1 Protocol Test Suite

1.1 Protocol Testing

Protocol testing can be classified into three subcategories, each subcategory defining one part of the testing process:

- Protocol Validation,
- Conformance Testing,
- Implementation Assessment.

Protocol validation deals with formal protocol description. Its aim is to find logical errors on formal protocol specifications. Conformance testing is used to check consistency of the protocol implementation with its specifications. That is, whether the external behavior of a given implementation of a protocol is equivalent to its formal specifications. From that point of view, it is obvious that, if there is a logical error on the protocol specifications, the conformance test will not catch an error unless there is another error on conformance test generation. Implementation assessment, on the other hand, is more general. It is mostly related with properties of the protocol that are not part of the protocol specifications such as how the implementation reacts to unexpected (invalid) user commends, how many simultaneous connections can be supported, or evaluation of the performance of the implementation compared with analytical results related with the given protocol and with competitor protocols. Therefore, from the point of the implementation assessment, the implementation environment should be taken into account unless the conformance testing for which the implementation details are unknown.

The test process can be illustrated using Figure 1.

1.2 Protocol Validation

The aim of the protocol validation is to define the protocol specifications in a formal language and then test the logical consistency of the protocol description. The first step of the protocol validation has been performed in Appendix J by defining the verbal protocol specifications with an EFSM model. The next step on the validation process is to check for logical consistencies. The classical validation technique is to make a reachability analysis. There are three approaches for reachability analysis: full search(for systems up to 10^5 states), controlled partial search (for systems up to 10^8 states) and random simulation (for larger systems). Jumpstart protocol has approximately 400 states as a compound system machine. Therefore, a full search using a exhaustive search algorithm is the best candidate. The result of this search will create all valid sequences, eliminating erronous sequences. However, unlike the algorithms used on communicating finite state machines, in this section, using decomposition technique, we will apply the full search technique to each individual state machine independently,

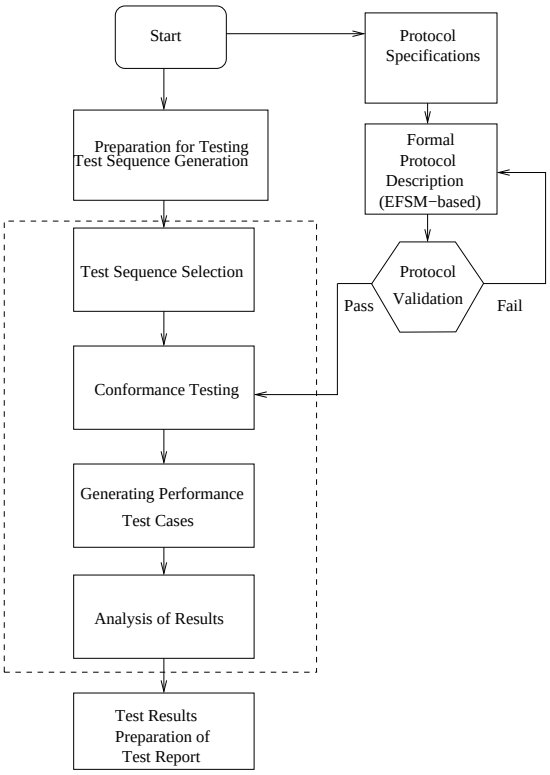

Figure 1: Protocol Test Process

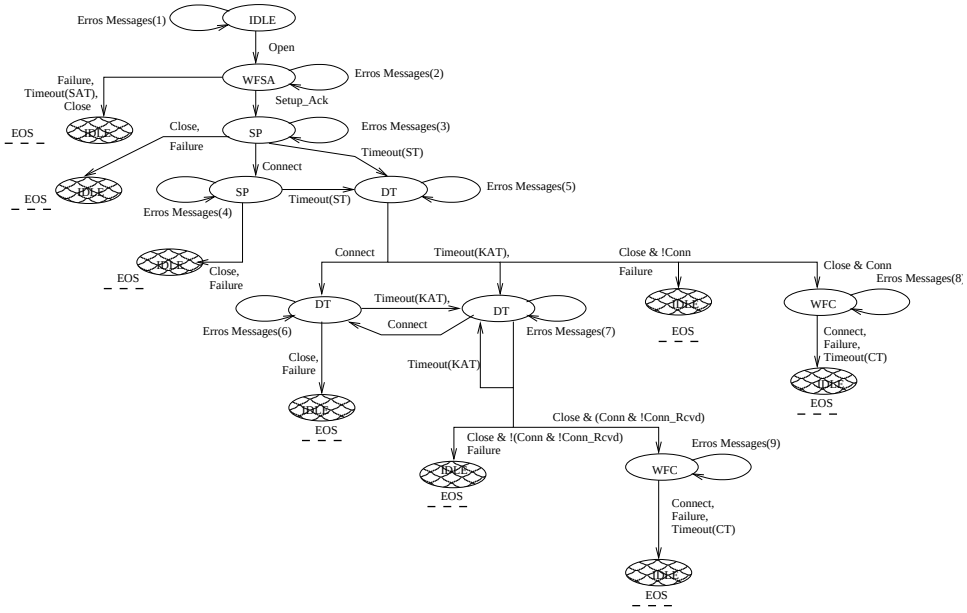

Figure 2: Reachability Tree for the Source

because the EFSMs do not make state transitions at the same time. The rest of this section is devoted to the reachability analysis of individual state machines for the source, destination, ingress switch and intermediate switch. EFSMs will not be redefined here.

1.2.1 Reachability Analysis for the Source

Source state diagram has five states, and 11 messages. In this subsection, a tree structure is used to show the reachability starting from the initial state IDLE as the root of the tree. The reachability tree is shown in Figure 2.

This tree can now be used to create valid input sequences following the branches. The list of the valid input message sequences generated is given below:

Starting State: IDLE:

1. Open,
2. Open, Setup_Ack,
3. Open, Failure,
4. Open, Timeout(SA_Timer),
5. Open, Close,
6. Open, Setup_Ack, Close,

7. Open, Setup_Ack, Failure,
8. Open, Setup_Ack, Connect,
9. Open, Setup_Ack, Timeout(Setup_Timer),
10. Open, Setup_Ack, Connect, Close,
11. Open, Setup_Ack, Connect, Failure,
12. Open, Setup_Ack, Connect, Timeout(Setup_Timer),
13. Open, Setup_Ack, Timeout(Setup_Timer), Connect,
14. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer),
15. Open, Setup_Ack, Timeout(Setup_Timer), Close(!Conn),
16. Open, Setup_Ack, Timeout(Setup_Timer), Failure,
17. Open, Setup_Ack, Timeout(Setup_Timer), Close(Conn),
18. Open, Setup_Ack, Connect, Timeout(Setup_Timer), Timeout(KA_Timer),
19. Open, Setup_Ack, Connect, Timeout(Setup_Timer), Close,
20. Open, Setup_Ack, Connect, Timeout(Setup_Timer), Failure,
21. Open, Setup_Ack, Timeout(Setup_Timer), Connect, Close,
22. Open, Setup_Ack, Timeout(Setup_Timer), Connect, Failure,
23. Open, Setup_Ack, Timeout(Setup_Timer), Connect, Timeout(KA_Timer),
24. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Connect,
25. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N),
26. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Close(Conn),
27. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Close(!Conn),
28. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Failure,
29. Open, Setup_Ack, Timeout(Setup_Timer), Close(Conn),Connect,
30. Open, Setup_Ack, Timeout(Setup_Timer), Close(Conn), Failure,
31. Open, Setup_Ack, Timeout(Setup_Timer), Close(Conn), Timeout(Conn_Timer),
32. Open, Setup_Ack, Connect, Timeout(Setup_Timer), Timeout(KA_Timer)(N),

33. Open, Setup_Ack, Connect, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Close,

34. Open, Setup_Ack, Connect, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Failure,

35. Open, Setup_Ack, Timeout(Setup_Timer), Connect, Timeout(KA_Timer)(N),

36. Open, Setup_Ack, Timeout(Setup_Timer), Connect, Timeout(KA_Timer)(N), Close,

37. Open, Setup_Ack, Timeout(Setup_Timer), Connect, Timeout(KA_Timer)(N), Failure,

38. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Connect, Close,

39. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Connect, Failure,

40. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Connect, Timeout(KA_Timer)(N),

41. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Close(Conn), Connect,

42. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Close(Conn), Failure,

43. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Close(Conn), Timeout(Conn_Timer),

44. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Connect, Timeout(KA_Timer)(N), Close,

45. Open, Setup_Ack, Timeout(Setup_Timer), Timeout(KA_Timer)(N), Connect, Timeout(KA_Timer)(N), Failure,

1.2.2 Reachability Analysis for the Destination

Destination state diagram has three states, and 11 messages. In this subsection, a tree structure is used to show the reachability starting from the initial state IDLE as the root of the tree. The reachability tree is shown in Figure 3.

This tree can now be used to create valid input sequences following the branches. The list of the valid input message sequences generated is given below:

Starting State: IDLE:

1. Setup

2. Setup, Close,

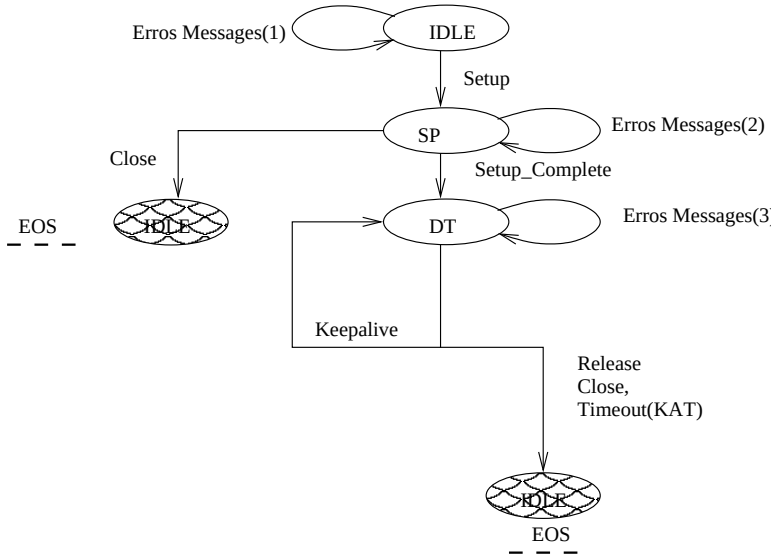

Figure 3: Reachability Tree for the Destination

3. Setup, Setup_Complete,
4. Setup, Setup_Complete, Keepalive(N),
5. Setup, Setup_Complete, Release,
6. Setup, Setup_Complete, Close,
7. Setup, Setup_Complete, Timeout(KA_Timer),
8. Setup, Setup_Complete, Keepalive(N), Release,
9. Setup, Setup_Complete, Keepalive(N), Close,
10. Setup, Setup_Complete, Keepalive(N), Timeout(KA_Timer).

1.2.3 Reachability Analysis for the Switch(FPGA)

Switch state diagram has three states, and 7 messages. In this subsection, a tree structure is used to show the reachability starting from the initial state IDLE as the root of the tree. The reachability tree is shown in Figure 4.

This tree can now be used to create valid input sequences following the branches. The list of the valid input message sequences generated is given below:

Starting State: IDLE:

1. Setup

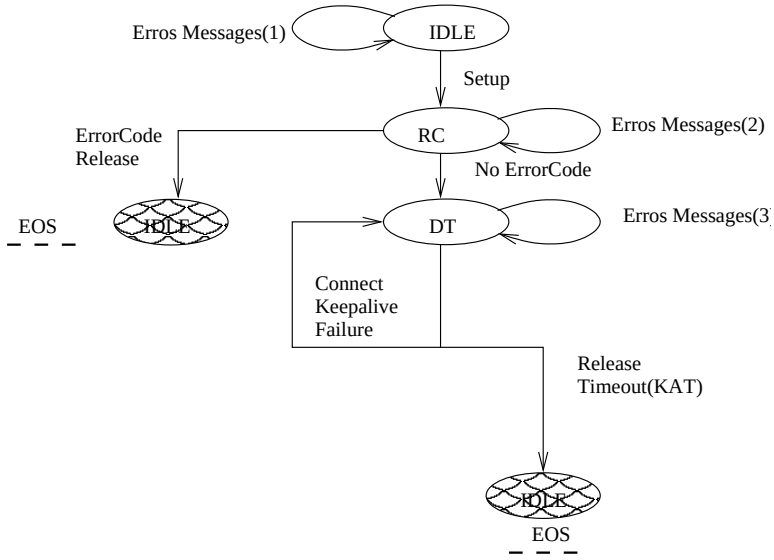

Figure 4: Reachability Tree for the Ingress Switch

2. Setup, (ErrorCode)
3. Setup, Release
4. Setup, (No ErrorCode)
5. Setup, (No ErrorCode), Connect
6. Setup, (No ErrorCode), Keepalive(N)
7. Setup, (No ErrorCode), Failure
8. Setup, (No ErrorCode), Release
9. Setup, (No ErrorCode), Timeout(KA_Timer)
10. Setup, (No ErrorCode), Connect, Keepalive(N)
11. Setup, (No ErrorCode), Connect, Failure
12. Setup, (No ErrorCode), Connect, Release
13. Setup, (No ErrorCode), Connect, Timeout(KA_Timer)
14. Setup, (No ErrorCode), Keepalive(N), Connect
15. Setup, (No ErrorCode), Keepalive(N), Failure
16. Setup, (No ErrorCode), Keepalive(N), Release

17. Setup, (No ErrorCode), Keepalive(N), Timeout(KA_Timer)
18. Setup, (No ErrorCode), Failure, Connect
19. Setup, (No ErrorCode), Failure, Keepalive(N)
20. Setup, (No ErrorCode), Failure, Release
21. Setup, (No ErrorCode), Failure, Timeout(KA_Timer)
22. Setup, (No ErrorCode), Connect, Keepalive(N), Failure
23. Setup, (No ErrorCode), Connect, Keepalive(N), Release
24. Setup, (No ErrorCode), Connect, Keepalive(N), Timeout(KA_Timer)
25. Setup, (No ErrorCode), Connect, Failure, Keepalive(N)
26. Setup, (No ErrorCode), Connect, Failure, Release
27. Setup, (No ErrorCode), Connect, Failure, Timeout(KA_Timer)
28. Setup, (No ErrorCode), Keepalive(N), Connect, Failure
29. Setup, (No ErrorCode), Keepalive(N), Connect, Keepalive(N)
30. Setup, (No ErrorCode), Keepalive(N), Connect, Release
31. Setup, (No ErrorCode), Keepalive(N), Connect, Timeout(KA_Timer)
32. Setup, (No ErrorCode), Keepalive(N), Failure, Connect
33. Setup, (No ErrorCode), Keepalive(N), Failure, Keepalive(N)
34. Setup, (No ErrorCode), Keepalive(N), Failure, Release
35. Setup, (No ErrorCode), Keepalive(N), Failure, Timeout(KA_Timer)
36. Setup, (No ErrorCode), Failure, Connect, Keepalive(N)
37. Setup, (No ErrorCode), Failure, Connect, Release
38. Setup, (No ErrorCode), Failure, Connect, Timeout(KA_Timer)
39. Setup, (No ErrorCode), Failure, Keepalive(N), Connect
40. Setup, (No ErrorCode), Failure, Keepalive(N), Release
41. Setup, (No ErrorCode), Failure, Keepalive(N), Timeout(KA_Timer)
42. Setup, (No ErrorCode), Connect, Keepalive(N), Failure, Keepalive(N)
43. Setup, (No ErrorCode), Connect, Keepalive(N), Failure, Release
44. Setup, (No ErrorCode), Connect, Keepalive(N), Failure, Timeout(KA_Timer)

45. Setup, (No ErrorCode), Connect, Failure, Keepalive(N), Release
46. Setup, (No ErrorCode), Connect, Failure, Keepalive(N), Timeout(KA_Timer)
47. Setup, (No ErrorCode), Keepalive(N), Connect, Failure, Keepalive(N)
48. Setup, (No ErrorCode), Keepalive(N), Connect, Failure, Release
49. Setup, (No ErrorCode), Keepalive(N), Connect, Failure, Timeout(KA_Timer)
50. Setup, (No ErrorCode), Keepalive(N), Connect, Keepalive(N), Failure
51. Setup, (No ErrorCode), Keepalive(N), Connect, Keepalive(N), Release
52. Setup, (No ErrorCode), Keepalive(N), Connect, Keepalive(N), Timeout(KA_Timer)
53. Setup, (No ErrorCode), Keepalive(N), Failure, Connect, Keepalive(N)
54. Setup, (No ErrorCode), Keepalive(N), Failure, Connect, Release
55. Setup, (No ErrorCode), Keepalive(N), Failure, Connect, Timeout(KA_Timer)
56. Setup, (No ErrorCode), Keepalive(N), Failure, Keepalive(N), Connect
57. Setup, (No ErrorCode), Keepalive(N), Failure, Keepalive(N), Release
58. Setup, (No ErrorCode), Keepalive(N), Failure, Keepalive(N), Timeout(KA_Timer)
59. Setup, (No ErrorCode), Failure, Connect, Keepalive(N), Release
60. Setup, (No ErrorCode), Failure, Connect, Keepalive(N), Timeout(KA_Timer)
61. Setup, (No ErrorCode), Failure, Keepalive(N), Connect, Keepalive(N)
62. Setup, (No ErrorCode), Failure, Keepalive(N), Connect, Release
63. Setup, (No ErrorCode), Failure, Keepalive(N), Connect, Timeout(KA_Timer)
64. Setup, (No ErrorCode), Connect, Keepalive(N), Failure, Keepalive(N), Release
65. Setup, (No ErrorCode), Connect, Keepalive(N), Failure, Keepalive(N), Timeout(KA_Timer)
66. Setup, (No ErrorCode), Keepalive(N), Connect, Failure, Keepalive(N), Release
67. Setup, (No ErrorCode), Keepalive(N), Connect, Failure, Keepalive(N), Timeout(KA_Timer)
68. Setup, (No ErrorCode), Keepalive(N), Connect, Keepalive(N), Failure, Release

69. Setup, (No ErrorCode), Keepalive(N), Connect, Keepalive(N), Failure, Timeout(KA_Timer)

70. Setup, (No ErrorCode), Keepalive(N), Connect, Keepalive(N), Failure, Keepalive(N)

71. Setup, (No ErrorCode), Keepalive(N), Failure, Connect, Keepalive(N), Release

72. Setup, (No ErrorCode), Keepalive(N), Failure, Connect, Keepalive(N), Timeout(KA_Timer)

73. Setup, (No ErrorCode), Keepalive(N), Failure, Keepalive(N), Connect, Release

74. Setup, (No ErrorCode), Keepalive(N), Failure, Keepalive(N), Connect, Timeout(KA_Timer)

75. Setup, (No ErrorCode), Keepalive(N), Failure, Keepalive(N), Connect, Keepalive(N)

76. Setup, (No ErrorCode), Failure, Keepalive(N), Connect, Keepalive(N), Release

77. Setup, (No ErrorCode), Failure, Keepalive(N), Connect, Keepalive(N), Timeout(KA_Timer)

78. Setup, (No ErrorCode), Keepalive(N), Connect, Keepalive(N), Failure, Keepalive(N), Release

79. Setup, (No ErrorCode), Keepalive(N), Connect, Keepalive(N), Failure, Keepalive(N), Timeout(KA_Timer)

80. Setup, (No ErrorCode), Keepalive(N), Failure, Keepalive(N), Connect, Keepalive(N), Release

81. Setup, (No ErrorCode), Keepalive(N), Failure, Keepalive(N), Connect, Keepalive(N), Timeout(KA_Timer)

1.2.4 Reachability Analysis for the Switch(JITPAC)

In this subsection, the switch is treated as a whole and the input sequences coming to that node is generated. This set of sequences is a subset of the sequences generated in 1.2.3.

Starting State: IDLE:

1. Setup

2. Setup, Release

3. Setup, Connect

4. Setup, Keepalive(N)
5. Setup, Failure
6. Setup, Connect, Keepalive(N)
7. Setup, Connect, Failure
8. Setup, Connect, Release
9. Setup, Keepalive(N), Connect
10. Setup, Keepalive(N), Failure
11. Setup, Keepalive(N), Release
12. Setup, Failure, Connect
13. Setup, Failure, Keepalive(N)
14. Setup, Failure, Release
15. Setup, Connect, Keepalive(N), Failure
16. Setup, Connect, Keepalive(N), Release
17. Setup, Connect, Failure, Keepalive(N)
18. Setup, Connect, Failure, Release
19. Setup, Keepalive(N), Connect, Failure
20. Setup, Keepalive(N), Connect, Keepalive(N)
21. Setup, Keepalive(N), Connect, Release
22. Setup, Keepalive(N), Failure, Connect
23. Setup, Keepalive(N), Failure, Keepalive(N)
24. Setup, Keepalive(N), Failure, Release
25. Setup, Failure, Connect, Keepalive(N)
26. Setup, Failure, Connect, Release
27. Setup, Failure, Keepalive(N), Connect
28. Setup, Failure, Keepalive(N), Release
29. Setup, Connect, Keepalive(N), Failure, Keepalive(N)
30. Setup, Connect, Keepalive(N), Failure, Release
31. Setup, Connect, Failure, Keepalive(N), Release

32. Setup, Keepalive(N), Connect, Failure, Keepalive(N)
33. Setup, Keepalive(N), Connect, Failure, Release
34. Setup, Keepalive(N), Connect, Keepalive(N), Failure
35. Setup, Keepalive(N), Connect, Keepalive(N), Release
36. Setup, Keepalive(N), Failure, Connect, Keepalive(N)
37. Setup, Keepalive(N), Failure, Connect, Release
38. Setup, Keepalive(N), Failure, Keepalive(N), Connect
39. Setup, Keepalive(N), Failure, Keepalive(N), Release
40. Setup, Failure, Connect, Keepalive(N), Release
41. Setup, Failure, Keepalive(N), Connect, Keepalive(N)
42. Setup, Failure, Keepalive(N), Connect, Release
43. Setup, Connect, Keepalive(N), Failure, Keepalive(N), Release
44. Setup, Keepalive(N), Connect, Failure, Keepalive(N), Release
45. Setup, Keepalive(N), Connect, Keepalive(N), Failure, Release
46. Setup, Keepalive(N), Connect, Keepalive(N), Failure, Keepalive(N)
47. Setup, Keepalive(N), Failure, Connect, Keepalive(N), Release
48. Setup, Keepalive(N), Failure, Keepalive(N), Connect, Release
49. Setup, Keepalive(N), Failure, Keepalive(N), Connect, Keepalive(N)
50. Setup, Failure, Keepalive(N), Connect, Keepalive(N), Release
51. Setup, Keepalive(N), Connect, Keepalive(N), Failure, Keepalive(N), Release
52. Setup, Keepalive(N), Failure, Keepalive(N), Connect, Keepalive(N), Release

1.3 Conformance Testing

Testing is carried out by using test sequences. A test sequence is a list of inputs and expected outputs. The input sequences are generated in the previous section. In this section, the expected output sequences for each input sequences defined will be generated.

The conformance test for the just-in-time protocol can be performed as illustrated in Figure 5. There will be two tester units, Peer Layer Tester (PLT) and Upper Layer Tester (ULT) to send input messages generated in Section

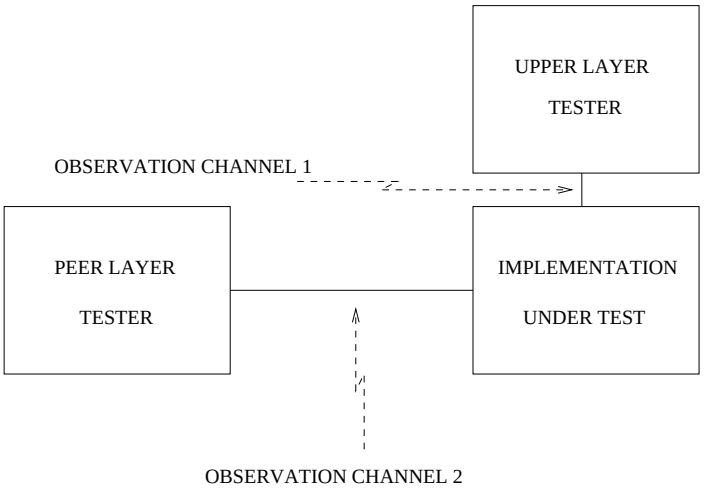

Figure 5: Protocol Testing Architecture

1.2. The reactions of Implementation Under Test (IUT) to these messages or in other words, the valid output messages will be observed in Observation Channel 1 (OC1) and Observation Channel 2 (OC2). In order to make the testing easier, during the conformance test, the valid output sequences will also include the channel in which each message is expected. The rest of this chapter is devoted to valid output sequences generated for each element mentioned in Section 1.2.

1.3.1 Valid Output Sequences for the Source Client

1. Setup(OC2)
2. Setup(OC2)
3. Setup(OC2), Connection_Failure(OC1) and Release(OC2)
4. Setup(OC2), Connection_Failure(OC1) and Release(OC2)
5. Setup(OC2), Release(OC2)
6. Setup(OC2), Release(OC2)
7. Setup(OC2), Connection_Failure(OC1) and Release(OC2)
8. Setup(OC2), Conn_Rcvd = TRUE
9. Setup(OC2), Clear_To_Send(OC1)
10. Setup(OC2), Conn_Rcvd = TRUE, Release(OC2)
11. Setup(OC2), Conn_Rcvd = TRUE, Connection_Failure(OC1) and Release(OC2)

12. Setup(OC2), Conn_Rcvd = TRUE, Clear_To_Send(OC1)
13. Setup(OC2), Clear_To_Send(OC1), Conn_Rcvd = TRUE
14. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)
15. Setup(OC2), Clear_To_Send(OC1), Release(OC2)
16. Setup(OC2), Clear_To_Send(OC1), Connection_Failure(OC1) and Release(OC2)
17. Setup(OC2), Clear_To_Send(OC1), Release(OC2)
18. Setup(OC2), Conn_Rcvd = TRUE, Clear_To_Send(OC1), Keepalive(OC2)
19. Setup(OC2), Conn_Rcvd = TRUE, Clear_To_Send(OC1), Release(OC2)
20. Setup(OC2), Conn_Rcvd = TRUE, Clear_To_Send(OC1), Connection_Failure(OC1) and Release(OC2)
21. Setup(OC2), Clear_To_Send(OC1), Conn_Rcvd = TRUE, Release(OC2)
22. Setup(OC2), Clear_To_Send(OC1), Conn_Rcvd = TRUE, Connection_Failure(OC1) and Release(OC2)
23. Setup(OC2), Clear_To_Send(OC1), Conn_Rcvd = TRUE, Keepalive(OC2)
24. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Conn_Rcvd = TRUE
25. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N)
26. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Release(OC2)
27. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Release(OC2)
28. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Connection_Failure(OC1) and Release(OC2)
29. Setup(OC2), Clear_To_Send(OC1), Release(OC2), Transmission_Complete(OC1)
30. Setup(OC2), Clear_To_Send(OC1), Release(OC2), Connection_Failure(OC1)
31. Setup(OC2), Clear_To_Send(OC1), Release(OC2), Connection_Failure(OC1)
32. Setup(OC2), Conn_Rcvd = TRUE, Clear_To_Send(OC1), Keepalive(OC2)(N)
33. Setup(OC2), Conn_Rcvd = TRUE, Clear_To_Send(OC1), Keepalive(OC2)(N), Release(OC2)
34. Setup(OC2), Conn_Rcvd = TRUE, Clear_To_Send(OC1), Keepalive(OC2)(N), Connection_Failure(OC1) and Release(OC2)
35. Setup(OC2), Clear_To_Send(OC1), Conn_Rcvd = TRUE, Keepalive(OC2)(N)

36. Setup(OC2), Clear_To_Send(OC1), Conn_Rcvd = TRUE, Keepalive(OC2)(N), Release(OC2)
37. Setup(OC2), Clear_To_Send(OC1), Conn_Rcvd = TRUE, Keepalive(OC2)(N), Connection_Failure(OC1) and Release(OC2)
38. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Conn_Rcvd = TRUE, Release(OC2)
39. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Conn_Rcvd = TRUE, Connection_Failure(OC1) and Release(OC2)
40. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Conn_Rcvd = TRUE, Keepalive(OC2)(N)
41. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Release(OC2), Transmission_Complete(OC1)
42. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Release(OC2), Connection_Failure(OC1)
43. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Release(OC2), Connection_Failure(OC1)
44. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Conn_Rcvd = TRUE, Keepalive(OC2)(N), Release(OC2)
45. Setup(OC2), Clear_To_Send(OC1), Keepalive(OC2)(N), Conn_Rcvd = TRUE, Keepalive(OC2)(N), Connection_Failure(OC1) and Release(OC2)

1.3.2 Valid Output Sequences for the Destination Client

1. Open(OC1)
2. Open(OC1), Failure(OC2)
3. Open(OC1), Connect(OC2) if Conn=TRUE
4. Open(OC1), Connect(OC2) if Conn=TRUE
5. Open(OC1), Connect(OC2) if Conn=TRUE, Transmission_Complete(OC1)
6. Open(OC1), Connect(OC2) if Conn=TRUE, Failure(OC2)
7. Open(OC1), Connect(OC2) if Conn=TRUE, Transmission_Failure(OC1)
8. Open(OC1), Connect(OC2) if Conn=TRUE, Transmission_Complete(OC1)
9. Open(OC1), Connect(OC2) if Conn=TRUE, Failure(OC2)
10. Open(OC1), Connect(OC2) if Conn=TRUE, Transmission_Failure(OC1)

1.3.3 Valid Output Sequences for the Switches(FPGA)

The only difference between an ingress and intermediate switch from the point of output sequences is the Setup_Ack message. Intermediate switch does not generate Setup_Ack message so if we remove Setup_Ack messages from the output sequences created below, we can obtain the output sequences for the intermediate switch.

1. -
2. Failure(OC2)
3. Release(OC2)
4. Setup(OC2) and Setup_Ack
5. Setup(OC2) and Setup_Ack, Connect(OC2)
6. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N)
7. Setup(OC2) and Setup_Ack, Failure(OC2)
8. Setup(OC2) and Setup_Ack, Release(OC2)
9. Setup(OC2) and Setup_Ack
10. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N)
11. Setup(OC2) and Setup_Ack, Connect(OC2), Failure(OC2)
12. Setup(OC2) and Setup_Ack, Connect(OC2), Release(OC2)
13. Setup(OC2) and Setup_Ack, Connect(OC2)
14. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2)
15. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2)
16. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Release(OC2)
17. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N)
18. Setup(OC2) and Setup_Ack, Failure(OC2), Connect(OC2)
19. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N)
20. Setup(OC2) and Setup_Ack, Failure(OC2), Release(OC2)
21. Setup(OC2) and Setup_Ack, Failure(OC2)
22. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Failure(OC2)
23. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Release(OC2)

24. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N)
25. Setup(OC2) and Setup_Ack, Connect(OC2), Failure(OC2), Keepalive(OC2)(N)
26. Setup(OC2) and Setup_Ack, Connect(OC2), Failure(OC2), Release(OC2)
27. Setup(OC2) and Setup_Ack, Connect(OC2), Failure(OC2)
28. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Failure(OC2)
29. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N)
30. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Release(OC2)
31. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2)
32. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Connect(OC2)
33. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N)
34. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Release(OC2)
35. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2)
36. Setup(OC2) and Setup_Ack, Failure(OC2), Connect(OC2), Keepalive(OC2)(N)
37. Setup(OC2) and Setup_Ack, Failure(OC2), Connect(OC2), Release(OC2)
38. Setup(OC2) and Setup_Ack, Failure(OC2), Connect(OC2)
39. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Connect(OC2)
40. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Release(OC2)
41. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N)
42. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N)
43. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Release(OC2)
44. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Failure(OC2)
45. Setup(OC2) and Setup_Ack, Connect(OC2), Failure(OC2), Keepalive(OC2)(N), Release(OC2)
46. Setup(OC2) and Setup_Ack, Connect(OC2), Failure(OC2), Keepalive(OC2)(N)
47. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Failure(OC2), Keepalive(OC2)(N)

48. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Failure(OC2), Release(OC2)

49. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Failure(OC2)

50. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Failure(OC2)

51. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Release(OC2)

52. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N)

53. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Connect(OC2), Keepalive(OC2)(N)

54. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Connect(OC2), Release(OC2)

55. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Connect(OC2)

56. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Connect(OC2)

57. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Release(OC2)

58. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N)

59. Setup(OC2) and Setup_Ack, Failure(OC2), Connect(OC2), Keepalive(OC2)(N), Release(OC2)

60. Setup(OC2) and Setup_Ack, Failure(OC2), Connect(OC2), Keepalive(OC2)(N)

61. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N)

62. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Release(OC2)

63. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Connect(OC2)

64. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Release(OC2)

65. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N)

66. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Failure(OC2), Keepalive(OC2)(N), Release(OC2)

67. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Failure(OC2), Keepalive(OC2)(N)

68. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Release(OC2)

69. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Failure(OC2)

70. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N)

71. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Connect(OC2), Keepalive(OC2)(N), Release(OC2)

72. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Connect(OC2), Keepalive(OC2)(N)

73. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Release(OC2)

74. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Connect(OC2)

75. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N)

76. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Release(OC2)

77. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N)

78. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Release(OC2)

79. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N)

80. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Release(OC2)

81. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N)

1.3.4 Valid Output Sequences for the Switches(JITPAC)

The only difference between an ingress and intermediate switch from the point of output sequences is the Setup_Ack message. Intermediate switch does not generate Setup_Ack message so if we remove Setup_Ack messages from the output sequences created below, we can obtain the output sequences for the intermediate switch.

1. - OR Failure(OC2)
2. Release(OC2)
3. Setup(OC2) and Setup_Ack, Connect(OC2)
4. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N)
5. Setup(OC2) and Setup_Ack, Failure(OC2)
6. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N)
7. Setup(OC2) and Setup_Ack, Connect(OC2), Failure(OC2)
8. Setup(OC2) and Setup_Ack, Connect(OC2), Release(OC2)
9. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2)
10. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2)
11. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Release(OC2)
12. Setup(OC2) and Setup_Ack, Failure(OC2), Connect(OC2)
13. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N)
14. Setup(OC2) and Setup_Ack, Failure(OC2), Release(OC2)
15. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Failure(OC2)
16. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Release(OC2)
17. Setup(OC2) and Setup_Ack, Connect(OC2), Failure(OC2), Keepalive(OC2)(N)
18. Setup(OC2) and Setup_Ack, Connect(OC2), Failure(OC2), Release(OC2)
19. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Failure(OC2)
20. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N)
21. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Release(OC2)
22. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Connect(OC2)
23. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N)
24. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Release(OC2)
25. Setup(OC2) and Setup_Ack, Failure(OC2), Connect(OC2), Keepalive(OC2)(N)
26. Setup(OC2) and Setup_Ack, Failure(OC2), Connect(OC2), Release(OC2)
27. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Connect(OC2)

28. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Release(OC2)

29. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N)

30. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Release(OC2)

31. Setup(OC2) and Setup_Ack, Connect(OC2), Failure(OC2), Keepalive(OC2)(N), Release(OC2)

32. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Failure(OC2), Keepalive(OC2)(N)

33. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Failure(OC2), Release(OC2)

34. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Failure(OC2)

35. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Release(OC2)

36. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Connect(OC2), Keepalive(OC2)(N)

37. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Connect(OC2), Release(OC2)

38. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Connect(OC2)

39. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Release(OC2)

40. Setup(OC2) and Setup_Ack, Failure(OC2), Connect(OC2), Keepalive(OC2)(N), Release(OC2)

41. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N)

42. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Release(OC2)

43. Setup(OC2) and Setup_Ack, Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Release(OC2)

44. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Failure(OC2), Keepalive(OC2)(N), Release(OC2)

45. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Release(OC2)

46. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N)

47. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Connect(OC2), Keepalive(OC2)(N), Release(OC2)

48. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Release(OC2)

49. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N)

50. Setup(OC2) and Setup_Ack, Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Release(OC2)

51. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Release(OC2)

52. Setup(OC2) and Setup_Ack, Keepalive(OC2)(N), Failure(OC2), Keepalive(OC2)(N), Connect(OC2), Keepalive(OC2)(N), Release(OC2)

1.3.5 Unique Input/Output (UIO) Sequences

Lai[1] defines a unique input/output (UIO) sequence as "A UIO sequence for a state is an input/output behaviour that is not exhibited by any other state". The protocol testing with UIO is performed by feeding the system with a specified input sequence and observe the output sequence. If the output sequence is the same with any UIO sequence defined, then this information gives us the initial state of the state machine.

UIO sequences and distinguishing sequences are not the same. In a distinguishing sequence, the input sequence is the same for all states. The states can be distinguished by the output sequence generated for the same input sequences coming to each state. However, in an UIO sequence, usually, each state has a different input sequence. For a given state s, the input part is applied to the state machine and the output is observed. If it is same with the expected output sequence, then it can be concluded that the state machine is in state s. In Tables 1-3, for each state machine (source, destination and switch), a UIO sequence is generated, respectively, where a λ denotes a null output.

1.4 Implementation Assessment

Implementation assessment mostly consists of generating test scenarios to assess the performance of the implementation. For our system under implementation, the system structure is shown in Figure 6.

[1] R. Lai, A Survey of Communication Protocol Testing, The Journal of Systems and Software, 2001.

Table 1: UIO Sequence for the Source

State	UIO Sequence
IDLE	Open/Setup
WAIT_FOR_SETUP_ACK	Timeout(SA_Timer)/Release
SETUP_PROCEEDING	Timeout(Setup_Timer)/Clear_To_Send
DATA_TRANSMISSION	Timeout(KA_Timer)/Keepalive
WAIT_FOR_CONNECT	Timeout(Conn_Timer)/Connection_Failure

Table 2: UIO Sequence for the Destination

State	UIO Sequence
IDLE	Setup/Open
SETUP_PROCEEDING	Setup_Complete/λ
DATA_TRANSMISSION	Timeout(KA_Timer)/Transmission_Failure

Table 3: UIO Sequence for the Switch

State	UIO Sequence
IDLE	Setup/λ
RUNNING_CHECKS	Release/Failure
DATA_TRANSMISSION	Release/Release

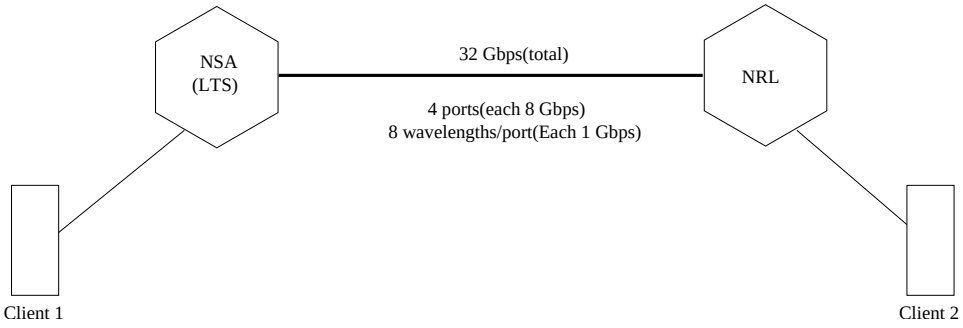

Figure 6: Testbed Architecture

As shown in Figure 6, our initial implementation consists of two end nodes, Client 1 and Client 2, and two OBS switches using Just-In-Time Protocol Accelarator Cards (JITPACs). Although each switch has four ports, only one port will be tested in this first implementation. Each port can carry up to eight wavelengths. Once the system architecture is given, we should mention the necessary input parameters and the output parameters that will be observed as the outcome of these tests.

This system is a prototype system, and as there are only two nodes, it can be used mainly for obtaining performance parameters. The parameters that may be of interest are Mean Burst Size, Mean Idle Time, System Throughput, Mean System Delay, and System Utilization.

In order to obtain throughput and utilization, the related formulation is given below:

$$Throughput = \frac{Number\_of\_Burst\_Sent}{Test\_Time} \tag{1}$$

$$Utilization = \frac{Total\_Burst\_Time}{Test\_Time} \tag{2}$$

Therefore, we need to use a counter to count the number of bursts sent and two variables to keep the timing values for burst times and test duration. These variables can also be used to calculate Mean Burst Size and Mean Idle Time according to the formulation given below:

$$Mean\_Burst\_Size = \frac{Total\_Burst\_Size}{Number\_of\_Burst\_Sent} \tag{3}$$

$$Mean\_Idle\_Time = \frac{Test\_Time - Total\_Burst\_Time}{Test\_Time} \tag{4}$$

In order to obtain the last parameter, System Delay, we need three more timing variables. The first one will be used to keep tract of the time at the source node when the Setup message is sent to the Ingress switch. The second one keeps the time value at the destination when the connection is closed due to a Release message received or a forced release by a timeout message. The last one is used to store the difference of these two variables to calculate total time spend for transmission. Once we computed that total time, the mean system delay is given

$$Mean\_System\_Delay = \frac{Total\_System\_Delay}{Number\_of\_Burst\_Sent} \tag{5}$$

Once these parameters are obtained, they can be used as inputs to analytical analysis.