Chapter 4

Implementation of Outer Receiver

Depending on chapter 3 we have described, this chapter will discuss the implementation of outer receiver. The implementation result, including chip size, core size, pin assignment, and dynamic power consumption will be analyzed in this chapter.

4.1 Demapping Module

The metric generation function of demapping module consists of a constant-multiplier. A constant-multiplier can be decomposed into several individual adders applying shift-bit operation. The scheme is called shift-adder method. For example, Figure 29 shows the example of constant-multiplier with constant number 7.

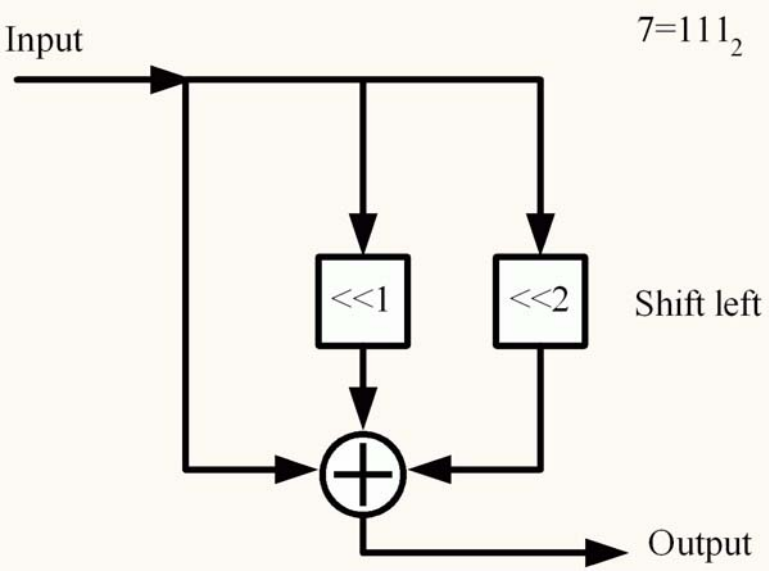

Figure 29: Constant-multiplier with constant number 7

4.2 Deinterleaver Module

From the discussion about deinterleaver, the deinterleaver main architecture can be built of double buffers. The double buffers can be registers or SRAM. For design convenience, we choose the registers for the double buffers. Besides, we take the deinterleaver as the main control unit for outer receiver. IEEE 802.11a owns eight different data rates, that is, the outer receiver has to support the throughput of eight different data rates. Each data rate case owns self decoding mechanism and timing procedure. The deinterleaver is the most characteristic module of data rate. That is why we choose the deinterleaver to be a control unit.

Every OFDM symbol interval, the equalizer gives 48 symbols to the demapping module. From Table 3, we know the data bits number per OFDM symbol. We adopt 3

stages radix-2 ACS architecture, and we can derive Table 6 as follows. Table 6 indicates the enable interval which the deinterleaver triggers for each data rate case, respectively. As depicted in Figure 30, the clock diagram indicates the following modules enable interval after the deinterleaver module.

Table 6: The enable interval to the following modules

Data rate (Mbits/s)	Modulation	Coding rate	Coded bits Per Subcarrier	Data bits Per OFDM symbol	Enable interval (clock periods)
6	BPSK	1/2	1	24	8
9	BPSK	3/4	1	36	12
12	QPSK	1/2	2	48	16
18	QPSK	3/4	2	72	24
24	16-QAM	1/2	4	96	32
36	16-QAM	3/4	4	144	48
48	64-QAM	2/3	6	192	64
54	64-QAM	3/4	6	216	72

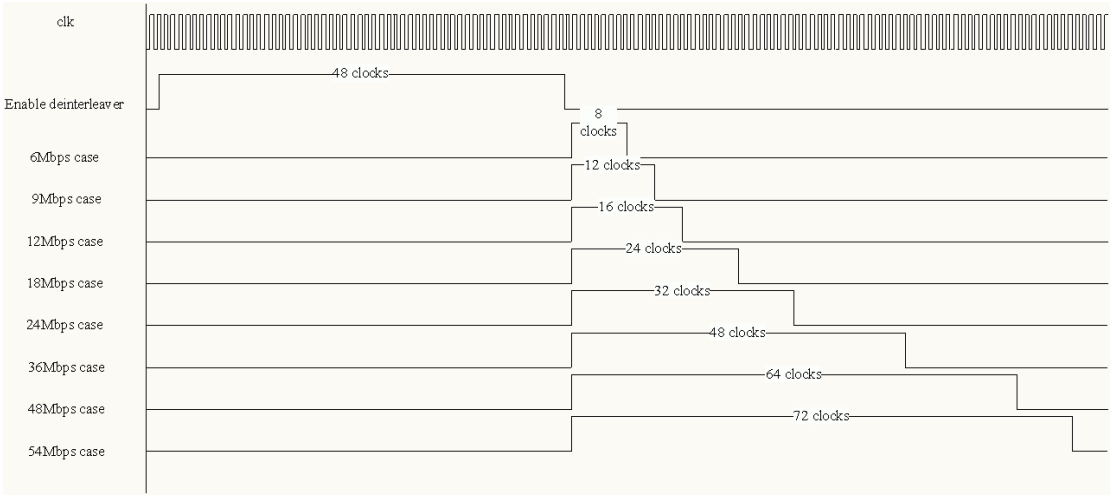

Figure 30: The enable clock diagram of the following modules

4.3 Depuncture Module

In chapter 3, we have discussed that the depuncture task is a control line of setting BMC metric to be zero. But the modules, including deinterleaver, depuncture, and Viterbi decoder, shall follow the rule of puncturing scheme. The following sections in this chapter will discuss the decoding mechanism of each modulation type, respectively. Notice although we use (2, 1, 3) convolutional code to depict these patterns, the same situations also apply for (2, 1, 7) convolutional code in the following discussions. As depicted in Figure 31, Figure 32, and Figure 33, the patterns of three kinds of coding rate are shown, respectively.

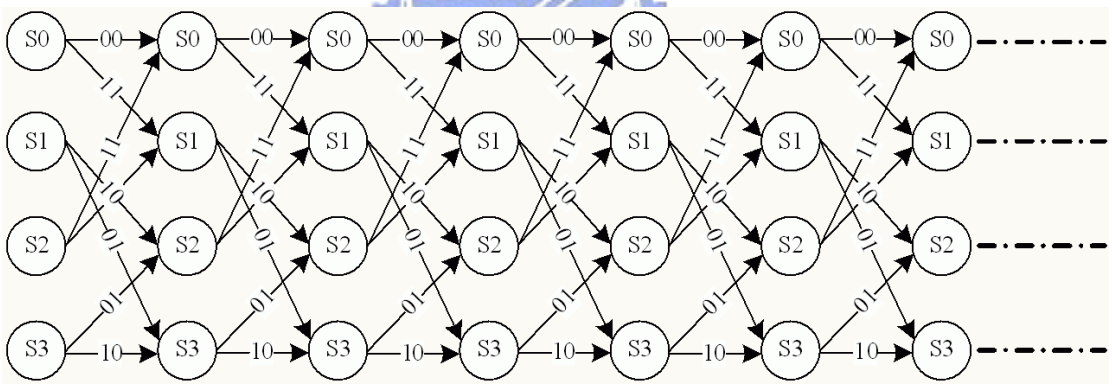

Figure 31: The pattern of coding rate 1/2

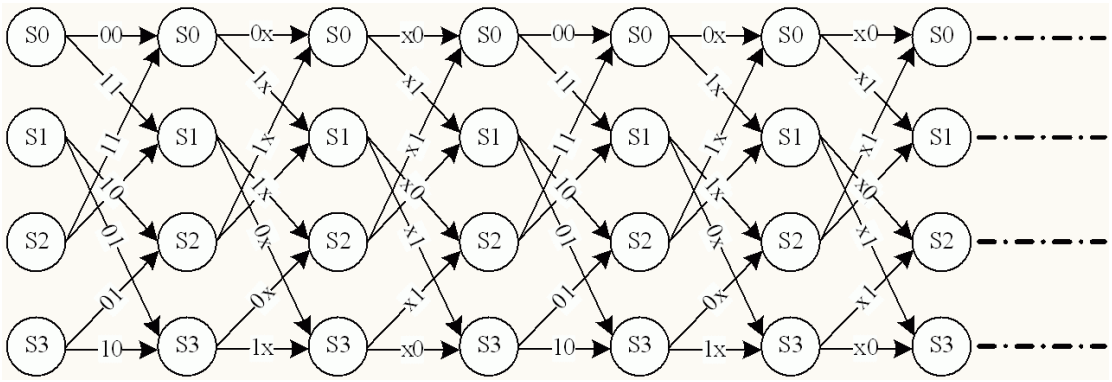

Figure 32: The pattern of coding rate 3/4

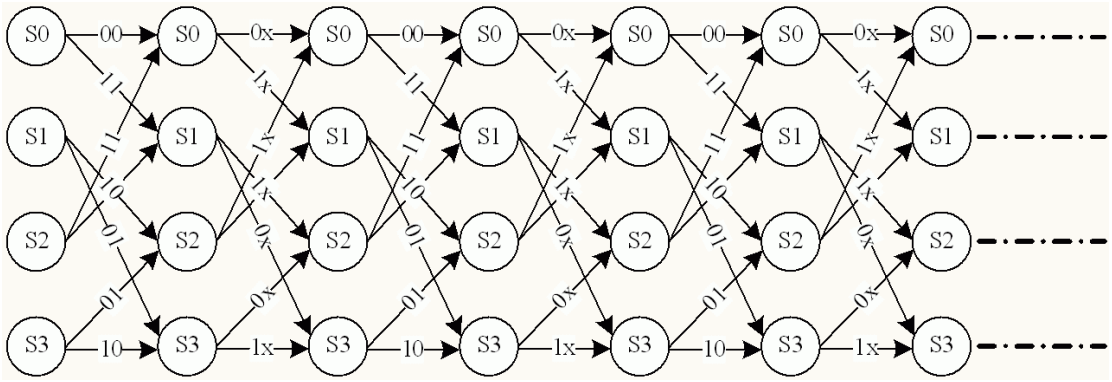

Figure 33: The pattern of coding rate 2/3

4.3.1 Decoding Mechanism of BPSK

The number of coded bits per subcarrier is one in BPSK, that is, there must be two subcarriers to form one decoding stage. For the maximum utilization rate of ACS architecture, there are 6 coded symbols received from the deinterleaver each clock cycle. In BPSK 1/2, there are 48 coded symbols for decoding, and then we take 8 clock cycles to complete the task. It seems that the throughput is 60Mbps when the Viterbi decoder operates decoding task, but actually there are only 8 clock cycles for decoding and other clock cycles of 24 clock cycles for idling.

In BPSK 3/4, the coded symbols suffer the puncturing scheme. The number of coded symbols stolen by puncturing scheme is 2, and the number of coded symbols provided by the deinterleaver is 4. Therefore, there are 4 coded symbols provided by the deinterleaver, and there are 3 bits decoded output each clock cycle. From above discussions, Figure 34 and Figure 35 depict the BPSK 1/2 and BPSK 3/4 decoding mechanism each OFDM symbol.

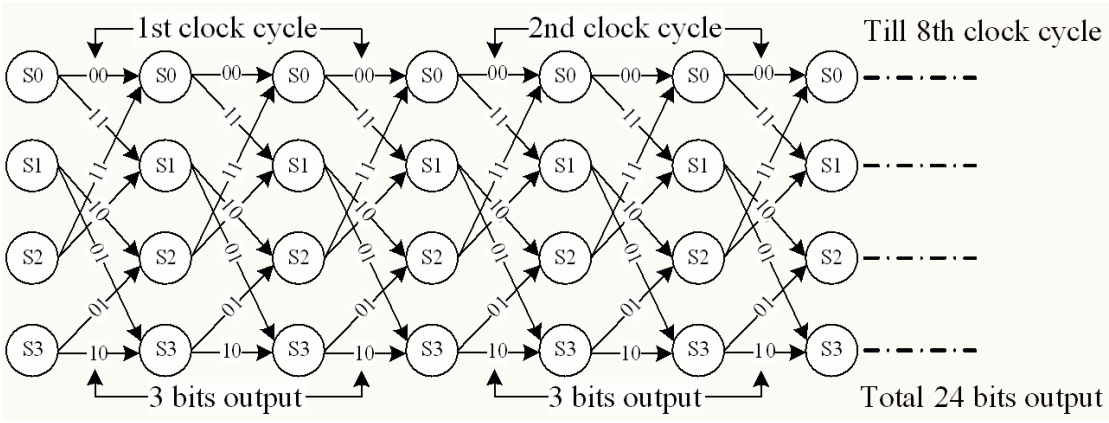

Figure 34: BPSK 1/2 decoding mechanism each OFDM symbol

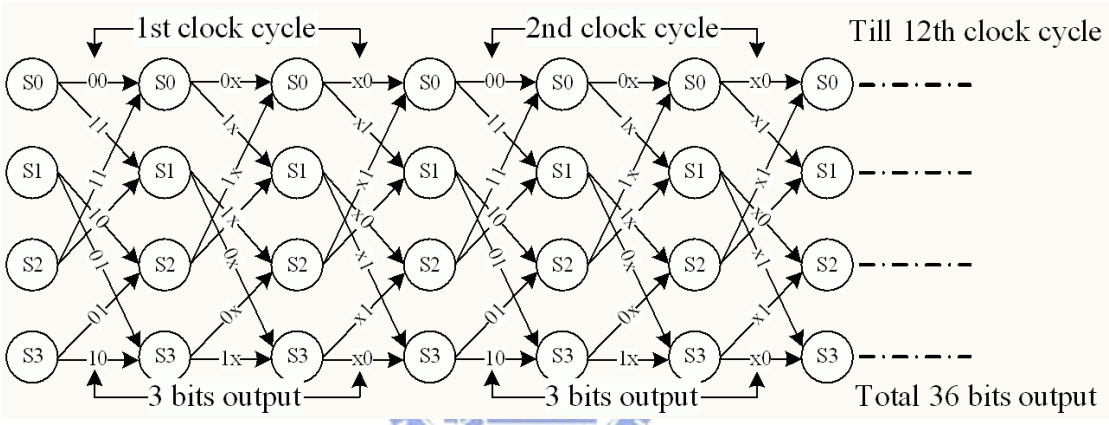

Figure 35: BPSK 3/4 decoding mechanism each OFDM symbol

4.3.2 Decoding Mechanism of QPSK

In QPSK, there are two coded bits per subcarrier. If coding rate 1/2 is applied, the decoding mechanism is the same case as BPSK 1/2. In QPSK 3/4, we also adopt the same decoding mechanism on QPSK 3/4 because the deinterleaver controls the data stream into the depuncture module. The QPSK 1/2 and QPSK 3/4 decoding mechanism per OFDM symbol are described in Figure 36 and Figure 37.

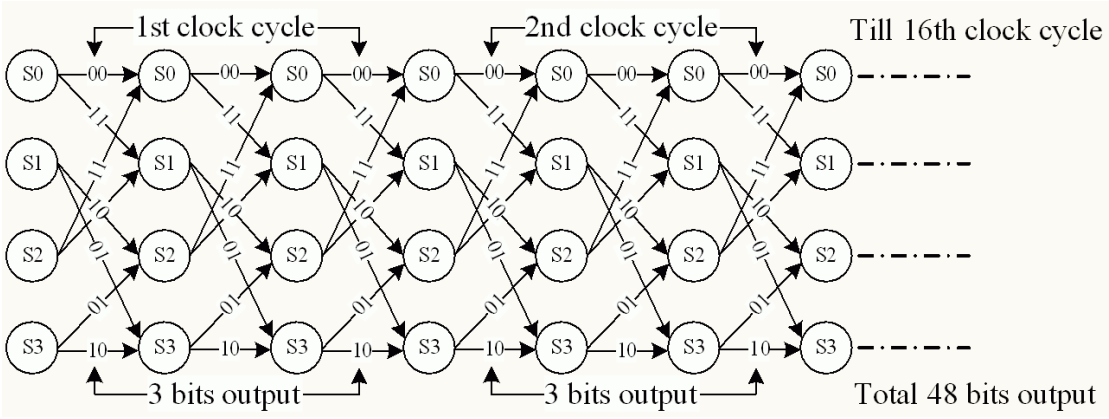

Figure 36: The QPSK 1/2 decoding mechanism each OFDM symbol

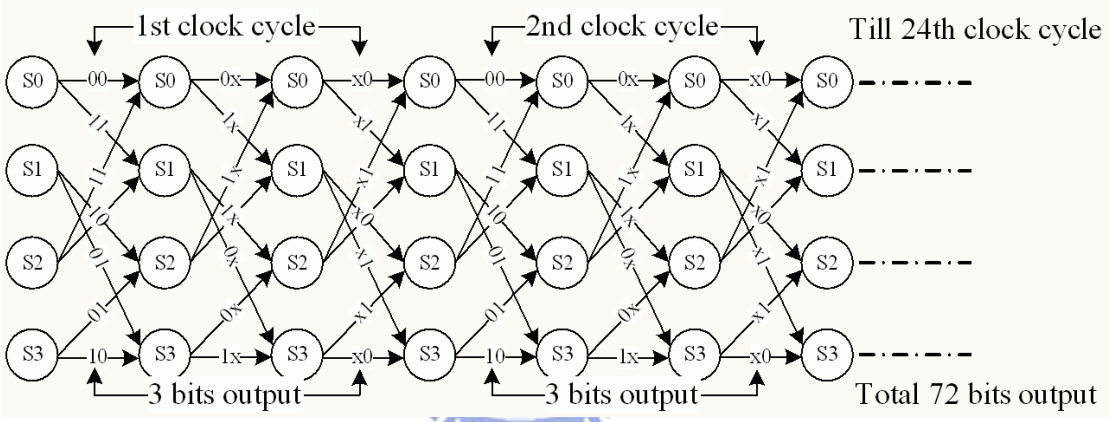

Figure 37: The QPSK 3/4 decoding mechanism each OFDM symbol

4.3.3 Decoding Mechanism of 16-QAM

In 16-QAM, there are four coded bits per subcarrier. However, 16-QAM 1/2 and 16-QAM 3/4 are the same as the above discussion about BPSK and QPSK. In Figure 38 and Figure 39, the decoding mechanism of 16-QAM 1/2 and 16-QAM 3/4 are shown.

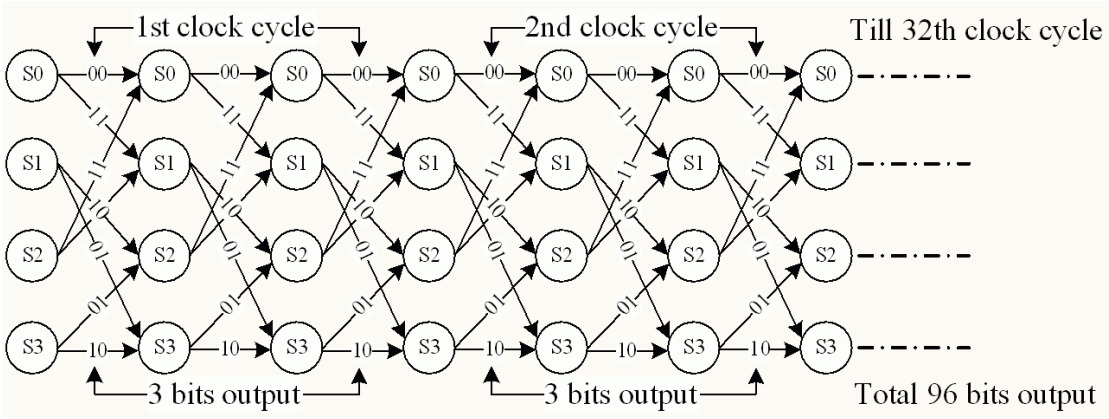

Figure 38: The 16-QAM 1/2 decoding mechanism each OFDM symbol

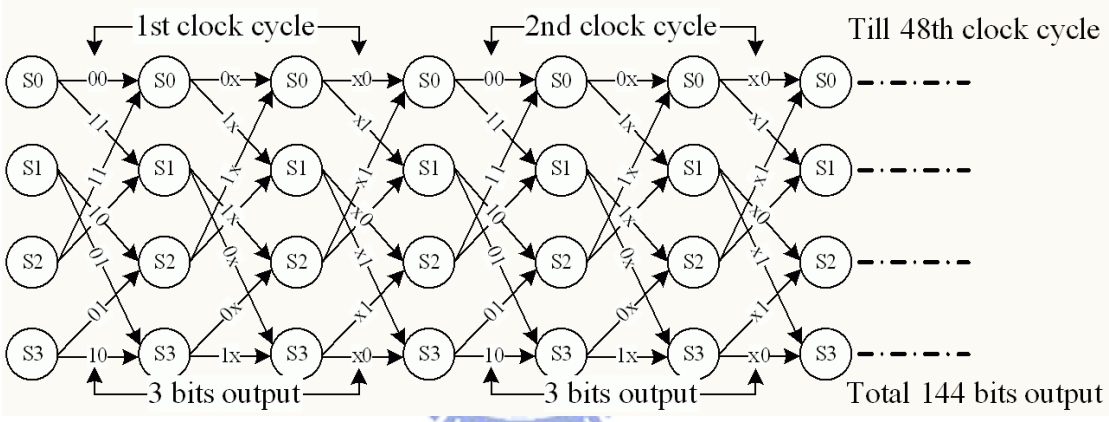

Figure 39: The 16-QAM 3/4 decoding mechanism each OFDM symbol

4.3.4 Decoding Mechanism of 64-QAM

In 64-QAM, there are six coded bits per subcarrier. In 64-QAM 2/3, this is a special case different from other cases. Through observing Figure 35, we find out that we need two un-punctured stages and one punctured stage of 2/3 decoding stage to form 3 decoding stages. We cannot take only one un-punctured and one punctured decoding stage because the total decoding clock cycles will over 80 clock cycles. In 64-QAM 3/4, it is the same as 16-QAM 3/4. In Figure 40 and Figure 41, the decoding

mechanism of 64-QAM 2/3 and 64-QAM 3/4 are shown.

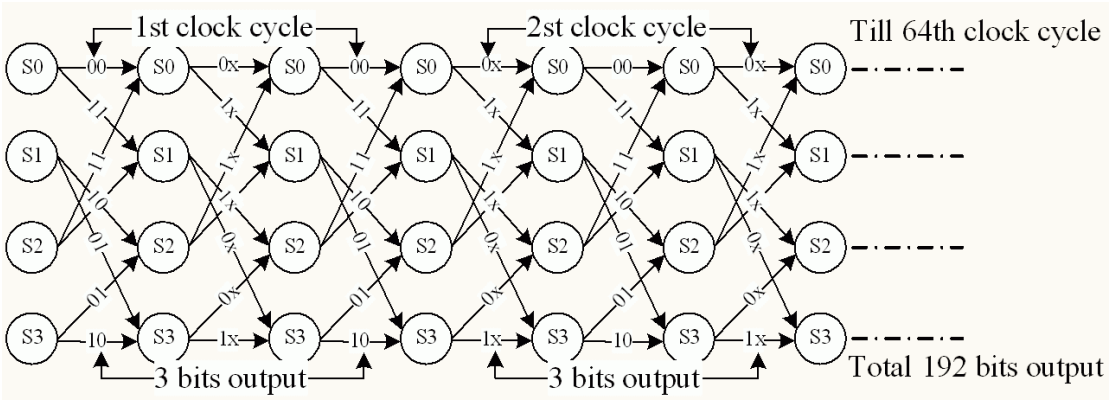

Figure 40: The 64-QAM 2/3 decoding mechanism each OFDM symbol

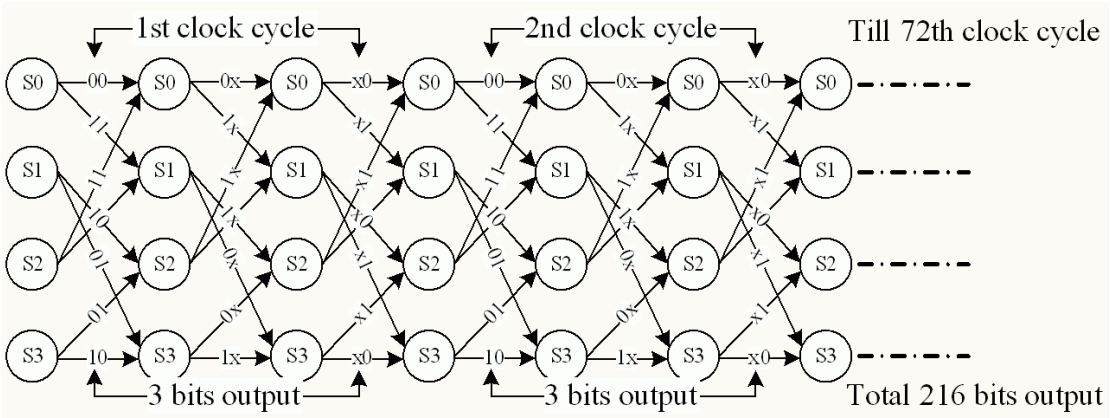

Figure 41: The 64-QAM 3/4 decoding mechanism each OFDM symbol

4.3.5 Throughput Issue

We know that there are 80 subcarriers in each OFDM symbol and only 48 subcarriers are data subcarriers. Therefore, we can utilize the extra 32 subcarriers to increase the throughput. But the problem is that we can accomplish the decoding task during one OFDM symbol interval. Through observing Table 6, it is clear that there are 8, 12, 16, 24, 32, 48, 64, 72 clock cycles for specified modulation decoding task,

respectively. And there are 72, 68, 64, 56, 48, 16, 8 clock cycles for waiting next OFDM symbol. Therefore, as long as the equalizer throws the data to the outer receiver during the interval less than 80 clock cycles, and the throughput will be improved even if we use low-QAM modulation.

4.4 Viterbi Decoder

4.4.1 Proposed Architecture

Since the employment of 3-stage radix-2 ACS architecture, it really produces huge demand of gate count. It is clear that the decrease of gate count can be achieved by making the word-length short inside ACS module. The traditional metric scheme of Euclidean distance makes the word-length longer than the one of correlator. The gate count comparison between the two algorithms will be shown in the following section. The architectures of two algorithms are shown in Figure 42 and Figure 43.

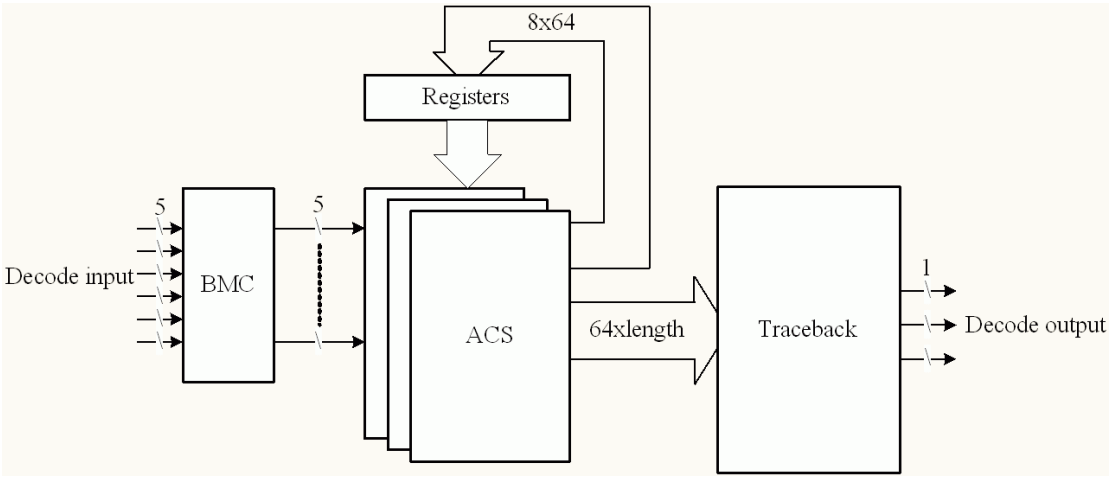

Figure 42: The architecture of correlator algorithm

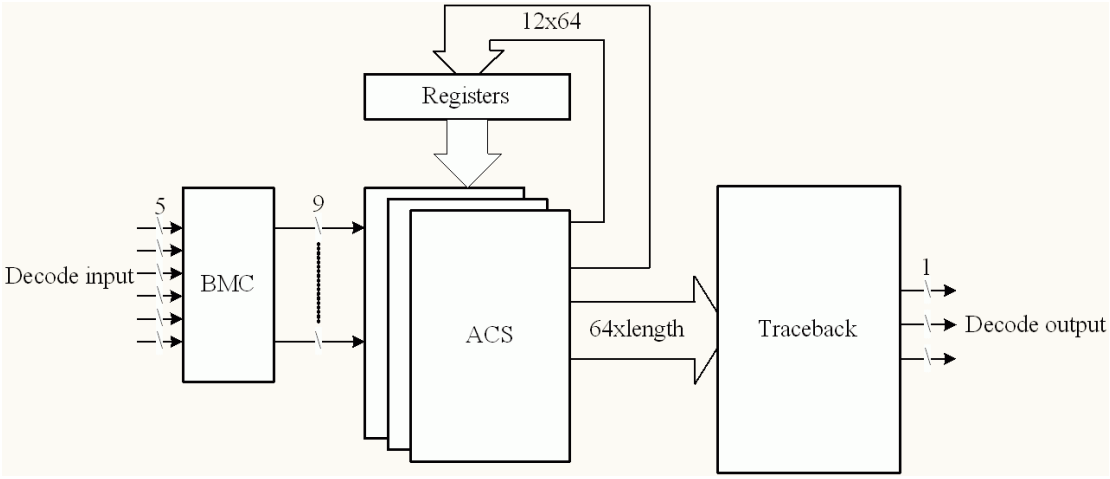

Figure 43: The architecture of Euclidean distance algorithm

4.4.2 BMC Module

The only difference between the two architectures mentioned in the previous section is the word-length of output. The soft decision resolutions we adopt is 4-bit and the word-length of the depuncture control line is 1-bit, and then it makes the input word-length of the BMC module is 5-bit. The BMC module consists of the calculation of real part and imaginary part metrics, and then the output word-length is 5-bit.

4.4.3 ACS Module

4.4.3.1 Implementation Issues

The proposed architecture of Viterbi decoder is shown is Figure 42. For the proposed ACS module, the decoding speed is always 60Mbps. IEEE 802.11a supports eight different kinds of data rates, which are all less than 60Mbps. Such architecture provides the ability of decoding to support the data rates .It is another viewpoint of applying the proposed architecture for IEEE 802.11a.

In general, the initial state of trellis diagram is the first state of trellis diagram. So the initial value of the registers storing survival metrics has to be considered. The first register shall be set to be zero, and other registers shall be set to be specified value while the minimum metric algorithm is adopted. In contrast with the minimum one, the first register shall be set to be specified value, and other registers shall be set to be zero. The specified value shall be small enough to distinguish from other states. The trellis diagram of convolutional code (2, 1, 7) has a characteristic: when it goes pass 6 stages of trellis diagram, the minimum state will replace other states. Therefore, the specified value shall be six times of metric value.

As for the word-length issue of ACS module, we considerate the worst case about the storage of metric data. Due to the reason we have talked about, six times of metric value appears probably. So we extend the word-length of registers from 5-bit to 8-bit. Through observing the condition of registers accessing, as long as we apply high-QAM technology and high SNR environment, the MSB of registers are not used probably. But for low-QAM technology and low SNR environment, the MSB of registers are usually used. For all possible condition, we need to consider the worst case of registers accessing.

4.4.3.2 ACS Overflow Prevention

We know that the operation of ACS module is recursive, and the word-length of ACS module is finite. Therefore, if we do not prevent the overflow from appearing, the results of survivors will go wrong. In the literature, the common method of overflow prevention is described as follows: when the overflow happens, we subtract the minimum path metric at each state. And the minimum path metric will be set to be zero. When the path goes through 6-stage trellis diagram, the minimum path will replace other metrics. Therefore, the overflow problem would never happen [6] [9] [10].

For the method we have discussed above, it makes the timing critical path exist. So, we propose another method to solve the overflow problem described as follows. We detect the value levels of registers. If one of the metrics arrives the specified value, we subtract the specified value at each state. But we shall take care on the difference between the minimum and maximum metrics. So we do not need to use the circuit which searches the minimum metric within 64 elements.

For the proposed ACS architecture, we can choose the interval of each 3-stage or each stage for overflow prevention operation. And the specified value depends on the interval of operation.

4.4.4 Traceback Module

According to section 3.4.3, we adopt the traceback algorithm for decoding mechanism. We propose the traceback element (TE) depicted in Figure 44. TE consists of two sub-components. One is the upper element, and the other is the lower element. In Figure 45, the nodes of the even number belong to the upper element and the nodes of the odd number belong to the lower element. It is clear that the decoding direction is the reverse direction of traceback algorithm. So the input of trellis diagram node is the output of TE, and the output of trellis diagram node is the input of TE. The upper element provides two upper branches of the next stage in traceback direction, and the lower element provides two lower branches. We set the survivor to be “0” while the survival path is the upper one, whereas “1” means the survival path is the lower one. If the upper branch and the lower branch are set to be “0” and “1” respectively, it means the survival path exists in this state. The traceback architecture is a combinational circuit shown in Figure 46, including 64xTraceback-length traceback elements. The enable component is to enable the state which has the minimum metric. If the state number belongs to the even one, the enable component set the input to be “0”, and if the state number belongs to the odd one, the enable component set the input to be ”1”. Therefore, it is easy for us to decode the final stage in traceback direction. To OR all the outputs of the lower branch, and detect whether

it is “1” or not, and then we know the decode bit being “1”.

For the demand of real-time decoding mechanism, the information of survivors must be transmitted to the traceback architecture parallel. Therefore, we use registers to be the storage components of survivors.

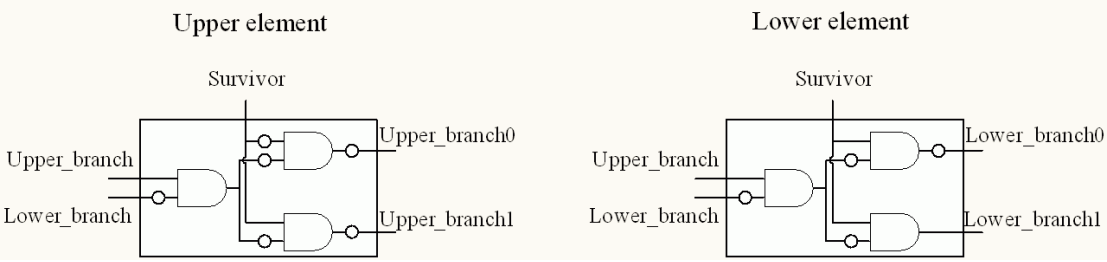

Figure 44: The traceback element

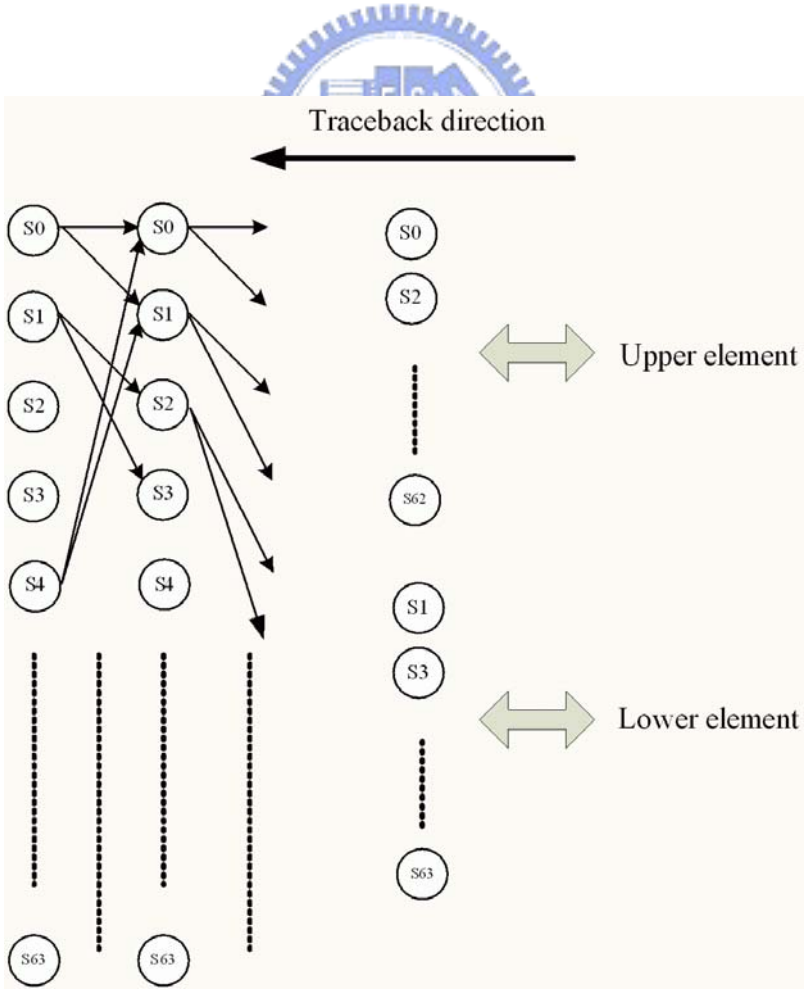

Figure 45: The location of the upper and lower elements

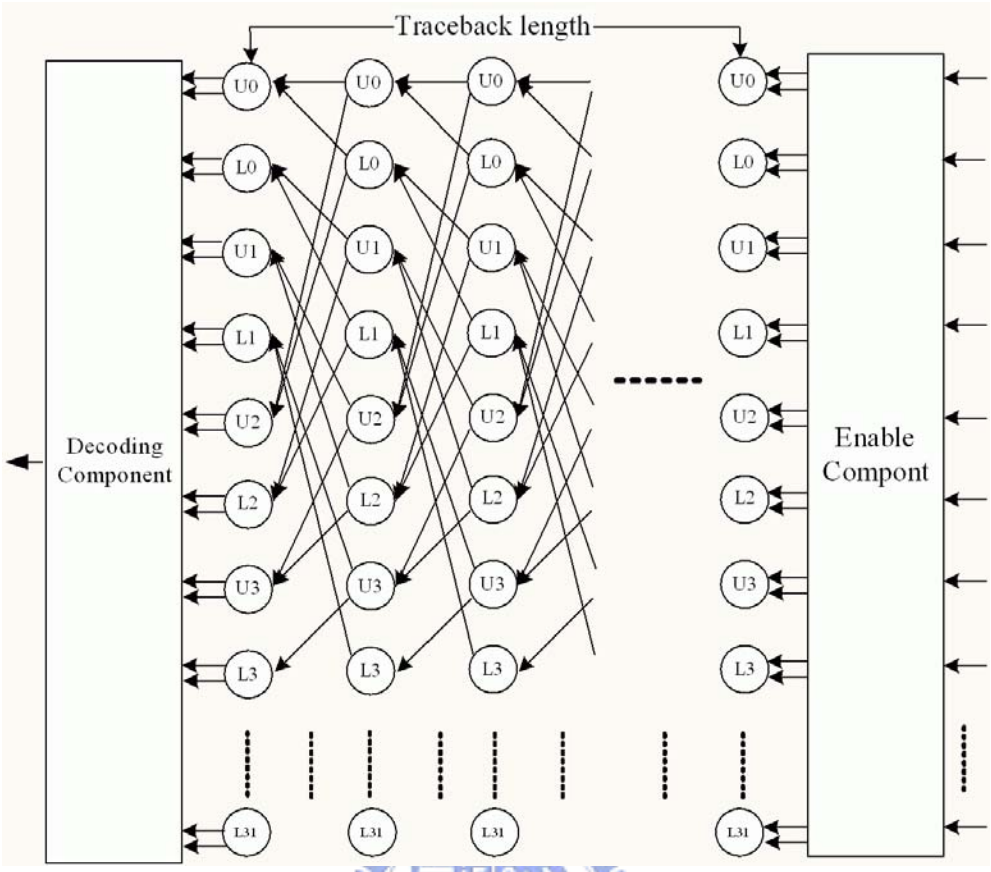

Figure 46: The traceback architecture

4.4.5 The Comparison between Different Traceback Length

To modify the traceback length affects the gate count, including ACS module and traceback module. As described in Table 7, the result of gate count varied with traceback length is presented. The relationship of the gate count with different traceback-length is linear.

4.4.6 Power Reduce

According to different modulation and coding rate, we can find the most suitable

traceback-length for the specified case. For the demand of real-time decoding, we use huge registers for the survivor storage. For the reason above, we can set different traceback-length parameter to be the range of traceback and ACS modules. Therefore, we will not move the data which locate outside the traceback-length and the power reducing can be achieved.

Table 7: The result of gate count with traceback-length 90 and 63

Length	90	63
ACS	123,532	101,006
Trace-back	43,525	30,723
Total	167,057	131,729

Unit: gate count

4.5 Implementation Results

The chip is implemented by cell-based design flow, and fabricated in 0.18 CMOS process. We use SYNOPSYS Design Analyzer to synthesize the gate-level Verilog file. And the parameters of the outer receiver are: 4-bit soft decision and traceback-length 63. The gate count of demapping, deinterleaver, depuncture, and Viterbi decoder is shown in Table 8, respectively. The maximum operation frequency is 25MHz. The PAR process of layout is applied with CADENCE SoC Encounter. The

area of the layout is shown in Table 9, including the cases of traceback-length 90 and traceback-length 63. Figure 47 and Figure 48 indicate the micro photo and packet photo of the proposed outer receiver, respectively.

Table 8: The gate count of the proposed outer receiver

	Gate count
Demapping	5,657
Deinterleaver	28,284
Depuncture	310
Viterbi decoder	131,729
Total	169,600

Table 9: The layout area of the proposed outer receiver

	Length 90	Length 63
Core size	1702x1702	1582x1582
Chip size	2198x2198	2075x2075
Power dissipation	82.06mW	78.85mW

Unit: um^2
@1.8V

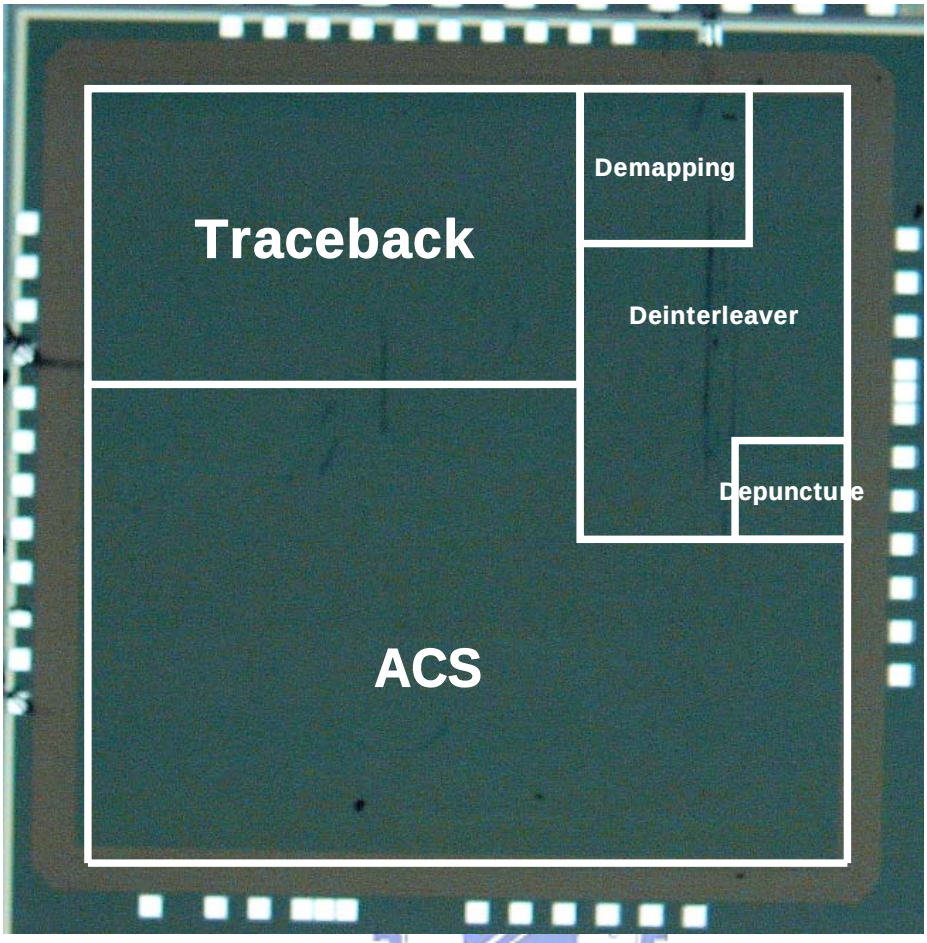

Figure 47: The micro photo of the outer receiver chip

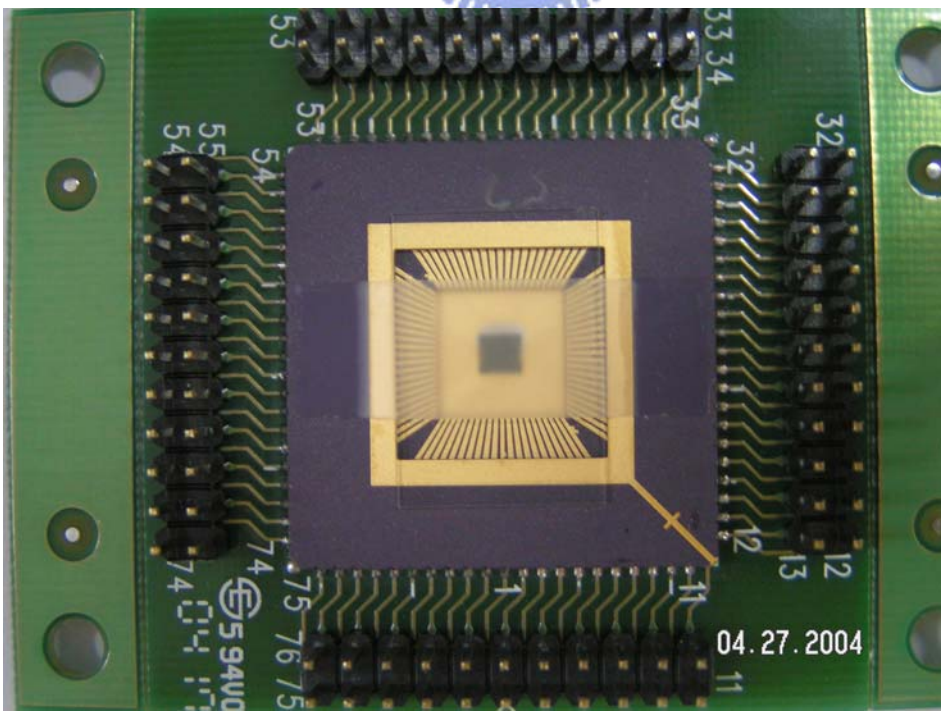

Figure 48: The packet view of the outer receiver

4.6 Verification

The environment of hardware verification is set up on FPGA, HP Pattern Generator, and Logic Analyzer. The specification of the FPGA is Xilinx xcv2000e6bg560. The setup of verification equipment is shown in Figure 49. HP Pattern Generator generates the input patterns which act as the equalizer. Let the test patterns pass through the outer receiver on FPGA and the output of FPGA is then sampled by Logic Analyzer.

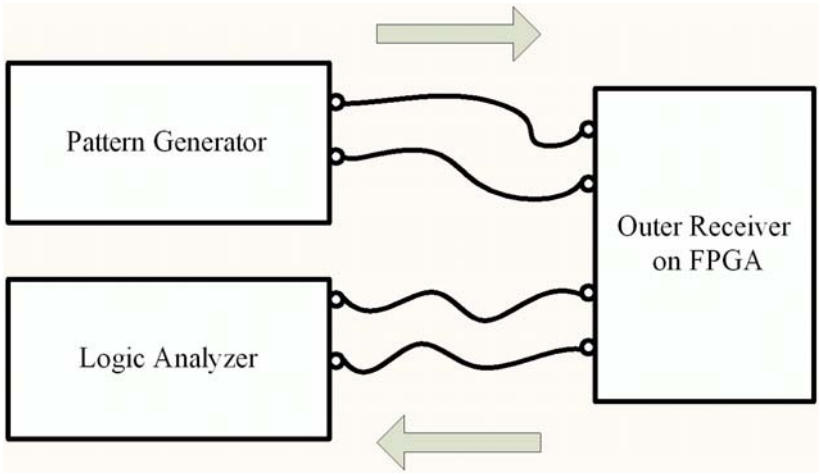

Figure 49: The setup of verification platform

We take the header as the test pattern. Figure 50 shows the result of the outer receiver on FPGA. If the header pattern passes the testing, it means the function works exactly.

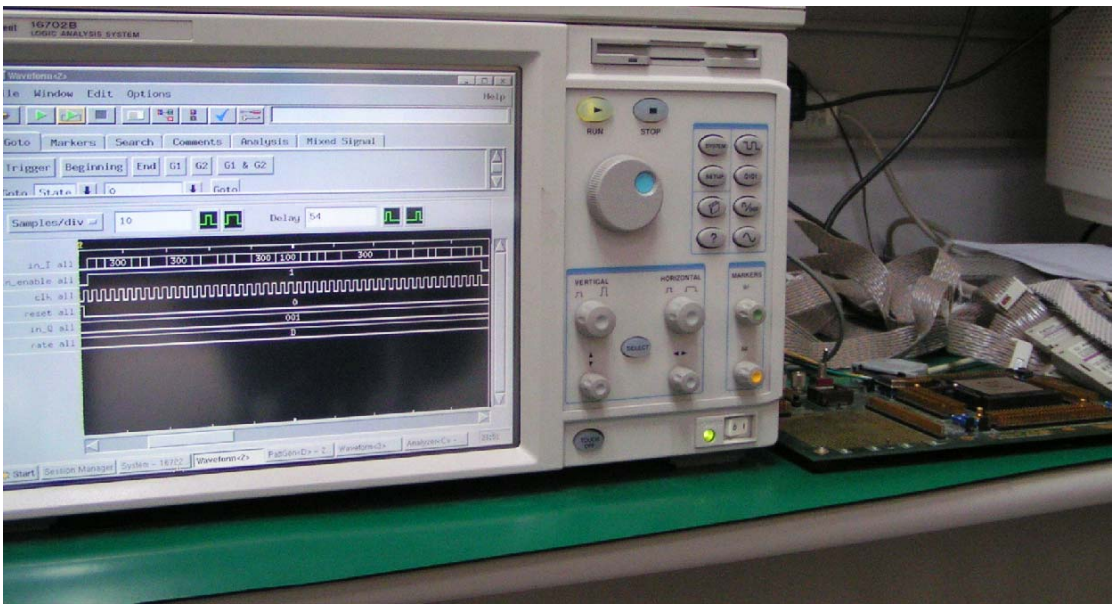

Figure 50: Pattern Generator, Logic Analyzer and Xilinx xcv2000e6bg560 FPGA.