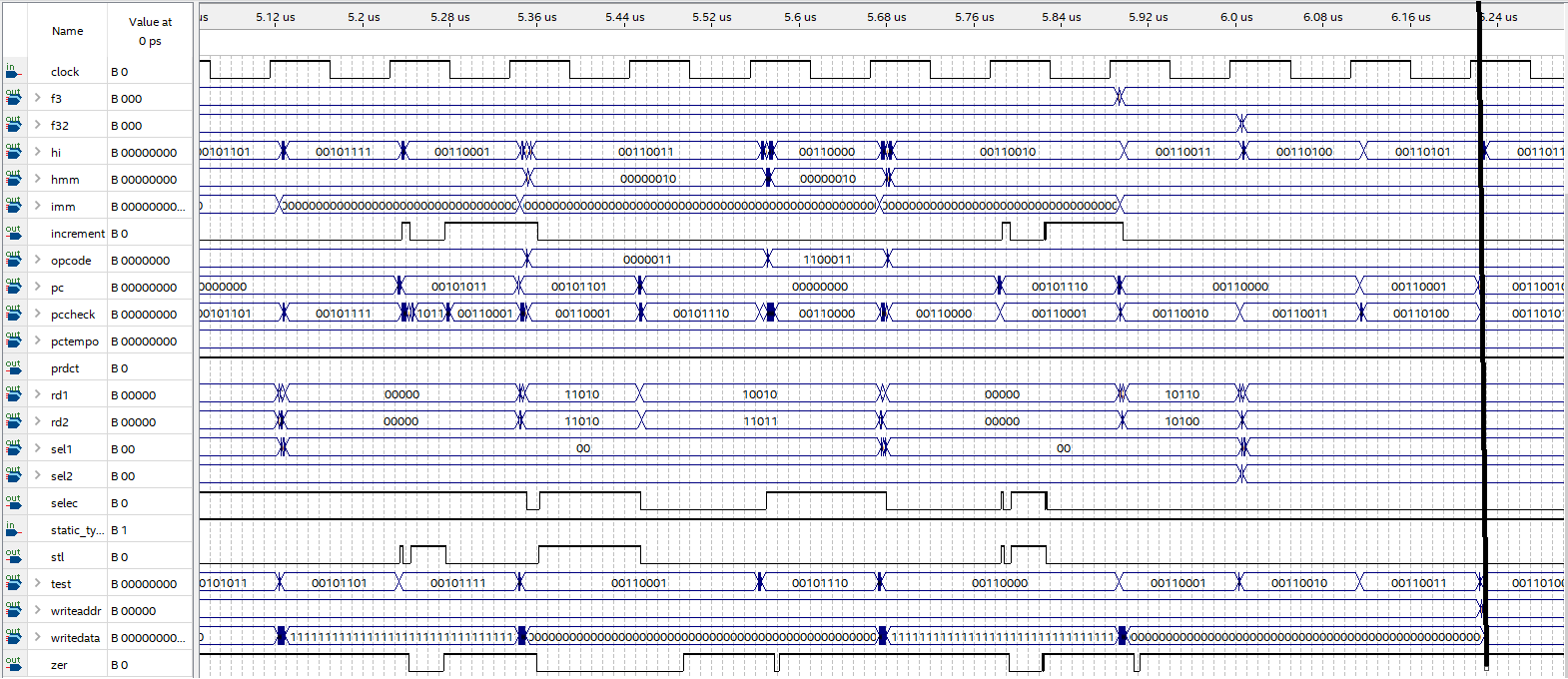


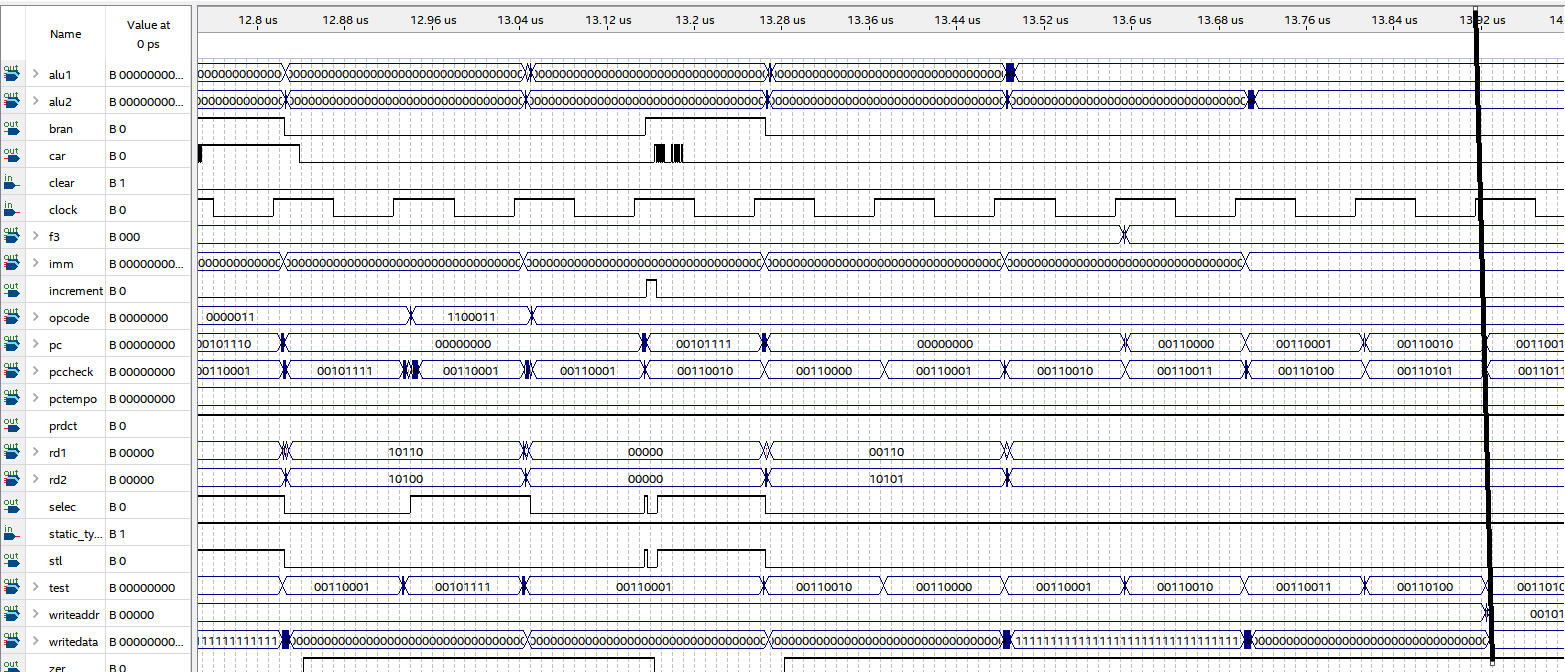
Never taken prediction end time: few branches



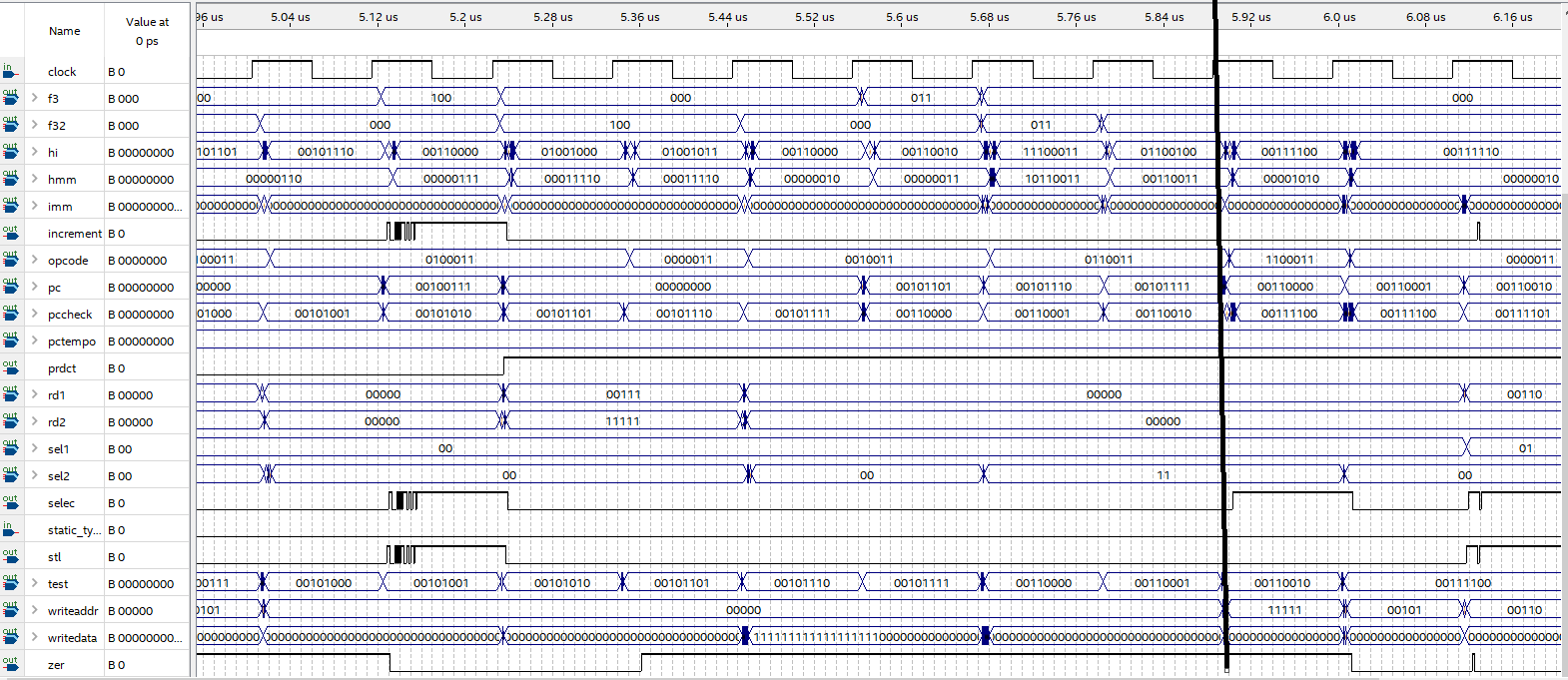
Always taken prediction end time: moderate

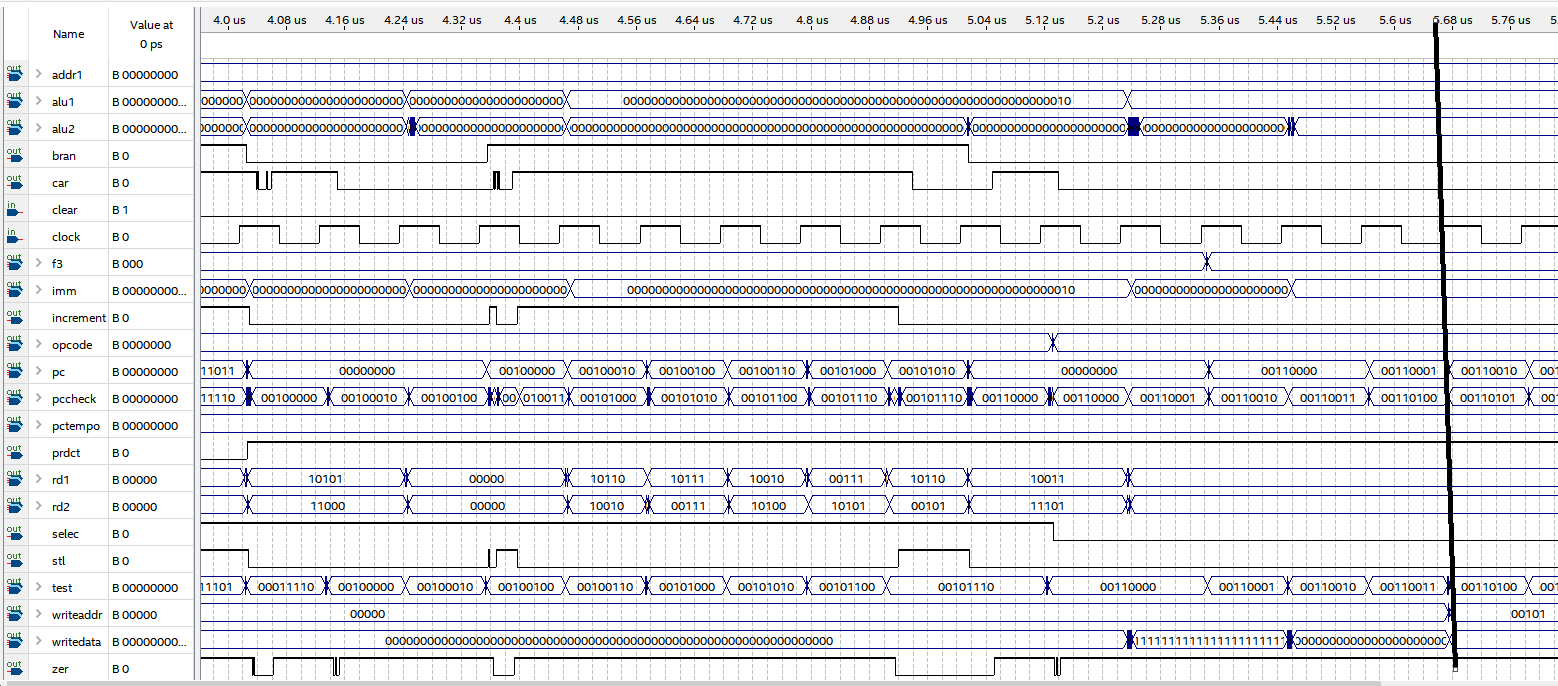
Always taken prediction end time: many branches

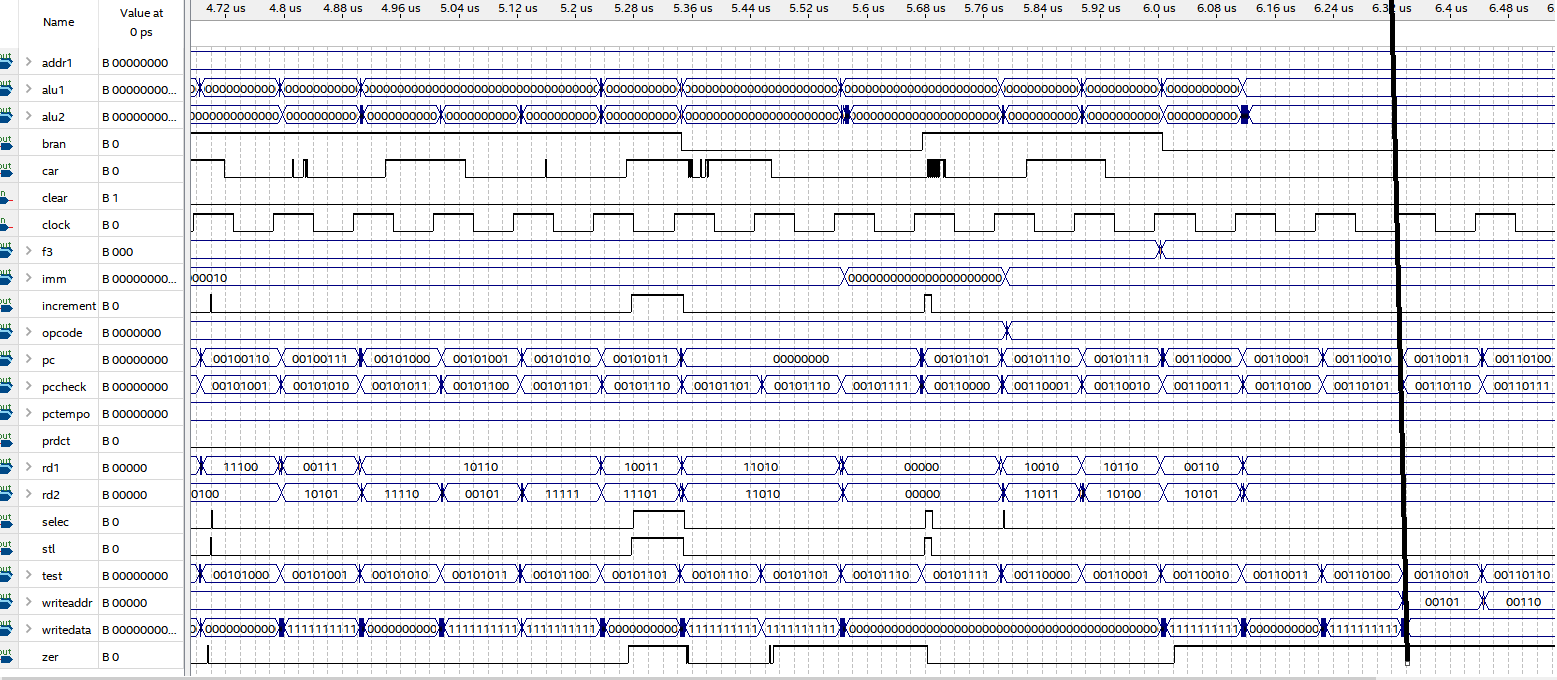


Always taken prediction end time: few branches

Dynamic two bit prediction end time: moderate



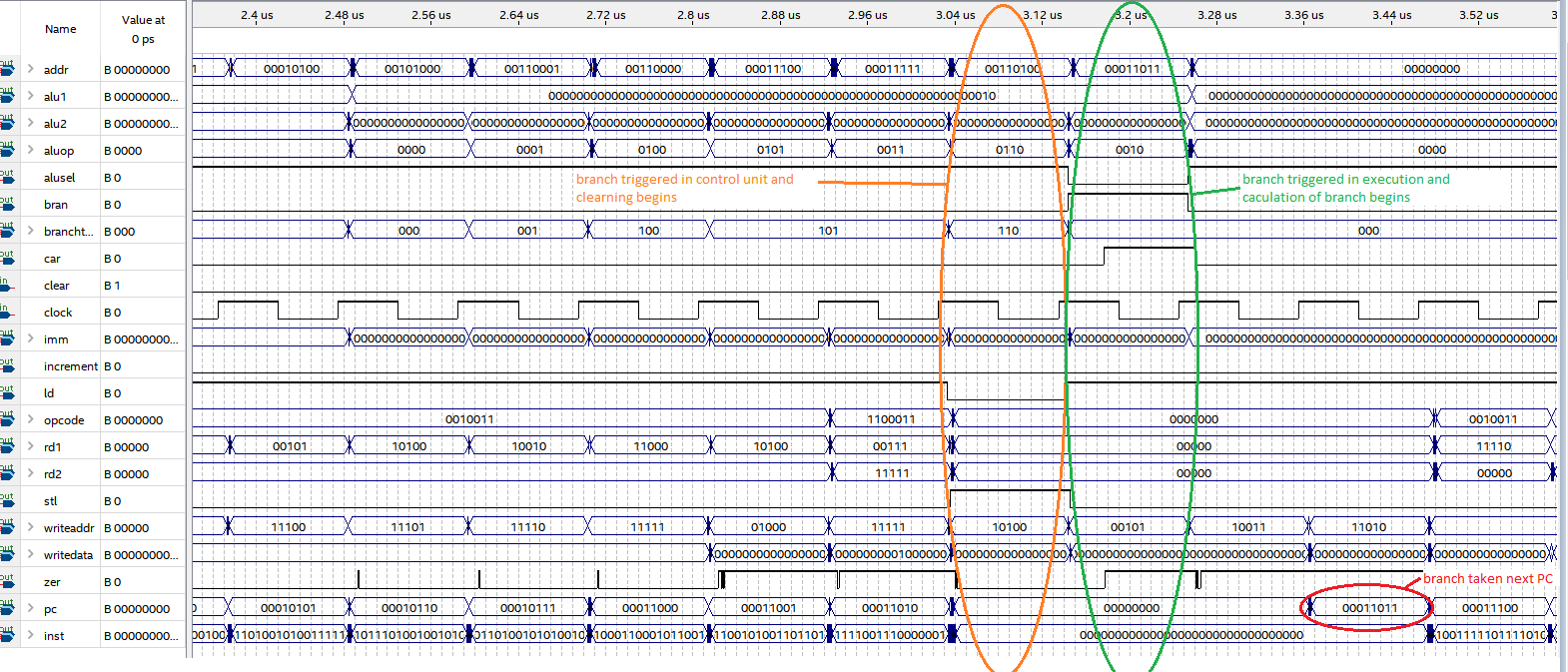
Dynamic two bit predictor end time : many branches

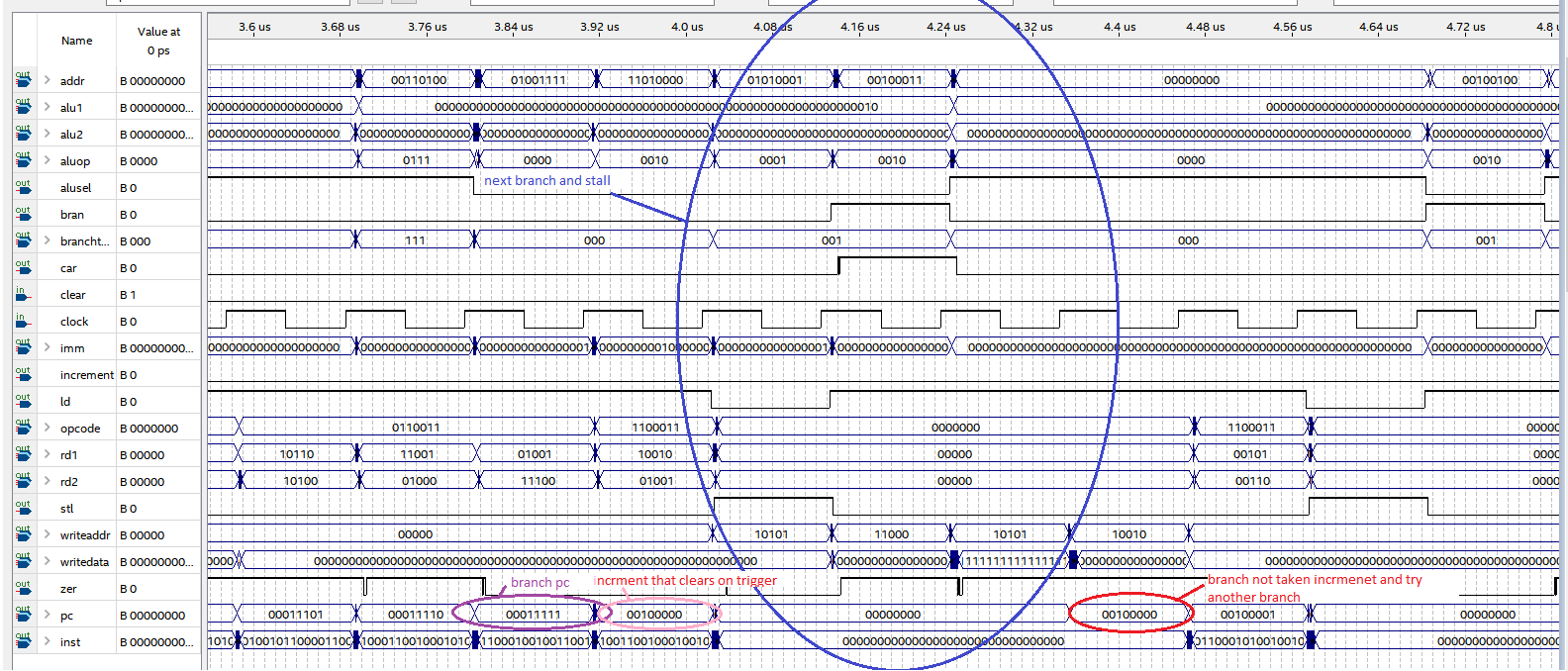
Dynamic two bit predictor end time: few branches

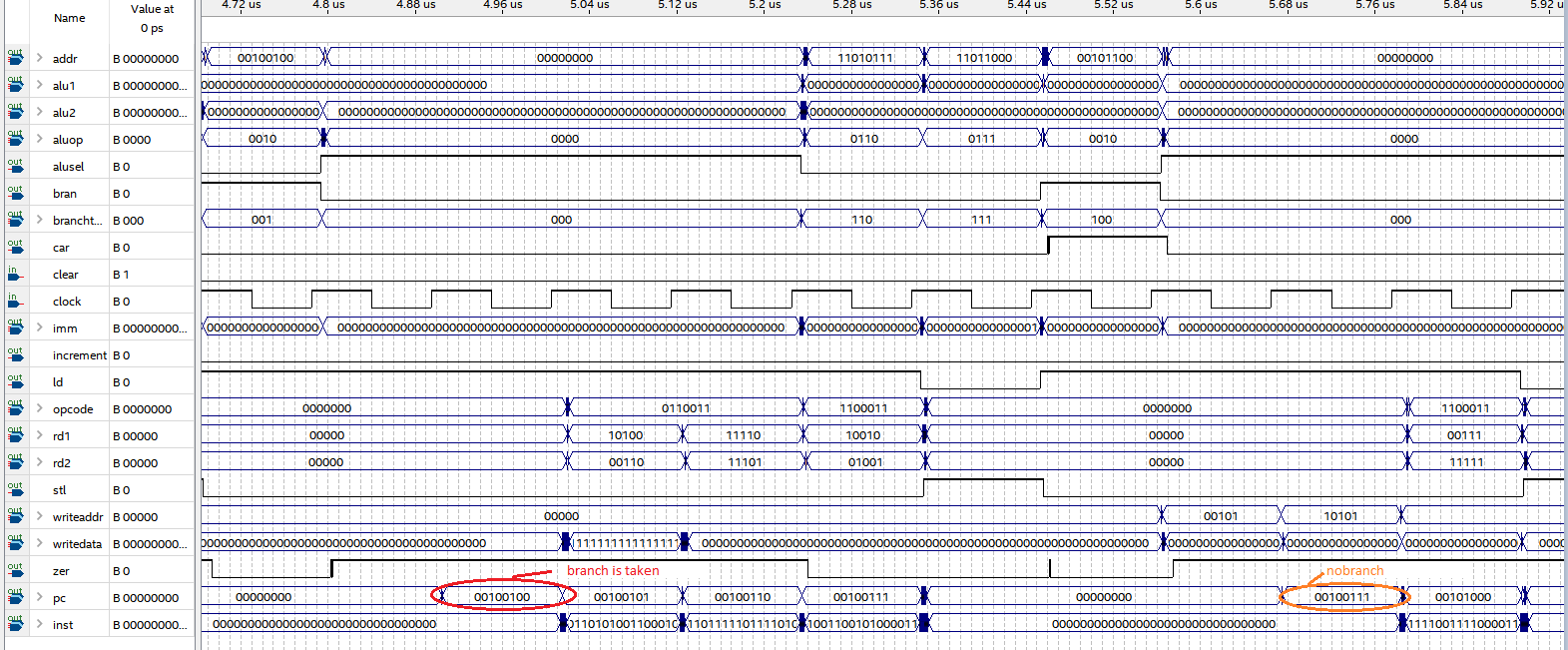
*Waveforms*

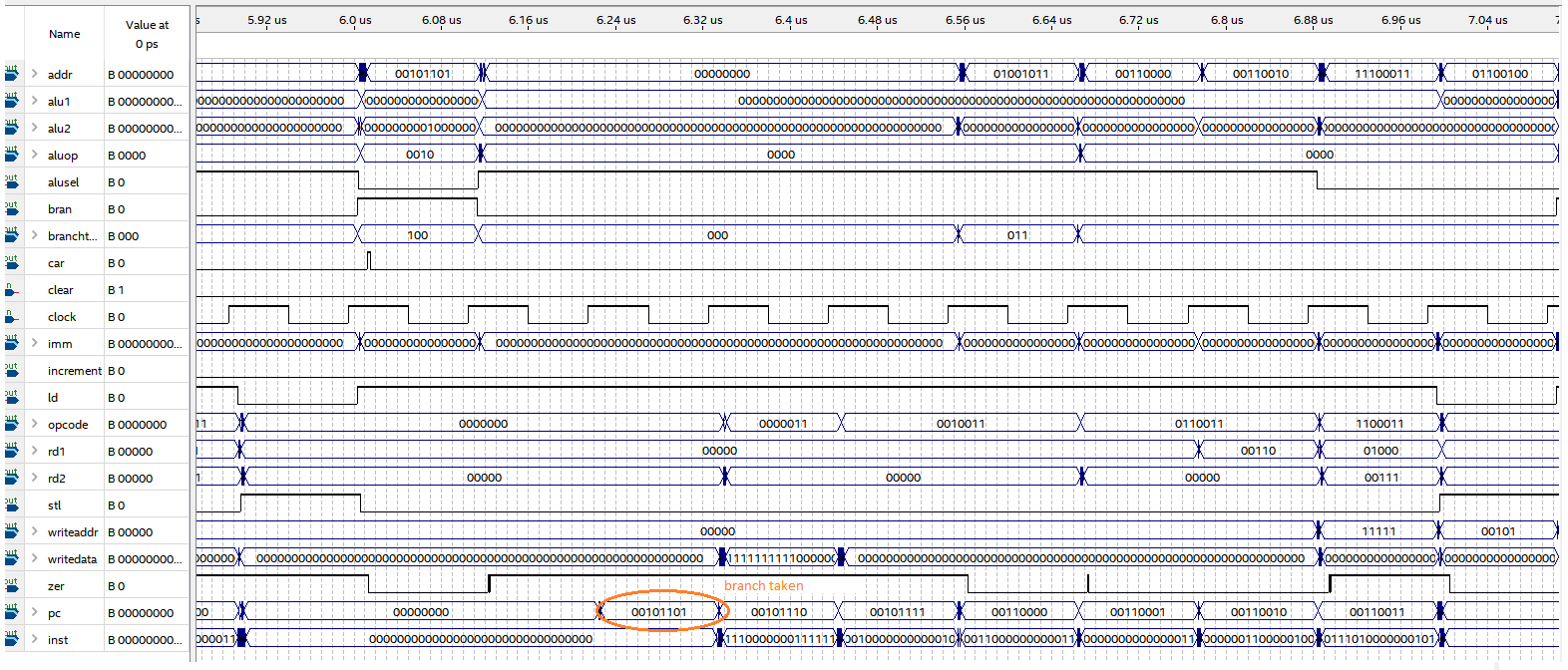
Fully pipelined, no forwarding , no prediction: Datapath1











Fully pipelined with data forwarding: Datapath2

Orange- non branching branch instruction

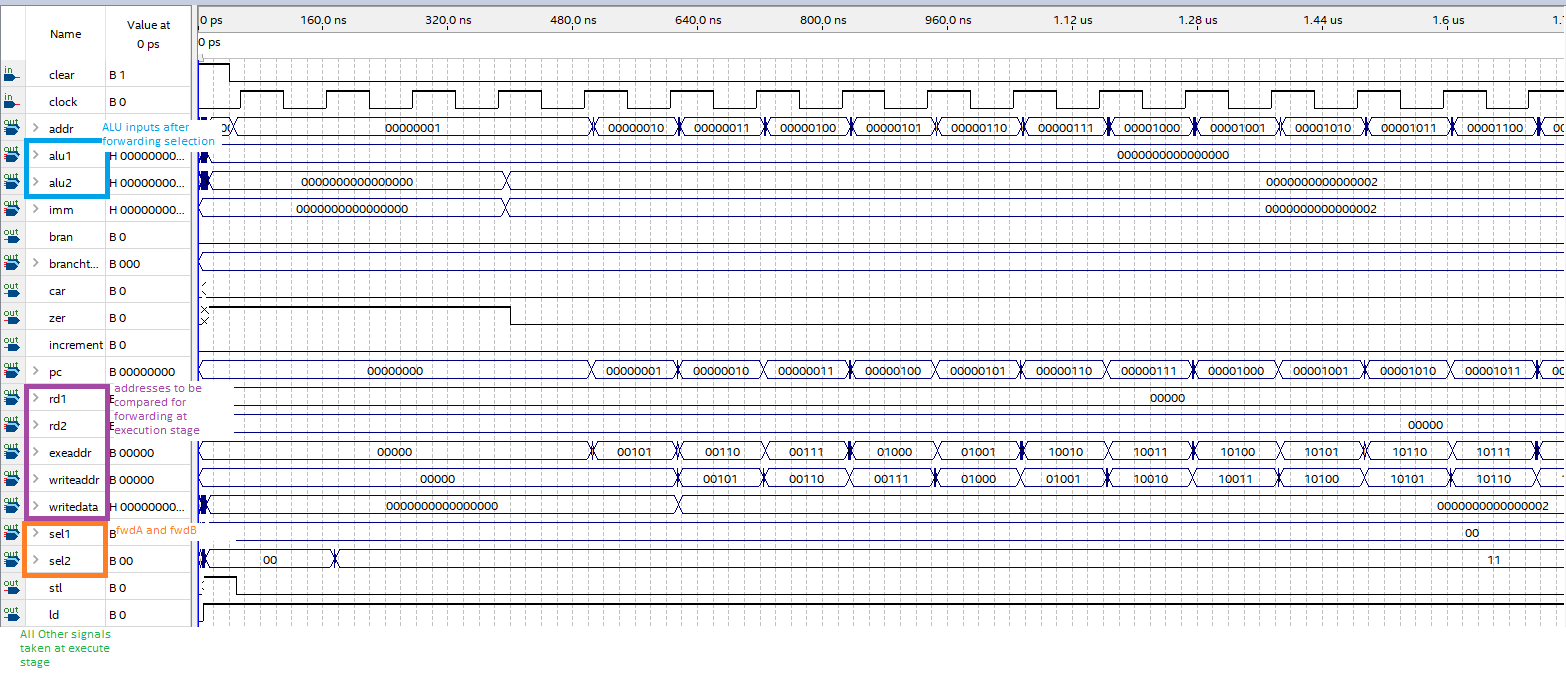
Red- branching branch instruction

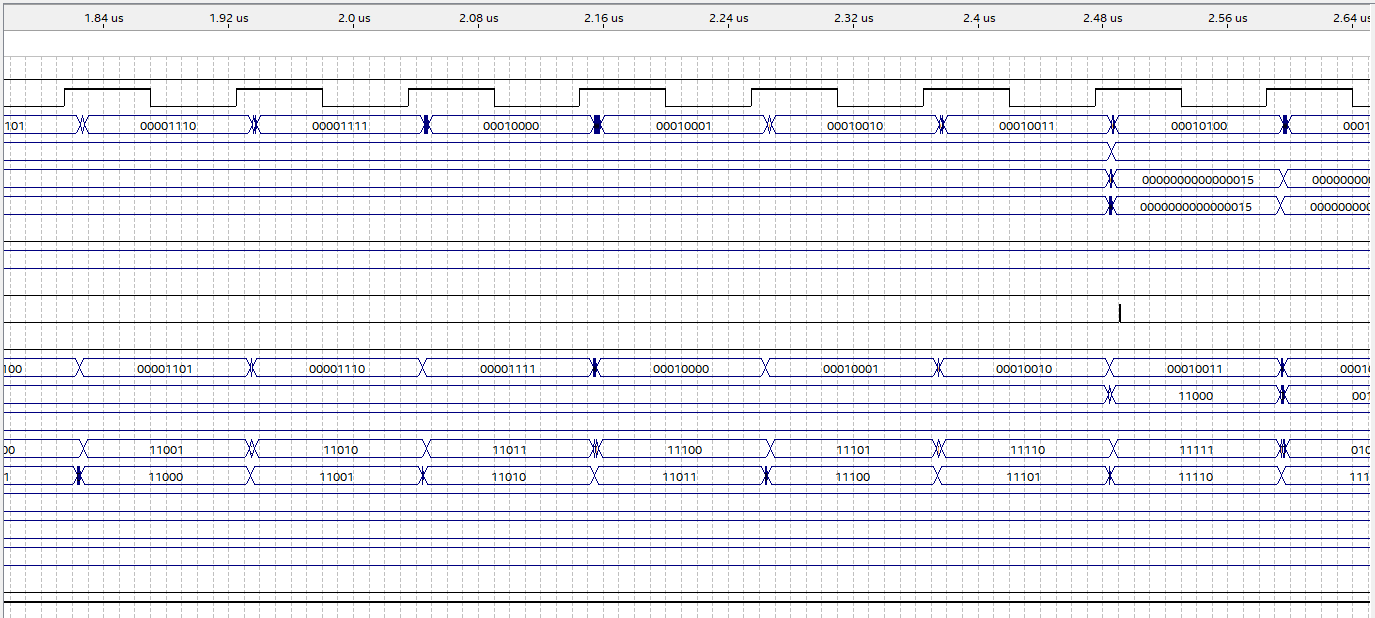
Blue – stall

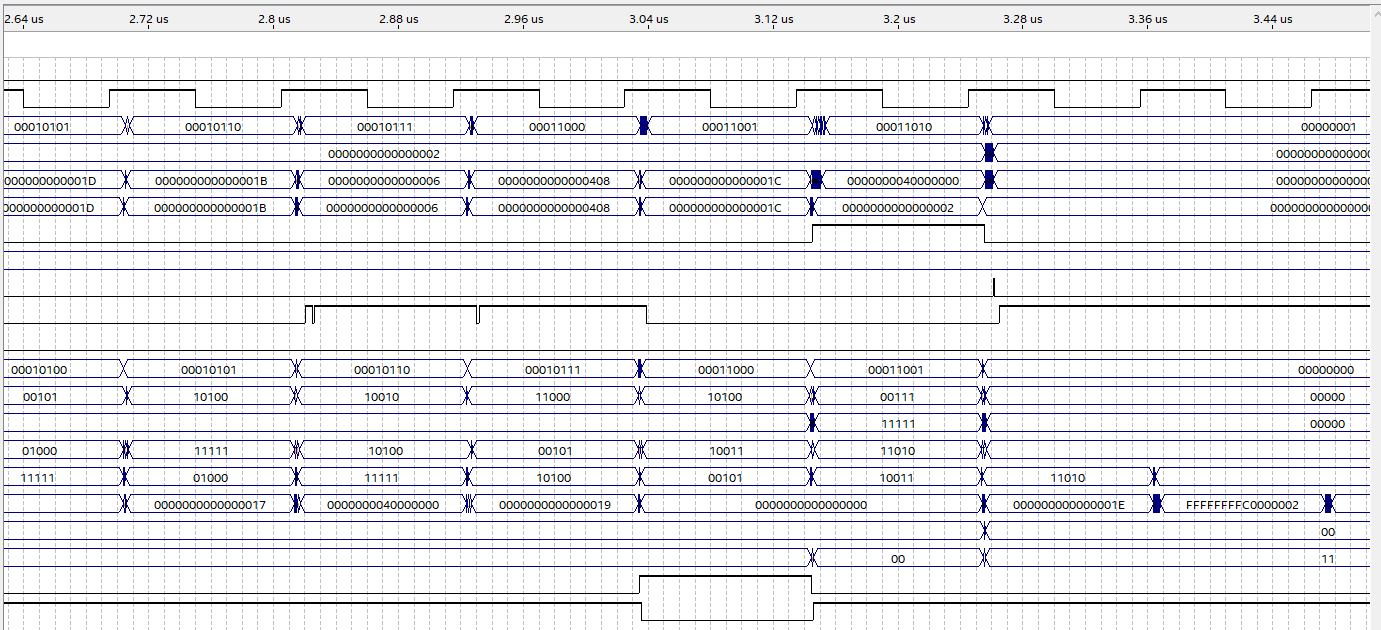
Purple- following branch

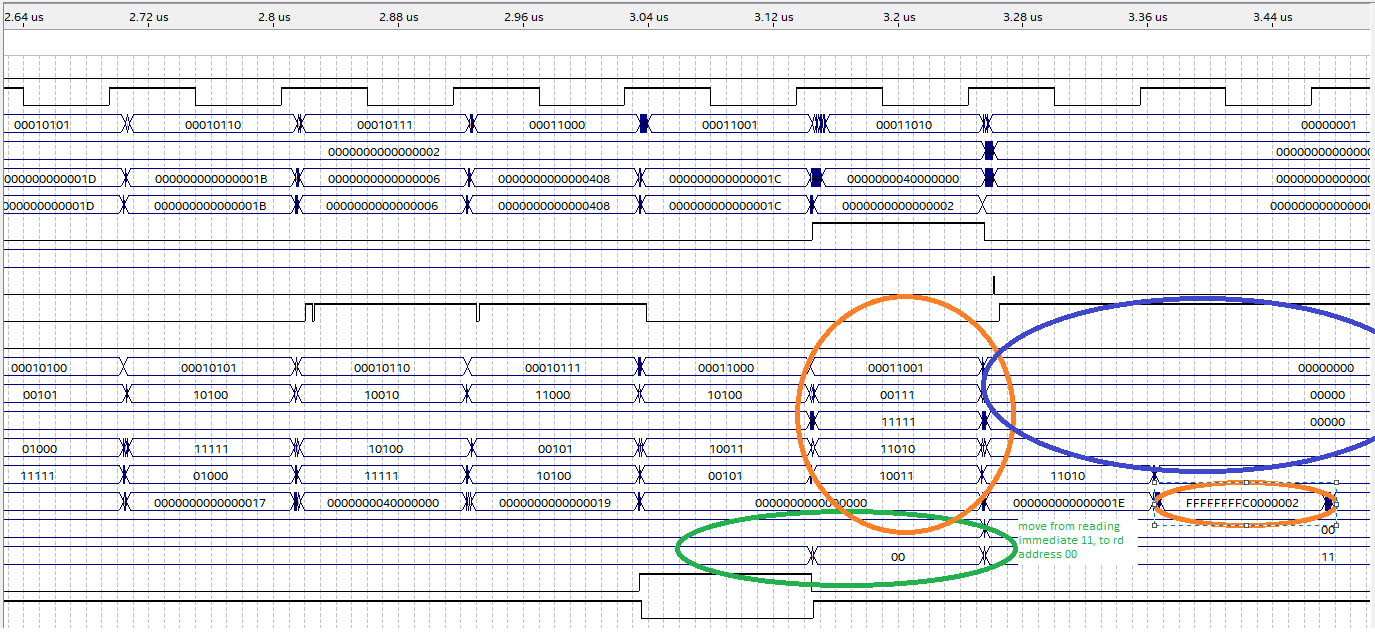
Green- change in forwarding data

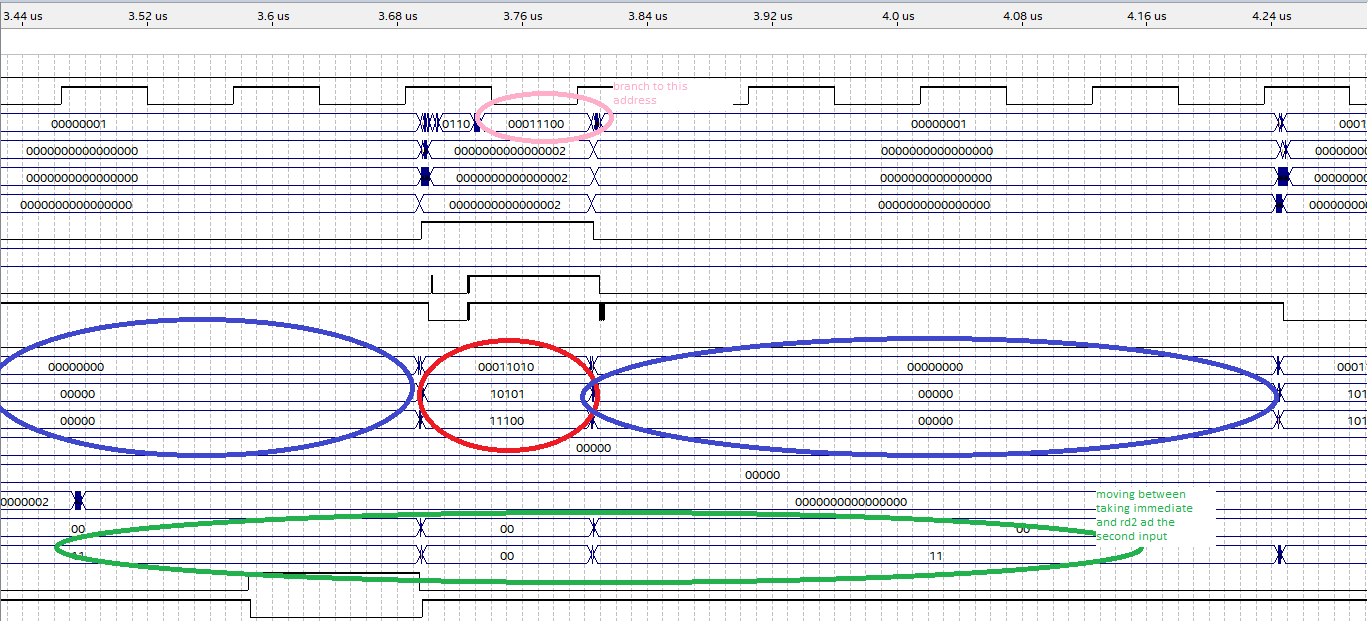
Annotation primarily in instructions preceding C code

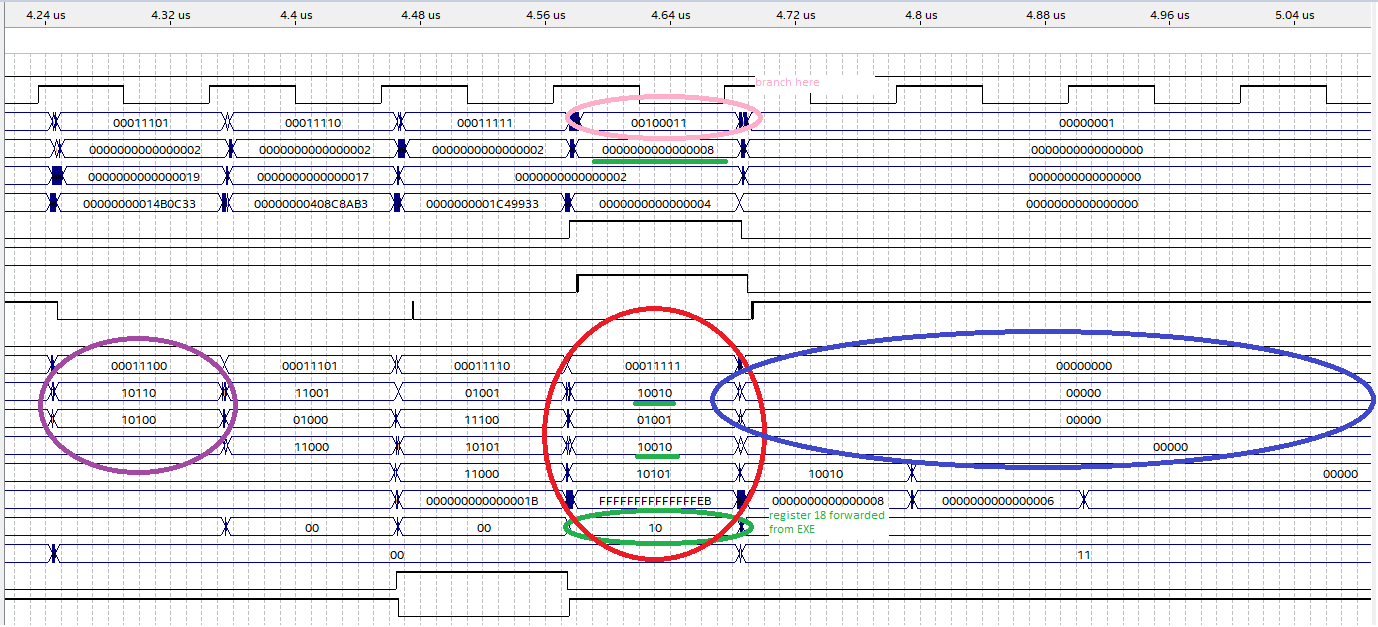


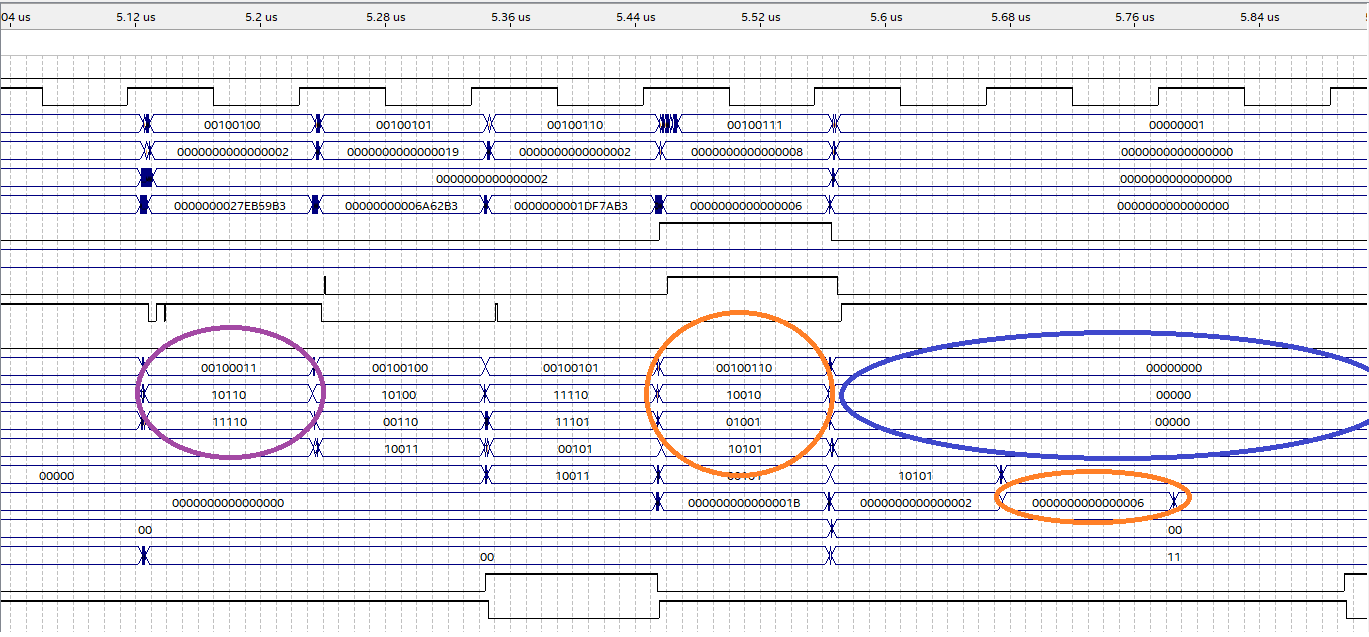


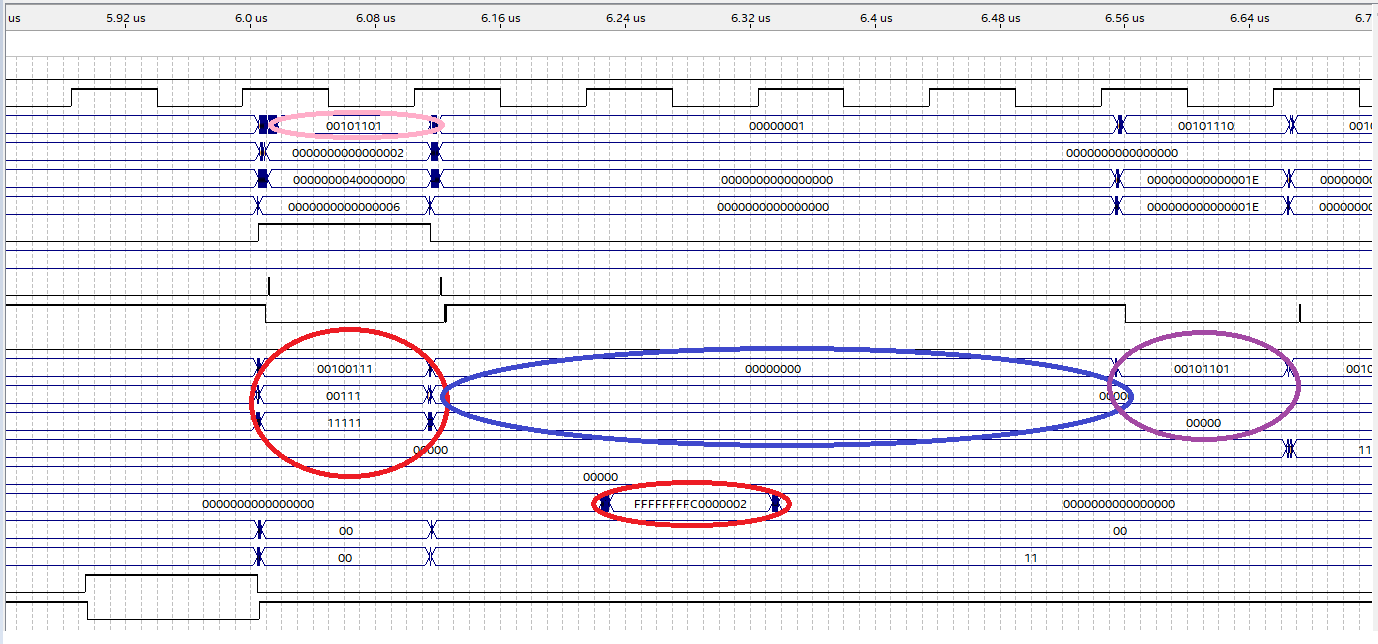


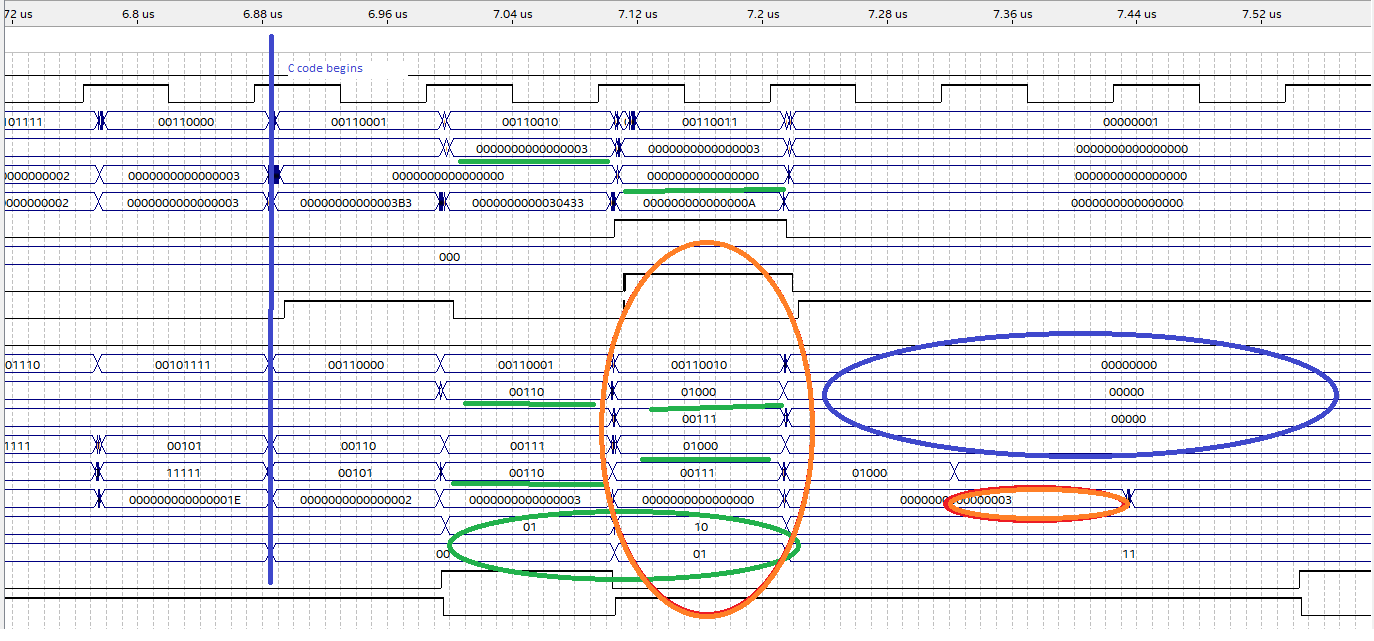


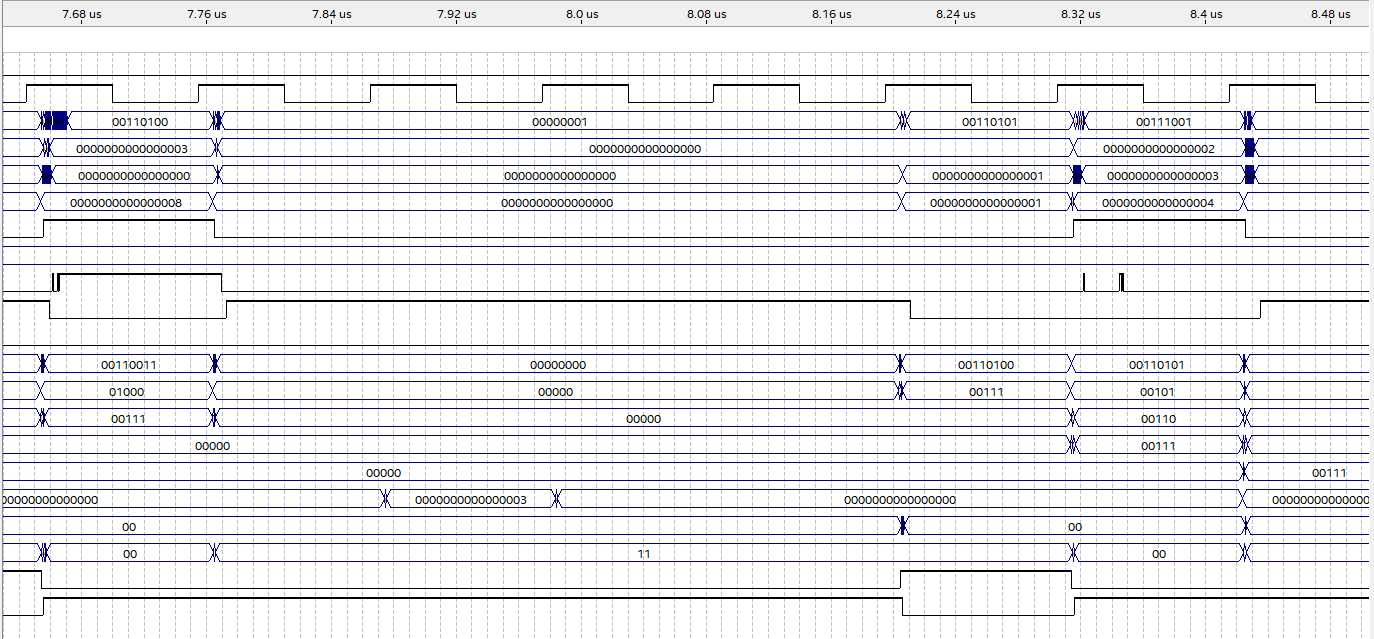


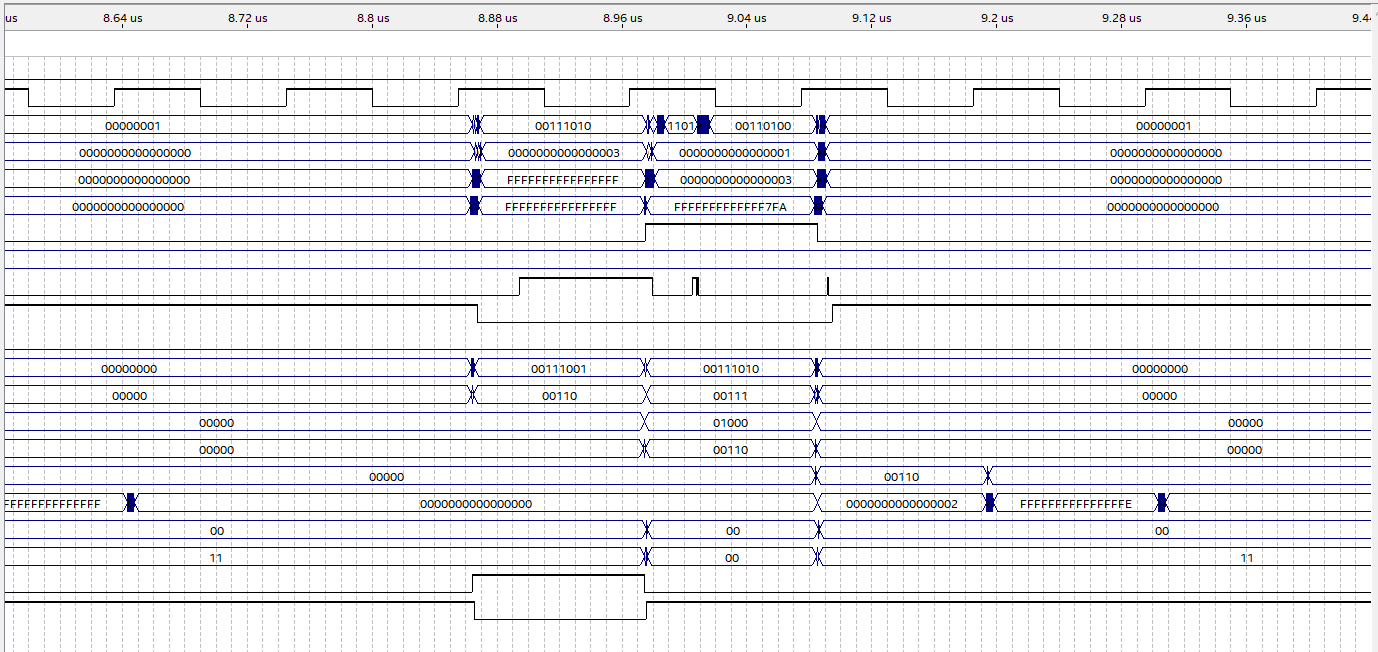


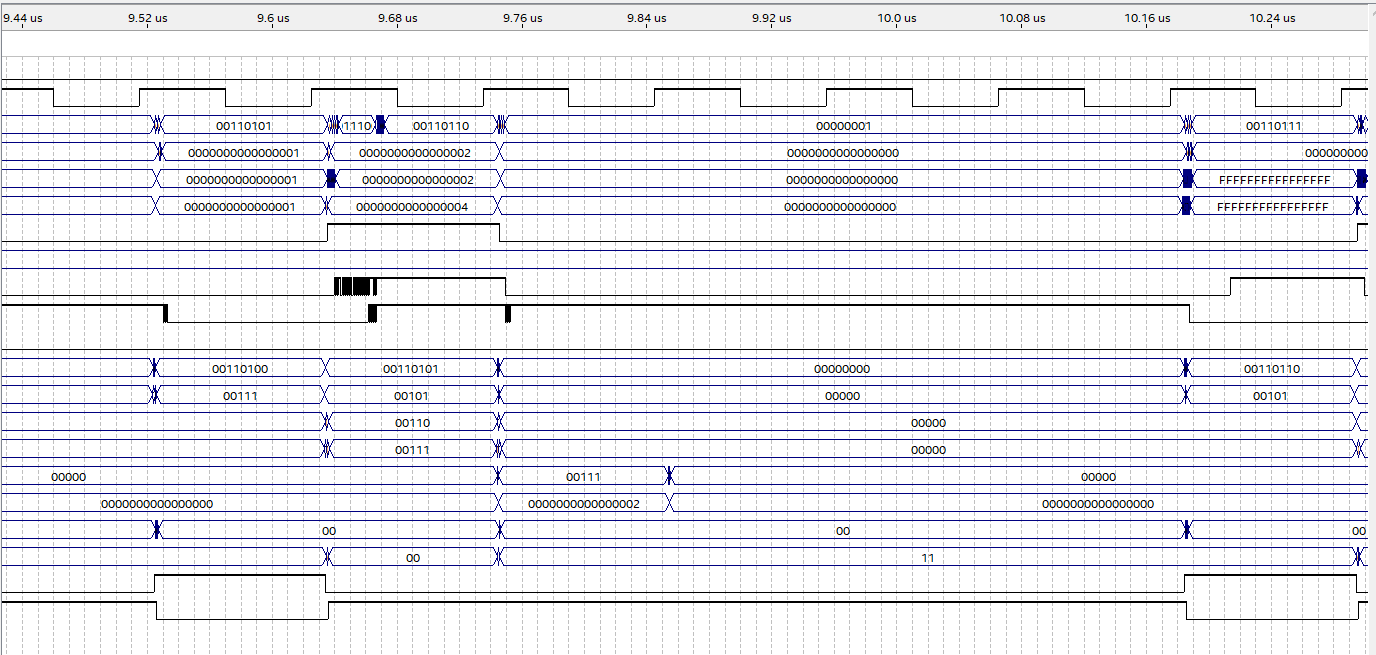


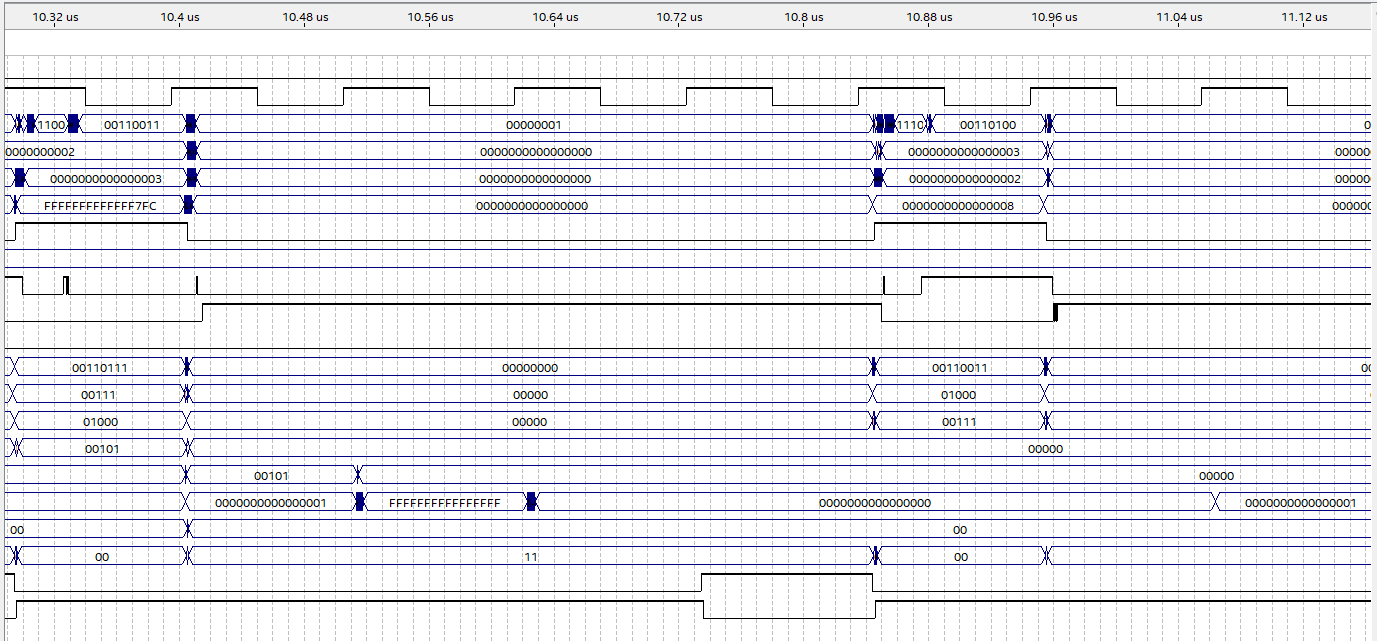


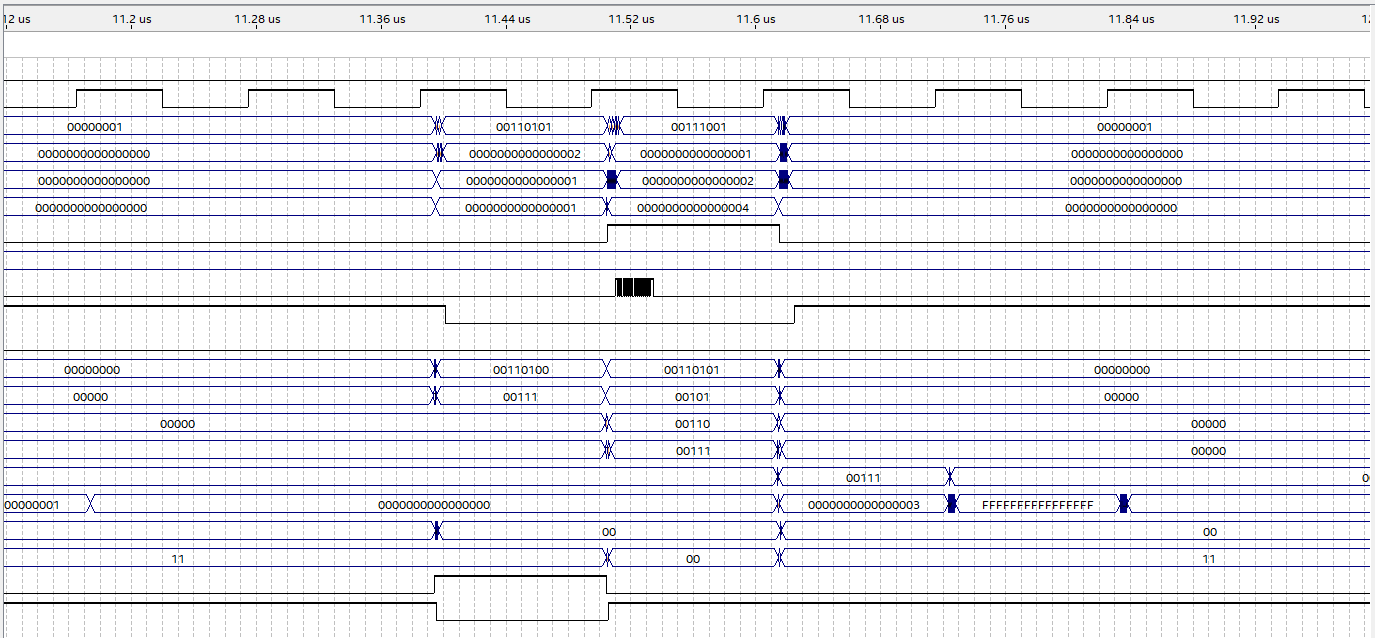


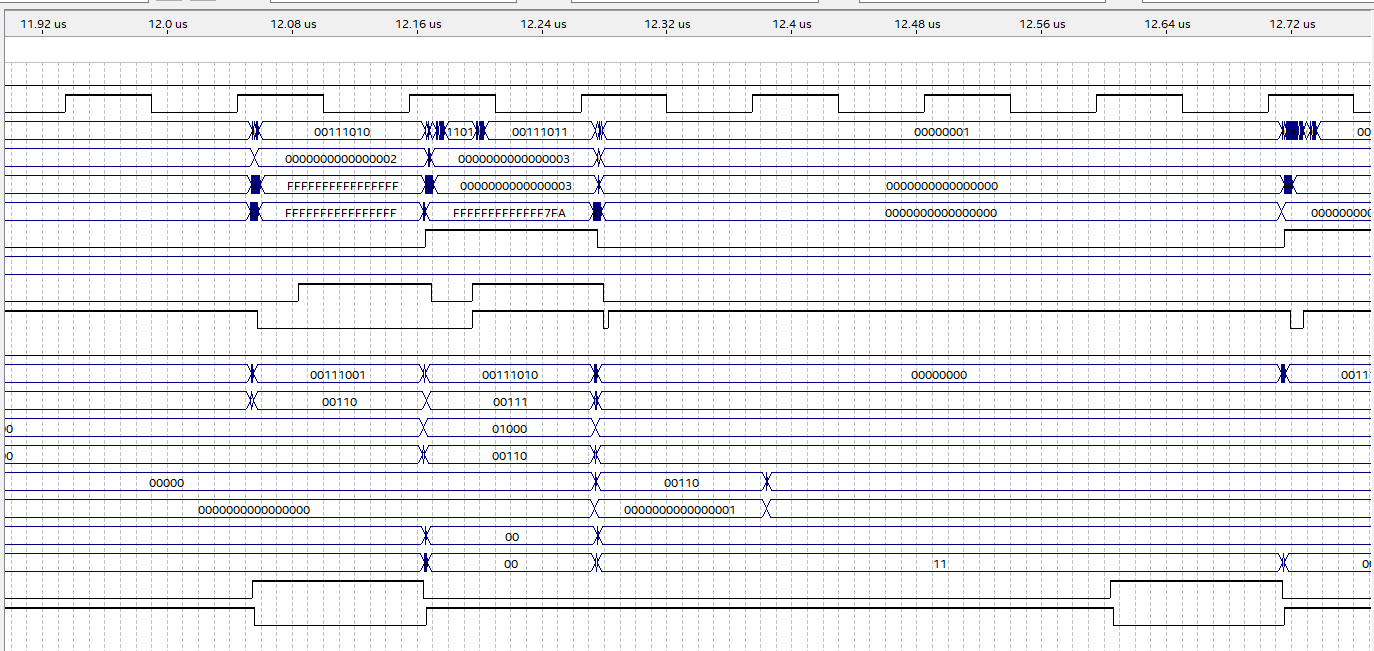


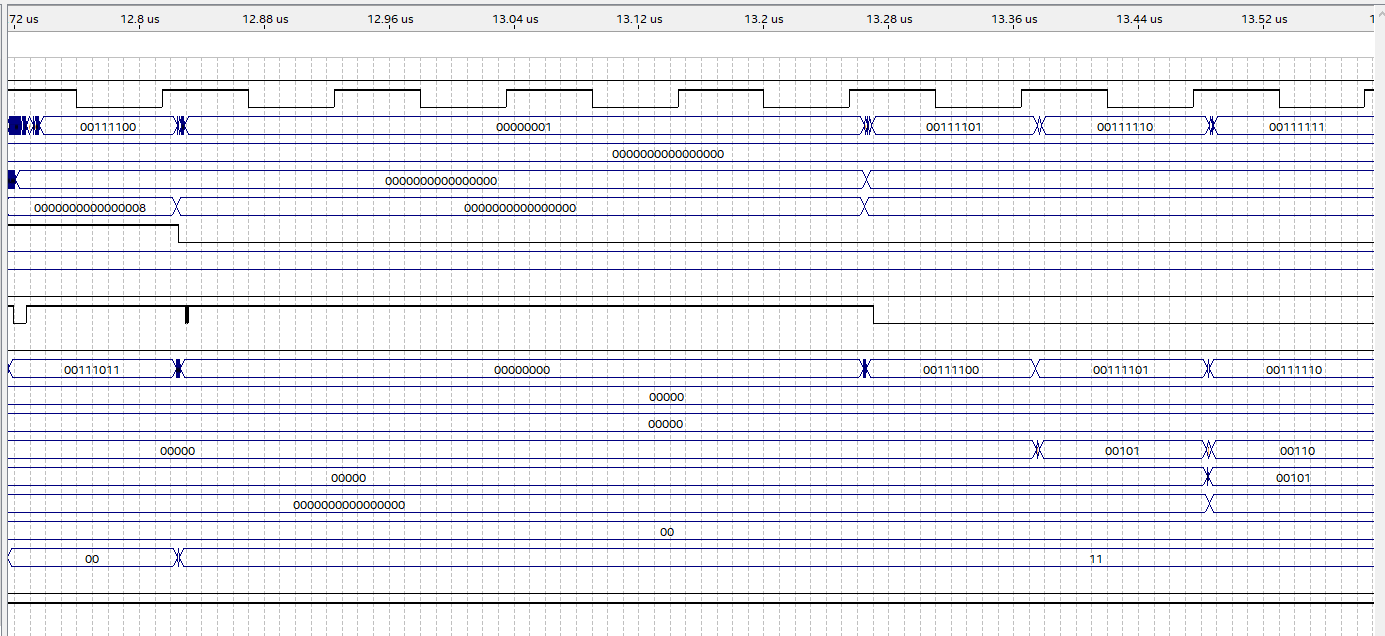












Static branching – never taken

Orange- non branching branch instruction

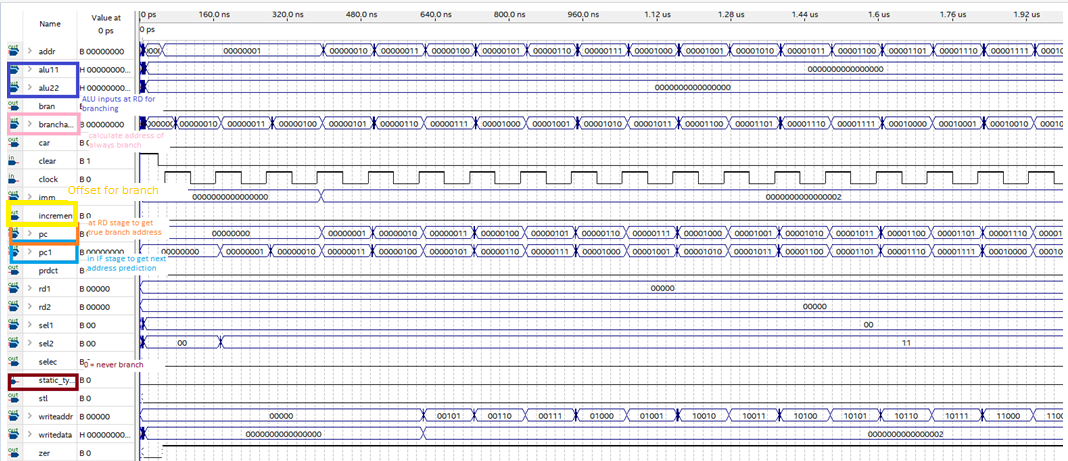
Red- branching branch instruction

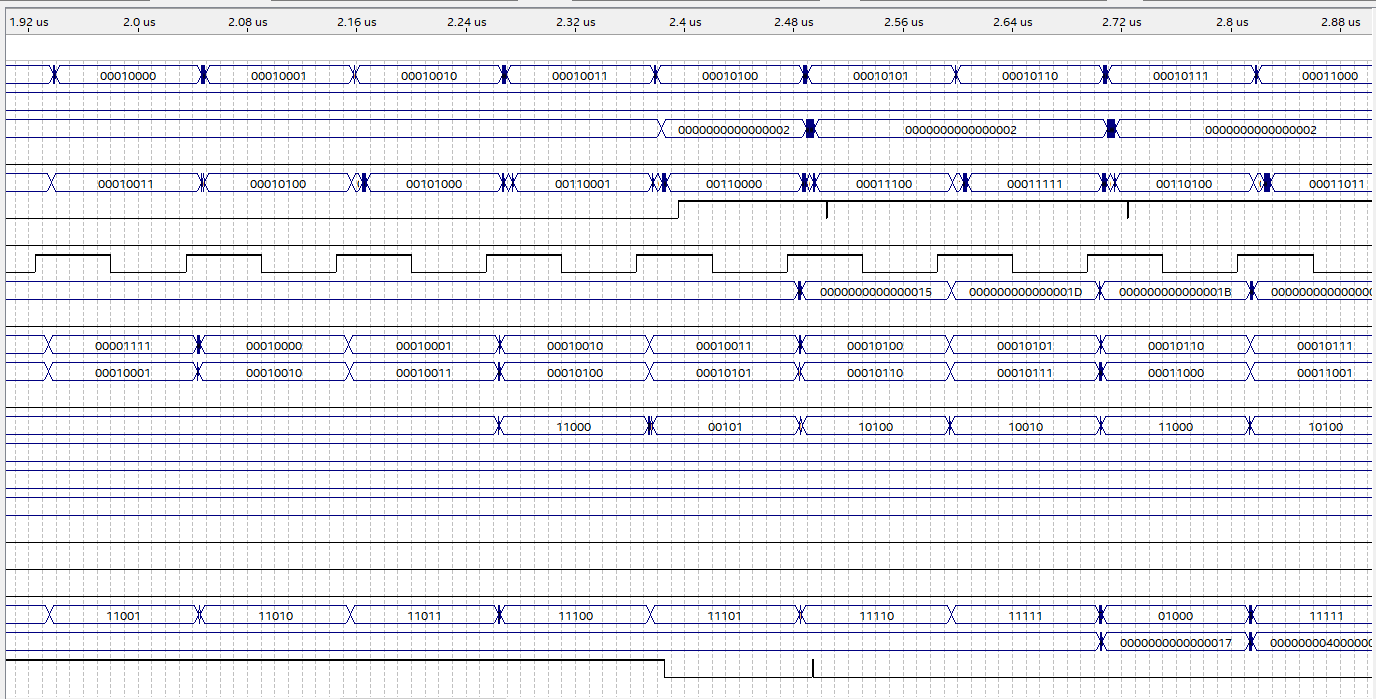
Blue – stall

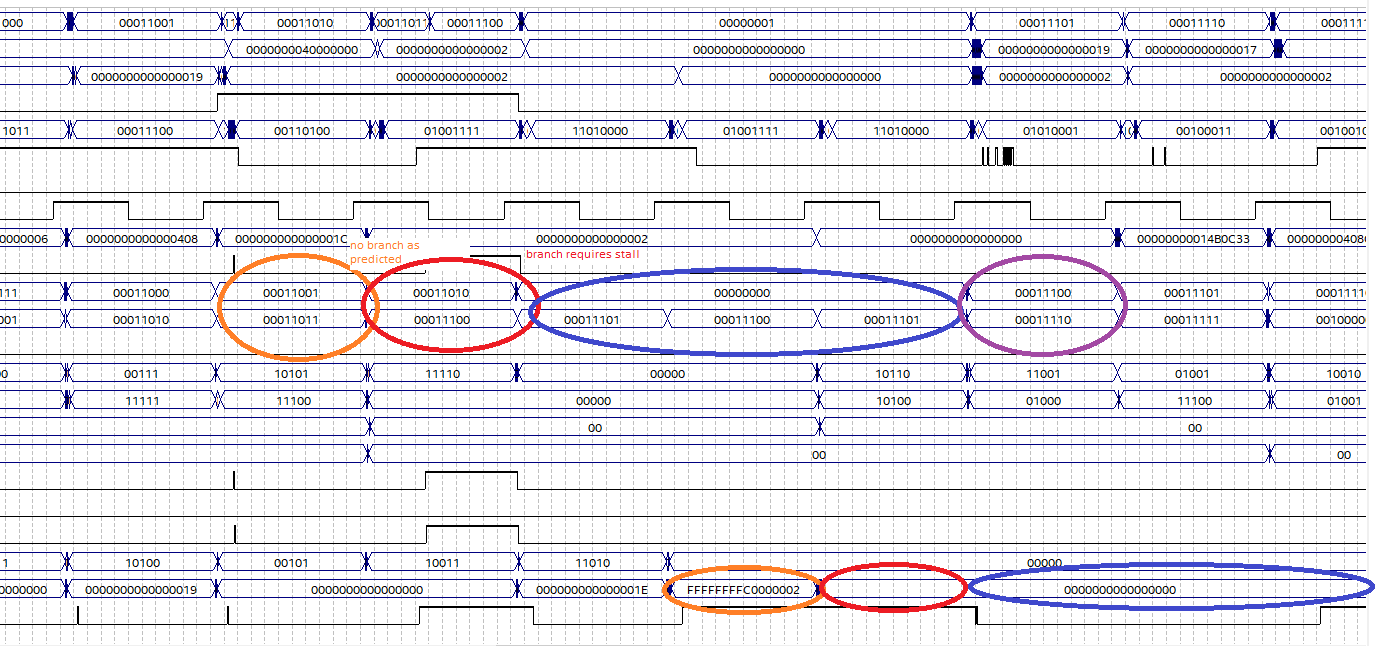
Purple- following branch

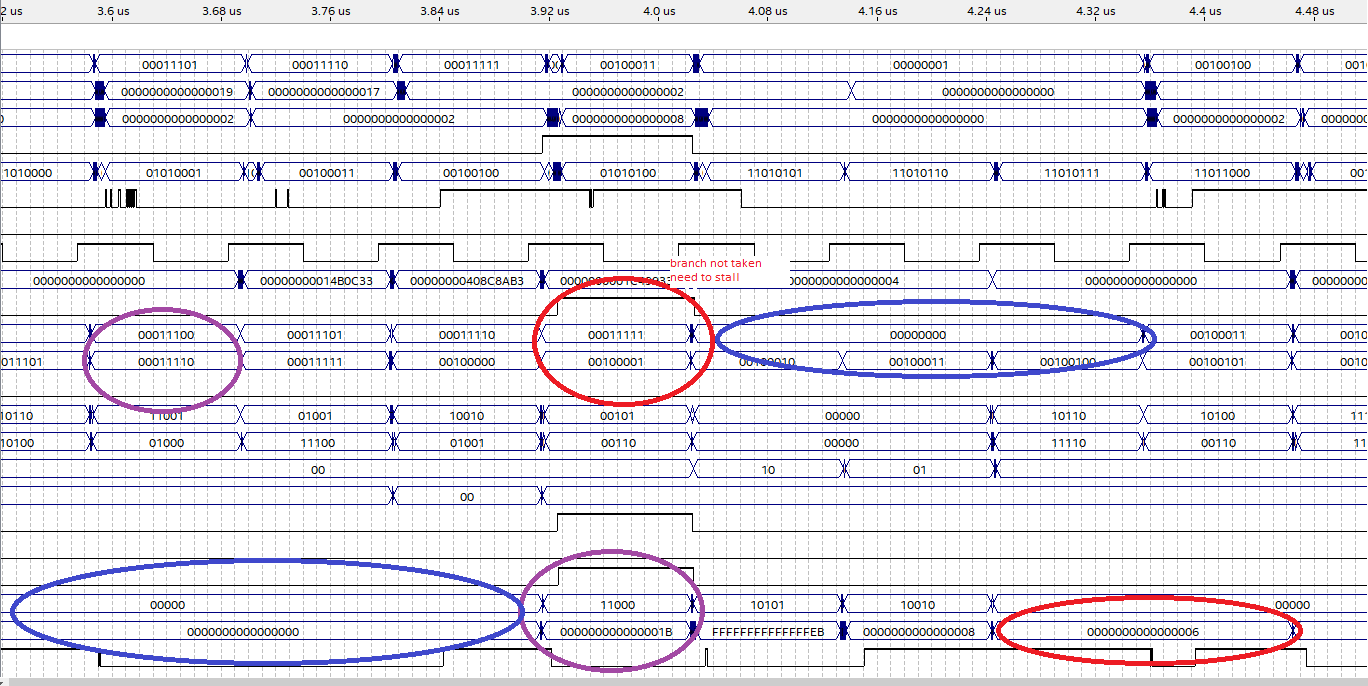
PC increments until stall, then branches

Static\_type = ‘0’ means never taken

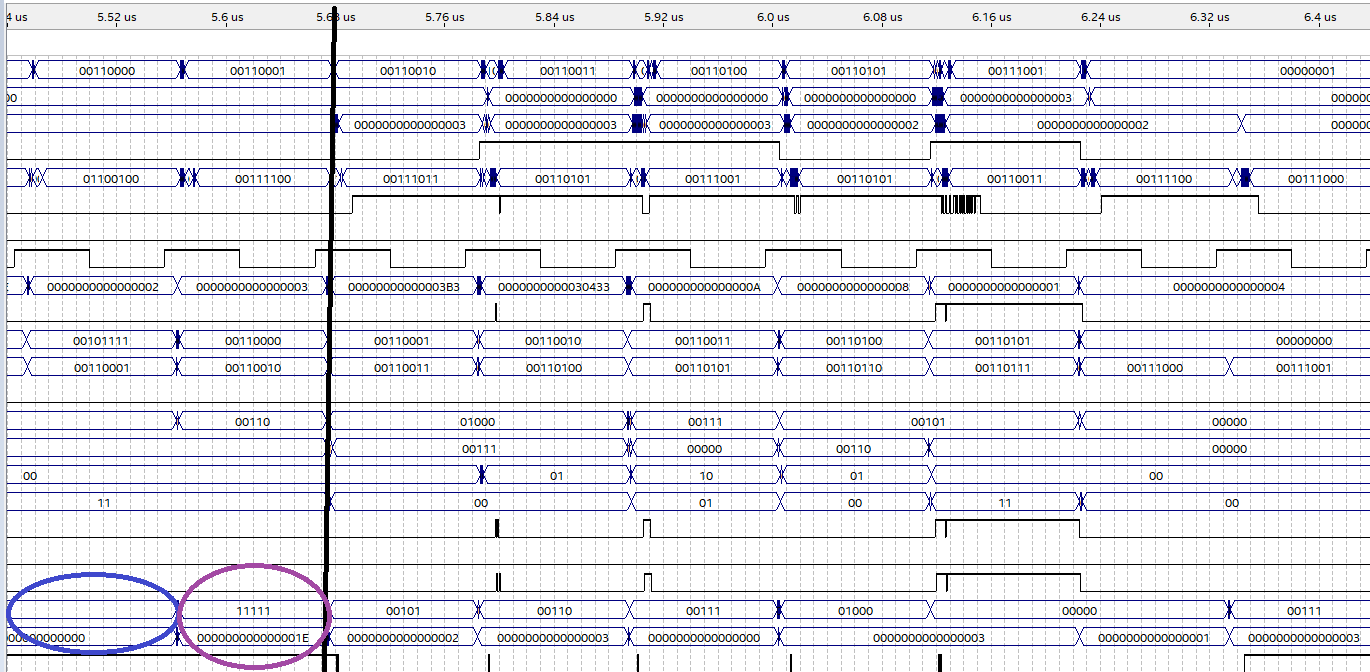












Static branching, always taken

Orange- non branching branch instruction

Red- branching branch instruction

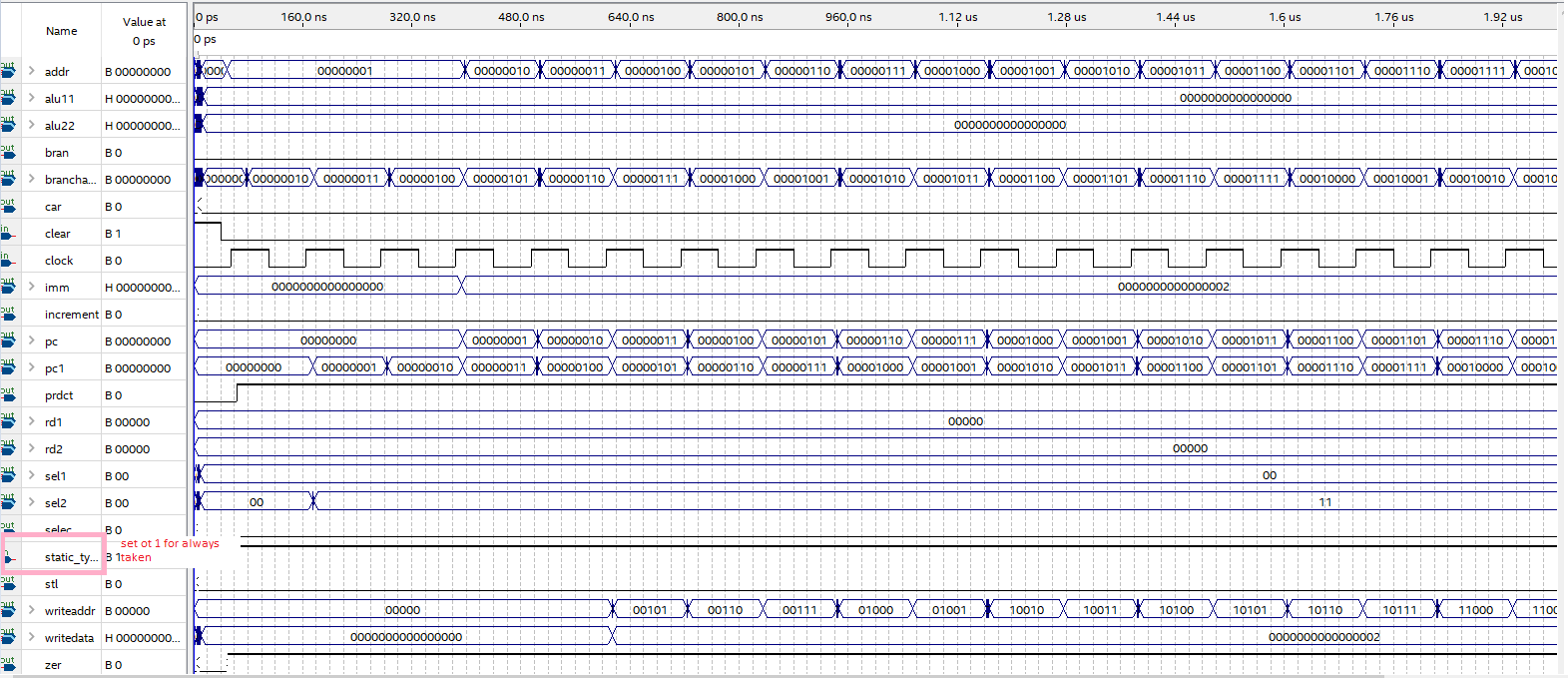
Blue – stall

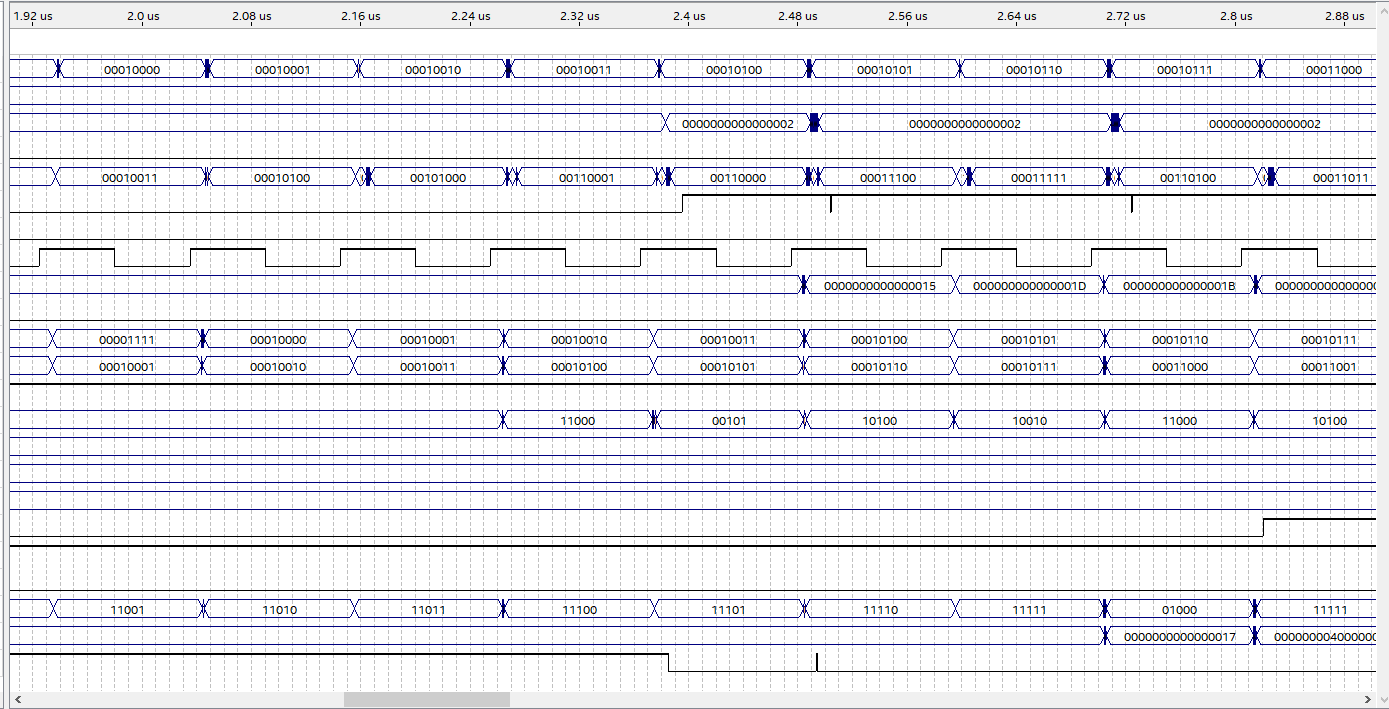
Purple- following branch

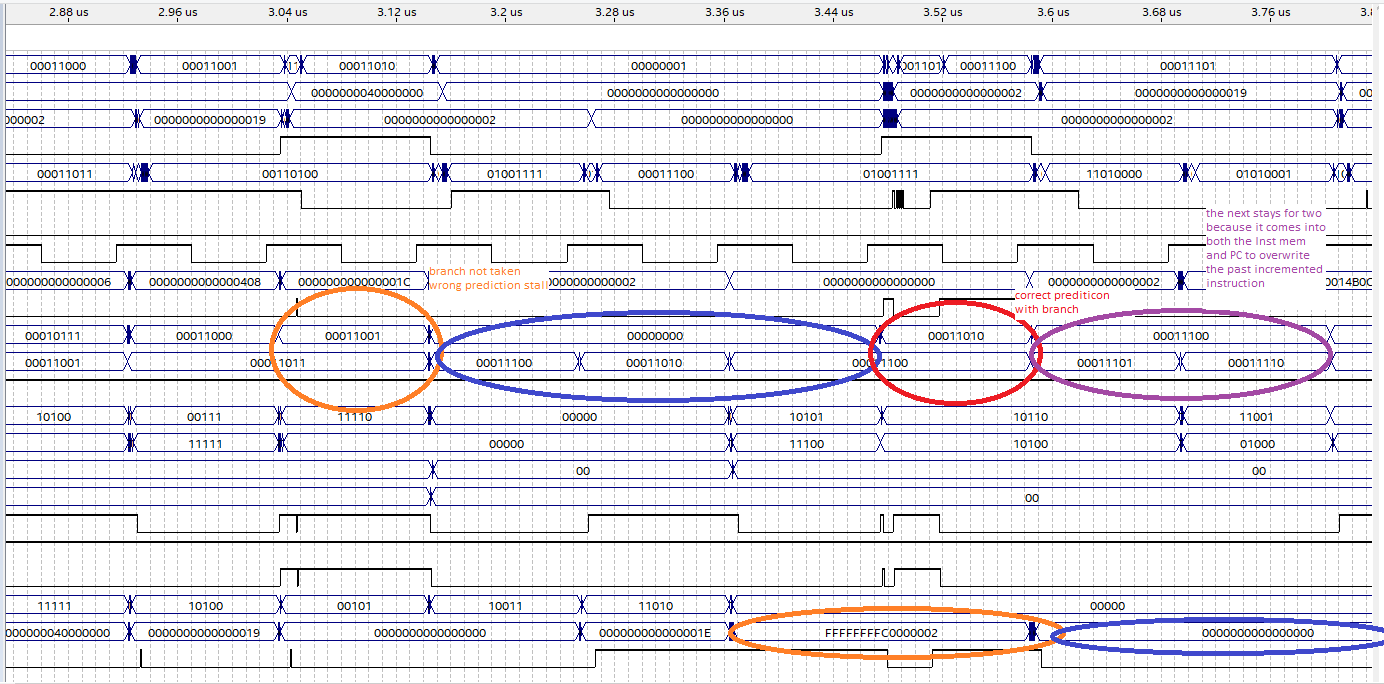
PC then branches until increment then stalls

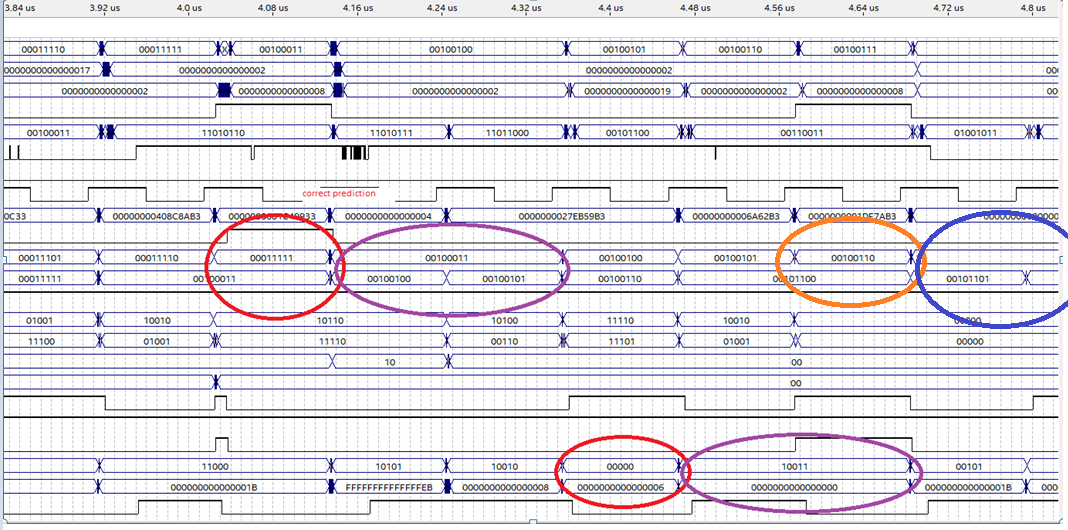
Static\_type = ‘1’ means never taken

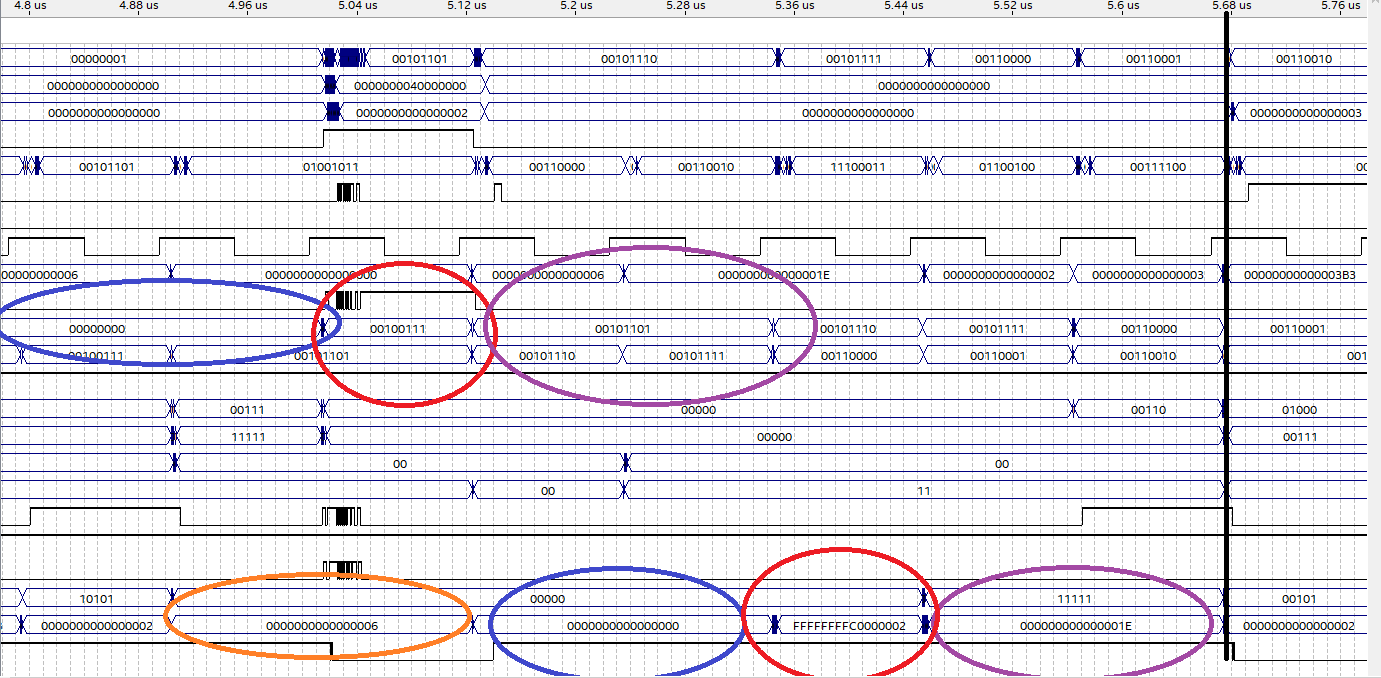
For signal descriptions see previous waveform











Dynamic branching – two bit

Orange- non branching branch instruction

Red- branching branch instruction

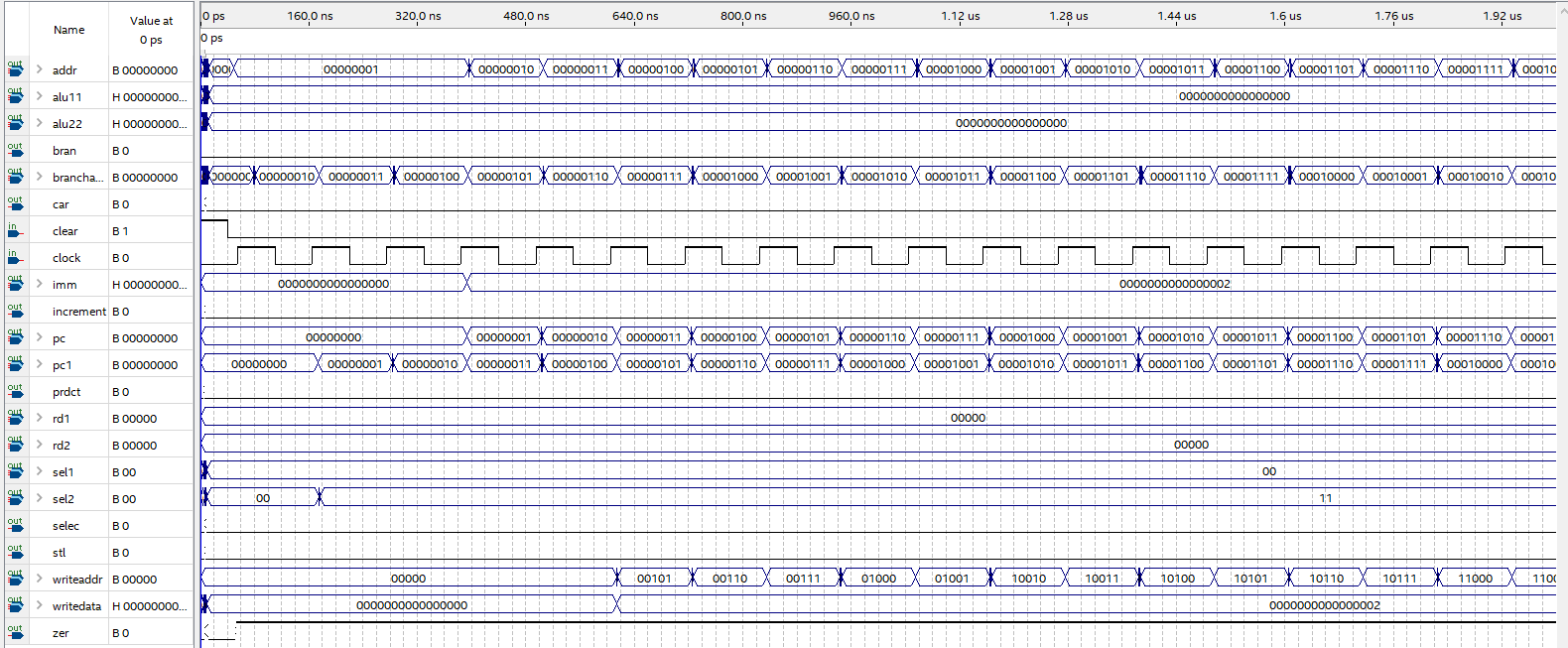
Blue – stall

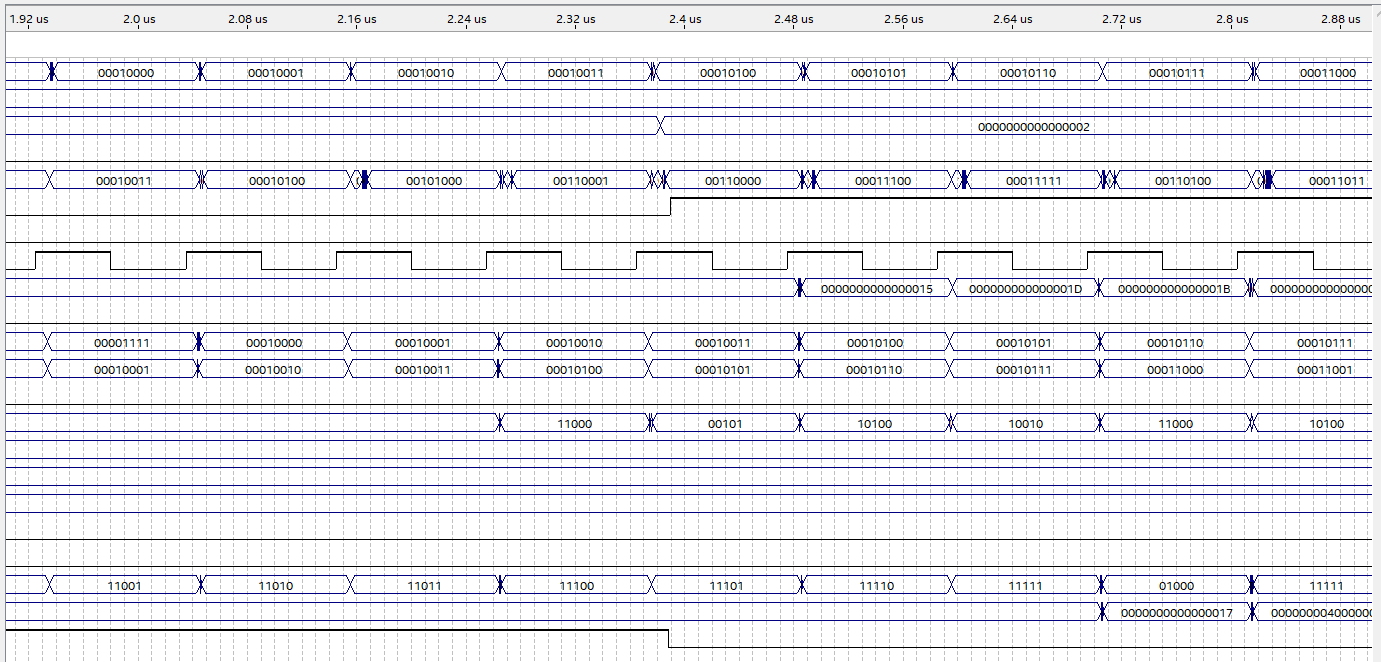
Purple- following branch

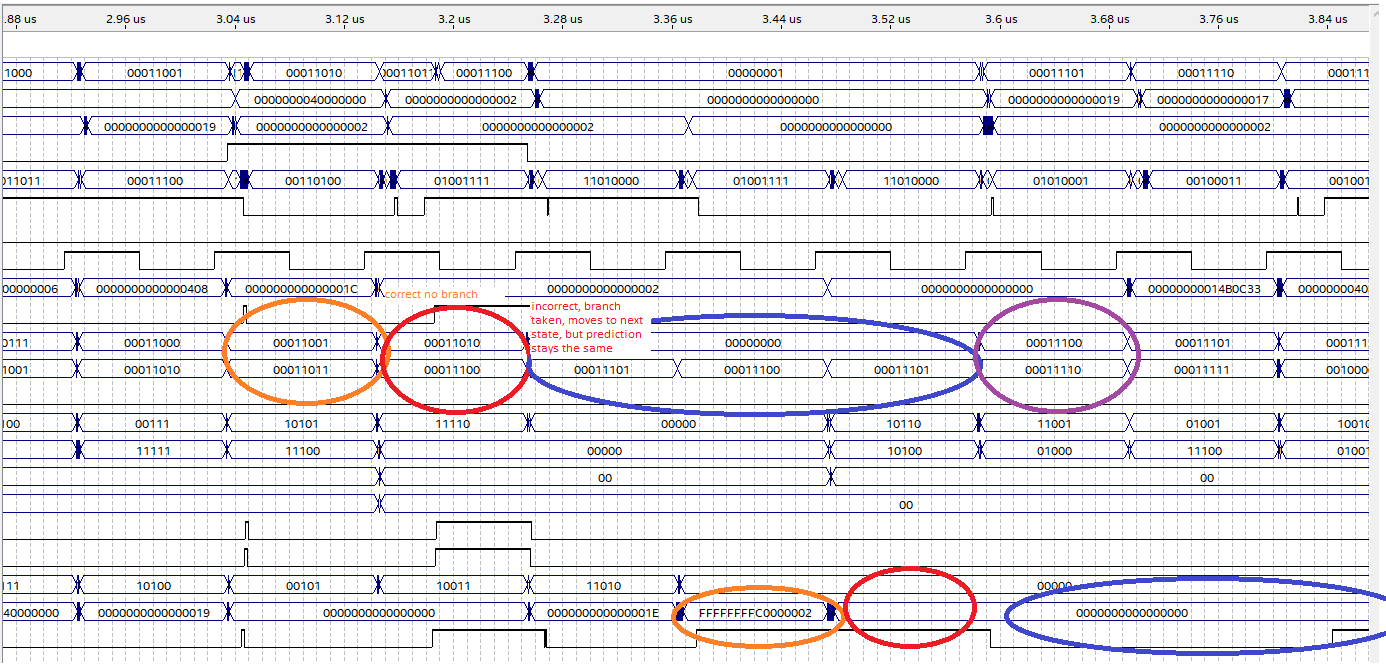
Green – prediction swap

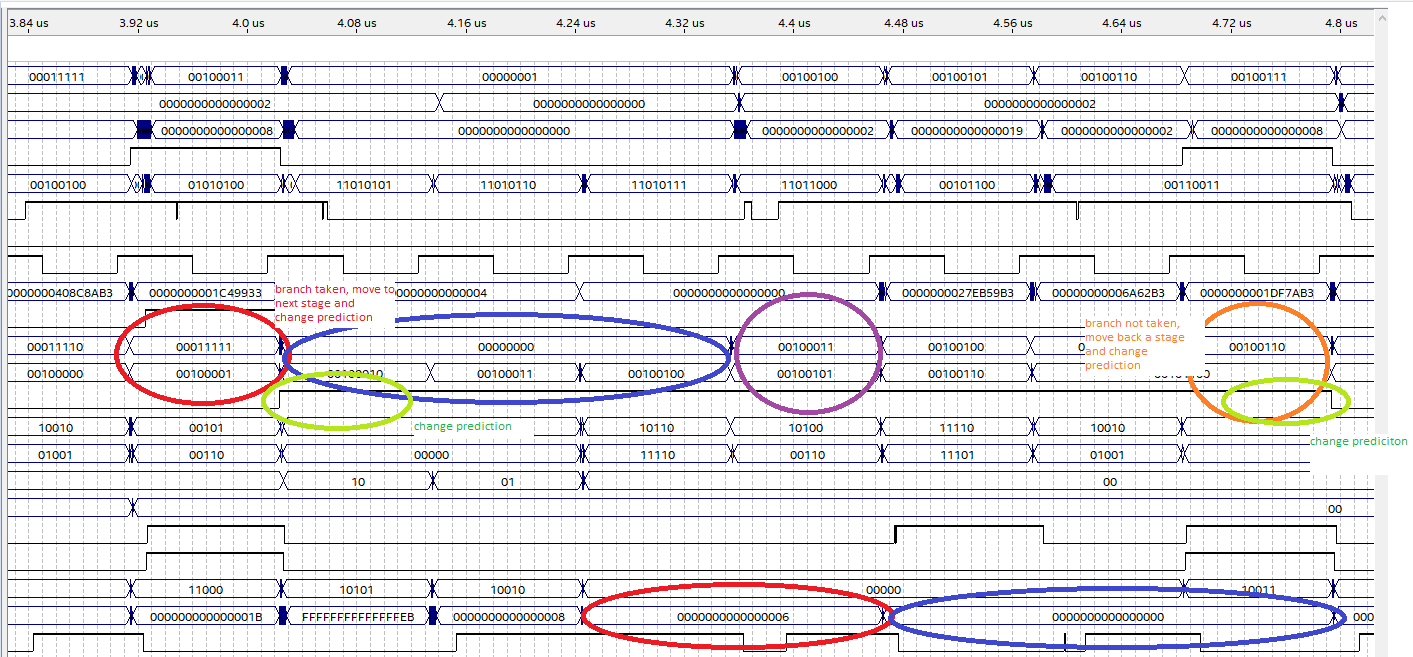
After two incorrect predictions the prediction will change. There are also two intermediate stages when one correct or incorrect has been taken and it can fluctuate between the two.

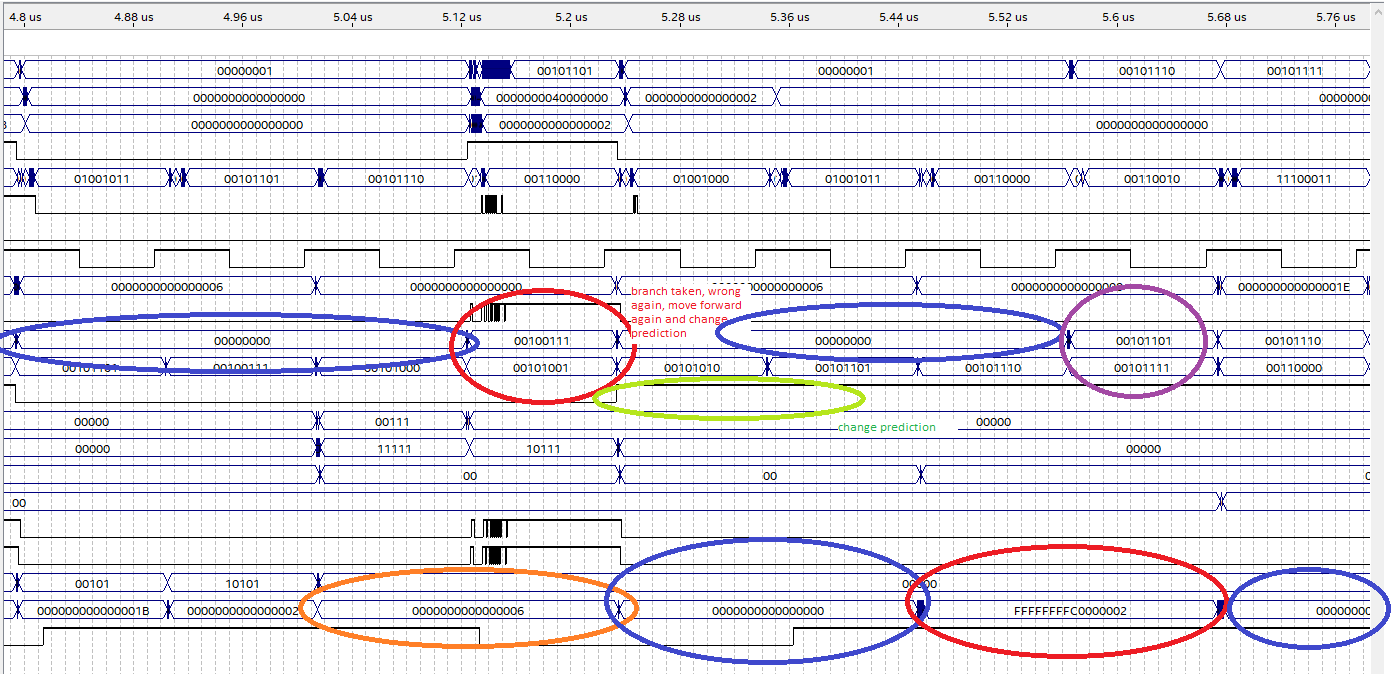
For signal descriptions see static never taken waveform

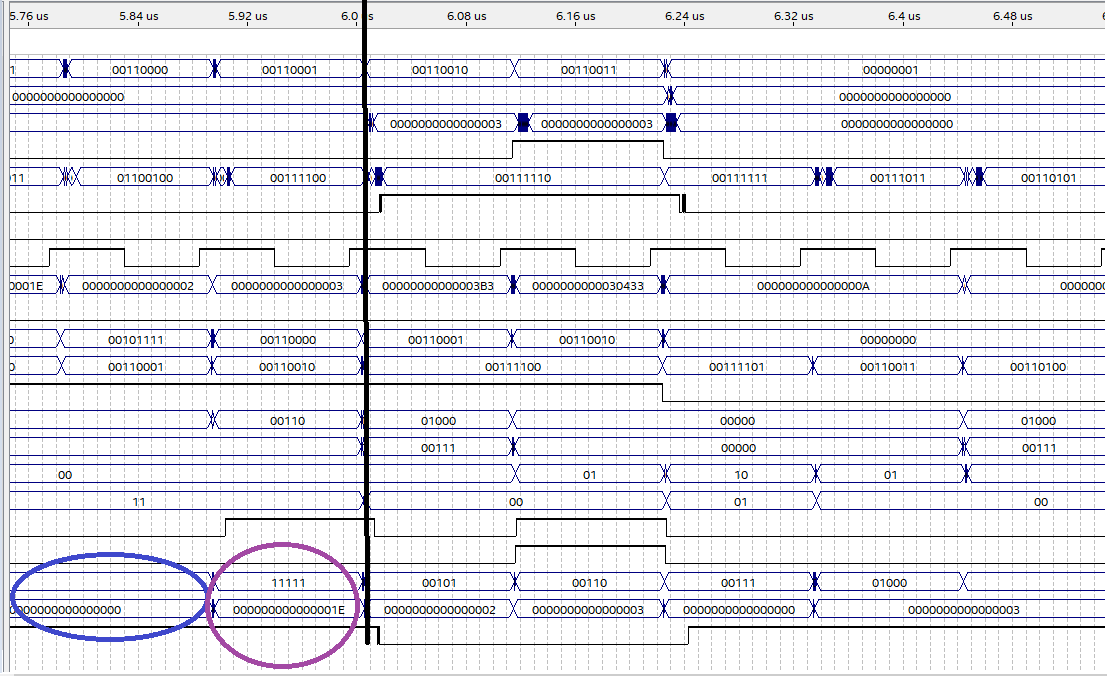












*Operations*

General Operations/moderate experiment:

1: load 0 into register 5

2: load 2 into register 6

3: load 2 into register 7

4: load 2 into register 8

5: load 2 into register 9

6: load 2 into register 18

7: load 2 into register 19

8: load 2 into register 20

9: load 2 into register 21

10: load 2 into register 22

11: load 2 into register 23

12: load 2 into register 24

13: load 2 into register 25

14: load 2 into register 26

15: load 2 into register 27

16: load 2 into register 28

17: load 2 into register 29

18: load 2 into register 30

19: load 2 into register 31

20: add register 24 with immediate store in register8

10+ 10101 = 10111

21: logical left shift register 5 by immediate store in register31

Shift 10 by 11101 = 1000000000000000000000000000000

22: xor register 20 with immediate store in register20

00010

11011 =

11001

23: shift logical right register 18 by immediate store in register5

10 by 110 = 0

24: shift arithmetic right register24 by immediate store in register19

10 by 10000001000 = 0

25: or register20 with immediate store in register26

11001

11100 =

11101

26: branch if register7 equal to register31

Sub 10 and 000000000000000000000000000000001000000000000000000000000000000

1111111111111111111111111111111111000000000000000000000000000010

Not equal NO BRANCH

27: Brach if register21 equal to register28

Sub 10 and 10 = 0

Equal BRANCH -> jump 2 addresses

28: and register30 with immediate store in register21

00010

11001 =

00000

28: add register 22 with register 20 store in register24

10 +11001 = 11011

29: subtract register25 with register 8 store in register21

10 – 10111 = 1111111111111111111111111111111111111111111111111111111111101011

30: logical left shift register 9 by register 28 store in register18

10 by 10 = 1000

31. Branch if register18 not equal to register9

Sub 1000 and 10 = 110

With carry

Not equal BRANCH -> jump 4 addresses

33: branch if register5 not equal to register6

Sub 0 and 10 1111111111111111111111111111111111111111111111111111111111111110

34: xor register 7 with register7 store in register8

10

10 =

0

35: shift logical right register25 by register5 store in register21

10 shift by 0 = 10

32: shift arithmetic right register22 by register30 store in register19

10 by 10 = 0

33: or register20 with register6 store in register5

11001

10 =

11011

34: and register30 with register29 store in register21

10

10 =

10

35: branch if register18 is less than register9

1000 sub 10 = 0110

Carry

Not less than NO BRANCH

36: branch if register 7 is less than register31

Sub 10 and 000000000000000000000000000000001000000000000000000000000000000

1111111111111111111111111111111111000000000000000000000000000010

No carry

Less than BRANCH -> jump 6 addresses

39: Store what is in register23 in data\_mem address 6

Store 10

40: store what is in register9 in data\_mem address7

Store 10

43: store what is in register 25 in data\_ mem address 30

Store 10

44: load from data\_mem address6 to register8

Load 10

45: load from data\_mem address7 to register24

Load 111

37: load from data\_mem addres30 to register31

Load 11110

STARTING THE C CODE

38. add register0 with immediate store in register5

10

39. add register0 with immediate store in register6

11

40. add register0 with register0 store in register7

0

41. add register0with register6 store in register8

11

42. branch if register 8 equal to register 7

11 – 0 = 11

NO BRANCH

43. branch if register 8 less than register 7

11- 0 = 11

NO BRANCH carry = 1

44. add immediate to register7 store in 7

0 + 1 = 1

45. branch if register 6 not equal to register 5

10-11 = 1111111111111111111111111111111111111111111111111111111111111111

BRANCH TO IF offset = 100

46. add immediate to register 6, store in register6

11-01 (+1111111111111111111111111111111111111111111111111111111111111111)= 10

47.branch if register 7 is less than register8

1 – 11 = 1111111111111111111111111111111111111111111111111111111111111110

No carry

BRANCH TO BEGINNING OF LOOP at (address) offset = 1010 (-6)

48. add register 7 and immediate, store in register 7

01+01 = 10

49. branch if register 6 not equal to register 5

10-10 = 0

NO BRANCH

50. add register5 with immediate store in 5

10 – 1 (+1111111111111111111111111111111111111111111111111111111111111111)= 1

51.branch if register 7 is less than register8

10 – 11 = 1111111111111111111111111111111111111111111111111111111111111111

No carry

BRANCH TO BEGINNING OF LOOP at (address) offset = 100 (-4)

52. branch if register 8 less than register 7

11- 10 = 1

NO BRANCH carry = 1

53. add register 7 and immediate, store in register 7

10+1 = 11

54. branch if register 6 not equal to register 5

1-10 = 1111111111111111111111111111111111111111111111111111111111111111

BRANCH TO IF offset = 100

55. add immediate to register 6, store in register6

10 – 1 (+1111111111111111111111111111111111111111111111111111111111111111) = 1

56.branch if register 7 is less than register8

11-11 = 0

NO BRANCH

57.ranch if not equal register 0 register 0

0 – 0 = 0

NO BRANCH

CONTINUE TO OTHER INSTURCTIONS IN THE MIF

Operations for Many Branch in experiment:

1: load 2 into register 5

2: load 2 into register 6

3: load 2 into register 7

4: load 2 into register 8

5: load 2 into register 9

6: load 2 into register 18

7: load 2 into register 19

8: load 2 into register 20

9: load 2 into register 21

10: load 2 into register 22

11: load 2 into register 23

12: load 2 into register 24

13: load 2 into register 25

14: load 2 into register 26

15: load 3 into register 27

16: load 4 into register 28

17: load 5 into register 29

18: load 6 into register 30

19: load 7 into register 31

20. BEQ 22 and 20

10-10 = 0

branch

21. BEQ 9 and 28

10 – 100 = 1111111111111111111111111111111111111111111111111111111111111110

No branch

22. BEQ 7 and 7

10-10 = 0

branch

23.BEQ 22 and 30

10 – 110 = 1111111111111111111111111111111111111111111111111111111111111100

No branch

24. BEQ 20 and 6

10-10 = 0

branch

25. BEQ 21 and 24

10-10 = 0

branch

26. BEQ 19 and 21

10-10 = 0

branch

27. BEQ 19 and 21

10-10 = 0

branch

28. BEQ 24 and 22

10-10 = 0

branch

29. BEQ 29 and 25

101-10 = 11

No branch

30. BEQ 18 and 20

10-10 = 0

branch

31. BEQ 7 and 21

10-10 = 0

branch

32. BEQ 22 and 5

10-10 = 0

branch

33. BEQ 19 and 29

10-101 = 1111111111111111111111111111111111111111111111111111111111111101

No branch

34. BEQ 26 and 26

10-10 = 0

branch

35. BEQ 18 and 27

10 – 11 = 1111111111111111111111111111111111111111111111111111111111111111

No branch

36. BEQ 22 and 20

10-10 = 0

branch

Operation for few branch in experiment:

1: load 2 into register 5

2: load 2 into register 6

3: load 2 into register 7

4: load 2 into register 8

5: load 3 into register 9

6: load 4 into register 18

7: load 5 into register 19

8: load 6 into register 20

9: load 7 into register 21

10: load 8 into register 22

11: load 9 into register 23

12: load 10 into register 24

13: load 11 into register 25

14: load 12 into register 26

15: load 13 into register 27

16: load 14 into register 28

17: load 15 into register 29

18: load 16 into register 30

19: load 17 into register 31

20. BEQ 22 and 20

1000-110 = 10

No branch

21. BEQ 25 and 8

1011-10 =1001

No branch

22. BEQ 9 and 28

11 – 1110 = 1111111111111111111111111111111111111111111111111111111111110101

No branch

23. BEQ 7 and 7

10-10 = 0

branch

24. BEQ 22 and 30

1000 – 10000 = 1111111111111111111111111111111111111111111111111111111111111000

No branch

25. BEQ 20 and 6

110-10 = 100

No branch

26. BEQ 30 and 29

10000 – 1111 = 1

No branch

27. BEQ 21 and 24

111-1010 = 1111111111111111111111111111111111111111111111111111111111111101

No branch

28. BEQ 8 and 18

10 – 100 = 1111111111111111111111111111111111111111111111111111111111111110

No branch

29. BEQ 19 and 21

101-10 = 1111111111111111111111111111111111111111111111111111111111111110

No branch

30. BEQ 21 and 5

111 – 10 = 101

No branch

31. BEQ 19 and 21

101-111= 1111111111111111111111111111111111111111111111111111111111111110

No branch

32. BEQ 22 and 18

1000 – 100 = 100

No branch

33. BEQ 24 and 22

1010-1000 = 10

No branch

34. BEQ 23 and 7

1001 – 10 = 111

No branch

35. BEQ 29 and 25

1111-1011 = 100

No branch

36. BEQ 18 and 20

100-110 = 1111111111111111111111111111111111111111111111111111111111111110

No branch

37. BEQ 28 and 20

1110-110 = 1000

No branch

38. BEQ 7 and 21

10-111 = 1111111111111111111111111111111111111111111111111111111111111011

No branch

39. BEQ 22 and 30

1000 – 10000 = 1111111111111111111111111111111111111111111111111111111111111000

No branch

40. BEQ 22 and 5

1000 – 10 = 110

No branch

41. BEQ 22 and 31

1000 – 10001 = 1111111111111111111111111111111111111111111111111111111111110111

No branch

42. BEQ 19 and 29

101-1111 = 1111111111111111111111111111111111111111111111111111111111110110

No branch

43. BEQ 26 and 26

1100-1100 = 0

branch

44. BEQ 18 and 27

100 – 1101 = 1111111111111111111111111111111111111111111111111111111111110111

No branch

45. BEQ 22 and 20

1000-110 = 10

No branch

46. BEQ 6 and 21

10 – 111 = 1111111111111111111111111111111111111111111111111111111111111011

No branch