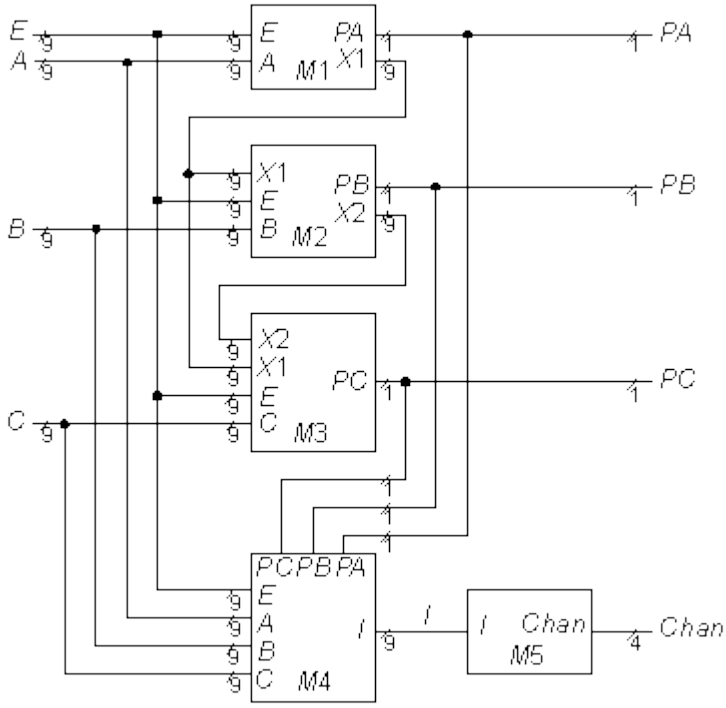
ISCAS-85 C432

Вырезано из http://web.eecs.umich.edu/~jhayes/iscas.restore/c432.html

27-канальный контроллер прерываний ISCAS-85 C432



Статистика: 36 входов; 7 выходов; 160 ворот; автобусные переводы

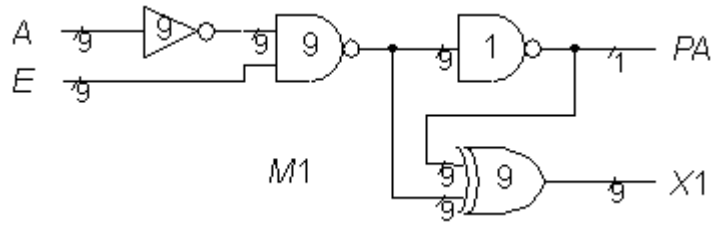
Функция: c432 - это 27-канальный контроллер прерываний. Входные каналы сгруппированы в три 9-битные шины (мы называем их A, B и C), где позиция бита в каждой шине определяет приоритет запроса прерывания. Четвертая 9-битная входная шина (называемая E) включает и отключает запросы прерывания в соответствующих положениях бит. На рисунке выше кратко изображена схема. На рисунке выше представлены модули с надписью M1 , M2 , M3 , M4 и M5 , которые содержат основную логику.

Контроллер прерывания имеет три шины запроса прерываний A, B и C, каждый из которых имеет девять битов или каналов и одну шину разрешения канала E. Применяются следующие правила приоритета: A [i]> B [j]> C [k], для любых i, j, k; т.е. шина A имеет самый высокий приоритет, а шина C - самая низкая. Внутри каждой шины канал с более высоким индексом имеет приоритет над одним с более низким индексом; например, A [i]> A [j], если i> j. Если E [i] = 0, то входы A [i], B [i] и C [i] не учитываются.

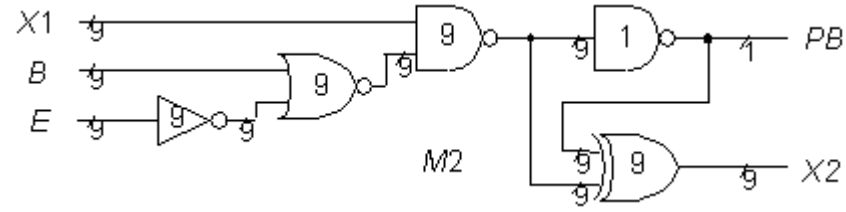
Семь выходов PA, PB, PC и Chan [3: 0] определяют, какие каналы подтвердили запросы прерывания. Признается только канал наивысшего приоритета в запрашивающей шине с наивысшим приоритетом. Исключением является то, что если два или более

прерываний выдают запросы на подтвержденный канал, каждая шина будет подтверждена. Например, если A [4], A [2], B [6] и C [4] имеют ожидающие запросы, A [4] и C [4] подтверждаются. Модуль M5 является 9-строчным-4-строчным приоритетным кодировщиком. Выходная строка с номером 421 фактически производит инвертированный ответ Chan [3], показанный в таблице истинности . Мы взяли на себя смелость добавить инвертор для вывода 421 для формирования Chan [3] для этой таблицы (но не в моделях).

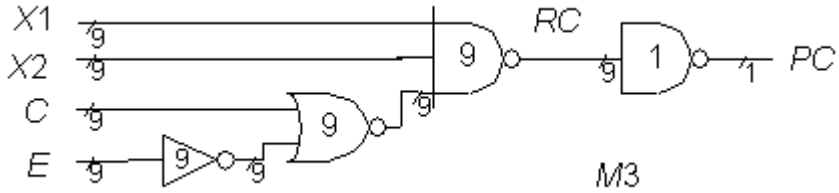
M1



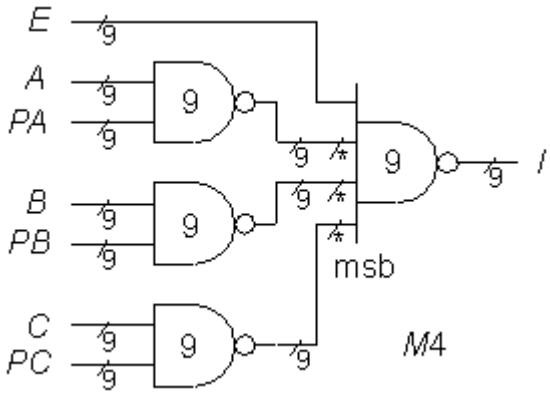
M2



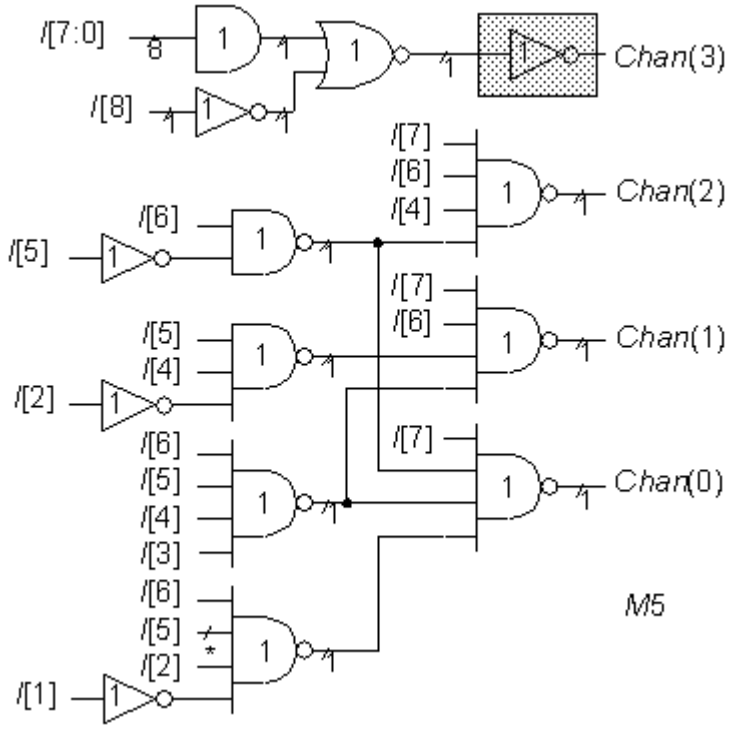
M3



M4



M5



ОПИСАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ VERILOG СХЕМЫ ЭТАЛОНА (BENCHMARK) c432

```
/*****************************************************************************
 *                                                                           *
 * VERILOG HIGH-LEVEL DESCRIPTION OF THE ISCAS-85 BENCHMARK CIRCUIT c432   *
 *                                                                           *
 * Function: 27-channel interrupt controller                        *
 *                                                                           *
 * Written by: Mark C. Hansen                                        *
 *                                                                           *
 * Last modified: Oct 17, 1997                                       *
 *                                                                           *
 *****************************************************************************/
module Circuit432 (in4, in17, in30, in43, in56, in69, in82, in95, in108,
              in1, in11, in24, in37, in50, in63, in76, in89, in102,
              in8, in21, in34, in47, in60, in73, in86, in99, in112,
              in14, in27, in40, in53, in66, in79, in92, in105, in115,
              out223, out329, out370,
              out421, out430, out431, out432);
input         in4, in17, in30, in43, in56, in69, in82, in95, in108,
             in1, in11, in24, in37, in50, in63, in76, in89, in102,
             in8, in21, in34, in47, in60, in73, in86, in99, in112,
             in14, in27, in40, in53, in66, in79, in92, in105, in115;
 output        out223, out329, out370,
             out421, out430, out431, out432;
wire [8:0]    A, B, C, E;
 wire          PA, PB, PC;
 wire [3:0]    Chan;
assign
    E[8:0] = { in4, in17, in30, in43, in56, in69, in82, in95, in108 },
    A[8:0] = { in1, in11, in24, in37, in50, in63, in76, in89, in102 },
    B[8:0] = { in8, in21, in34, in47, in60, in73, in86, in99, in112 },
    C[8:0] = { in14, in27, in40, in53, in66, in79, in92, in105, in115 },
    PA = out223,
    PB = out329,
```

```
    PC = out370,
    Chan[3:0] = { out421, out430, out431, out432 };

  TopLevel432 Ckt432 (E, A, B, C, PA, PB, PC, Chan);
endmodule /* Circuit432 */
/*************************************************************************/
module TopLevel432 (E, A, B, C, PA, PB, PC, Chan);
input[8:0]        E, A, B, C;
  output          PA, PB, PC;
  output[3:0] Chan;
  wire[8:0] X1, X2, I;
PriorityA M1(E, A, PA, X1);
  PriorityB M2(E, X1, B, PB, X2);
  PriorityC M3(E, X1, X2, C, PC);
  EncodeChan M4(E, A, B, C, PA, PB, PC, I);
  DecodeChan M5(I, Chan);
endmodule /* TopLevel432 */
/*************************************************************************/
module PriorityA(E, A, PA, X1);
input[8:0] E, A;
  output    PA;
  output[8:0] X1;

  wire [8:0] Ab, EAb;
not Ab0(Ab[0], A[0]);
  not Ab1(Ab[1], A[1]);
  not Ab2(Ab[2], A[2]);
  not Ab3(Ab[3], A[3]);
  not Ab4(Ab[4], A[4]);
  not Ab5(Ab[5], A[5]);
  not Ab6(Ab[6], A[6]);
  not Ab7(Ab[7], A[7]);
  not Ab8(Ab[8], A[8]);

  nand EAb0(EAb[0], Ab[0], E[0]);
  nand EAb1(EAb[1], Ab[1], E[1]);
  nand EAb2(EAb[2], Ab[2], E[2]);
  nand EAb3(EAb[3], Ab[3], E[3]);
  nand EAb4(EAb[4], Ab[4], E[4]);
  nand EAb5(EAb[5], Ab[5], E[5]);
  nand EAb6(EAb[6], Ab[6], E[6]);
  nand EAb7(EAb[7], Ab[7], E[7]);
  nand EAb8(EAb[8], Ab[8], E[8]);
and PAigate(PAi,EAb[0],EAb[1],EAb[2],EAb[3],EAb[4],EAb[5],EAb[6],EAb[7]);
  nand PAgate(PA,PAi,EAb[8]);
xor X10(X1[0], PA, EAb[0]);
  xor X11(X1[1], PA, EAb[1]);
  xor X12(X1[2], PA, EAb[2]);
  xor X13(X1[3], PA, EAb[3]);
  xor X14(X1[4], PA, EAb[4]);
  xor X15(X1[5], PA, EAb[5]);
  xor X16(X1[6], PA, EAb[6]);
  xor X17(X1[7], PA, EAb[7]);
  xor X18(X1[8], PA, EAb[8]);
endmodule /* PriorityA */
/*************************************************************************/
module PriorityB(E, X1, B, PB, X2);
```

```
input[8:0] E, X1, B;
  output    PB;
  output[8:0] X2;

  wire [8:0] Eb, EbB, XEB;
not Eb0(Eb[0], E[0]);
  not Eb1(Eb[1], E[1]);
  not Eb2(Eb[2], E[2]);
  not Eb3(Eb[3], E[3]);
  not Eb4(Eb[4], E[4]);
  not Eb5(Eb[5], E[5]);
  not Eb6(Eb[6], E[6]);
  not Eb7(Eb[7], E[7]);
  not Eb8(Eb[8], E[8]);

  nor EbB0(EbB[0], Eb[0], B[0]);
  nor EbB1(EbB[1], Eb[1], B[1]);
  nor EbB2(EbB[2], Eb[2], B[2]);
  nor EbB3(EbB[3], Eb[3], B[3]);
  nor EbB4(EbB[4], Eb[4], B[4]);
  nor EbB5(EbB[5], Eb[5], B[5]);
  nor EbB6(EbB[6], Eb[6], B[6]);
  nor EbB7(EbB[7], Eb[7], B[7]);
  nor EbB8(EbB[8], Eb[8], B[8]);
nand XEB0(XEB[0], EbB[0], X1[0]);
  nand XEB1(XEB[1], EbB[1], X1[1]);
  nand XEB2(XEB[2], EbB[2], X1[2]);
  nand XEB3(XEB[3], EbB[3], X1[3]);
  nand XEB4(XEB[4], EbB[4], X1[4]);
  nand XEB5(XEB[5], EbB[5], X1[5]);
  nand XEB6(XEB[6], EbB[6], X1[6]);
  nand XEB7(XEB[7], EbB[7], X1[7]);
  nand XEB8(XEB[8], EbB[8], X1[8]);
and PBigate(PBi,XEB[0],XEB[1],XEB[2],XEB[3],XEB[4],XEB[5],XEB[6],XEB[7]);
  nand PBgate(PB,PBi,XEB[8]);
xor X20(X2[0], PB, XEB[0]);
  xor X21(X2[1], PB, XEB[1]);
  xor X22(X2[2], PB, XEB[2]);
  xor X23(X2[3], PB, XEB[3]);
  xor X24(X2[4], PB, XEB[4]);
  xor X25(X2[5], PB, XEB[5]);
  xor X26(X2[6], PB, XEB[6]);
  xor X27(X2[7], PB, XEB[7]);
  xor X28(X2[8], PB, XEB[8]);
endmodule /* PriorityB */
/*************************************************************************/
module PriorityC(E, X1, X2, C, PC);
input[8:0] E, X1, X2, C;
  output    PC;

  wire [8:0] Eb, EbC, XEC;
not Eb0(Eb[0], E[0]);
  not Eb1(Eb[1], E[1]);
  not Eb2(Eb[2], E[2]);
  not Eb3(Eb[3], E[3]);
  not Eb4(Eb[4], E[4]);
  not Eb5(Eb[5], E[5]);
```

```
  not Eb6(Eb[6], E[6]);
  not Eb7(Eb[7], E[7]);
  not Eb8(Eb[8], E[8]);

  nor EbC0(EbC[0], Eb[0], C[0]);
  nor EbC1(EbC[1], Eb[1], C[1]);
  nor EbC2(EbC[2], Eb[2], C[2]);
  nor EbC3(EbC[3], Eb[3], C[3]);
  nor EbC4(EbC[4], Eb[4], C[4]);
  nor EbC5(EbC[5], Eb[5], C[5]);
  nor EbC6(EbC[6], Eb[6], C[6]);
  nor EbC7(EbC[7], Eb[7], C[7]);
  nor EbC8(EbC[8], Eb[8], C[8]);
nand XEC0(XEC[0], EbC[0], X1[0], X2[0]);
  nand XEC1(XEC[1], EbC[1], X1[1], X2[1]);
  nand XEC2(XEC[2], EbC[2], X1[2], X2[2]);
  nand XEC3(XEC[3], EbC[3], X1[3], X2[3]);
  nand XEC4(XEC[4], EbC[4], X1[4], X2[4]);
  nand XEC5(XEC[5], EbC[5], X1[5], X2[5]);
  nand XEC6(XEC[6], EbC[6], X1[6], X2[6]);
  nand XEC7(XEC[7], EbC[7], X1[7], X2[7]);
  nand XEC8(XEC[8], EbC[8], X1[8], X2[8]);
and PCigate(PCi, XEC[0],XEC[1],XEC[2],XEC[3],XEC[4],XEC[5],XEC[6],XEC[7]);
  nand PCgate(PC, PCi, XEC[8]);
endmodule /*PriorityC */
/*****************************************************************************/
module EncodeChan(E, A, B, C, PA, PB, PC, I);
input[8:0] E, A, B, C;
  input PA, PB, PC;
  output[8:0] I;
wire [8:0] APA, BPB, CPC;
nand APA0(APA[0], A[0], PA);
  nand APA1(APA[1], A[1], PA);
  nand APA2(APA[2], A[2], PA);
  nand APA3(APA[3], A[3], PA);
  nand APA4(APA[4], A[4], PA);
  nand APA5(APA[5], A[5], PA);
  nand APA6(APA[6], A[6], PA);
  nand APA7(APA[7], A[7], PA);
  nand APA8(APA[8], A[8], PA);
nand BPB0(BPB[0], B[0], PB);
  nand BPB1(BPB[1], B[1], PB);
  nand BPB2(BPB[2], B[2], PB);
  nand BPB3(BPB[3], B[3], PB);
  nand BPB4(BPB[4], B[4], PB);
  nand BPB5(BPB[5], B[5], PB);
  nand BPB6(BPB[6], B[6], PB);
  nand BPB7(BPB[7], B[7], PB);
  nand BPB8(BPB[8], B[8], PB);

  nand CPC0(CPC[0], C[0], PC);
  nand CPC1(CPC[1], C[1], PC);
  nand CPC2(CPC[2], C[2], PC);
  nand CPC3(CPC[3], C[3], PC);
  nand CPC4(CPC[4], C[4], PC);
  nand CPC5(CPC[5], C[5], PC);
  nand CPC6(CPC[6], C[6], PC);
```

```
  nand CPC7(CPC[7], C[7], PC);
  nand CPC8(CPC[8], C[8], PC);
nand I0(I[0], E[0], APA[0], BPB[0], CPC[0]);
  nand I1(I[1], E[1], APA[1], BPB[1], CPC[1]);
  nand I2(I[2], E[2], APA[2], BPB[2], CPC[2]);
  nand I3(I[3], E[3], APA[3], BPB[3], CPC[3]);
  nand I4(I[4], E[4], APA[4], BPB[4], CPC[4]);
  nand I5(I[5], E[5], APA[5], BPB[5], CPC[5]);
  nand I6(I[6], E[6], APA[6], BPB[6], CPC[6]);
  nand I7(I[7], E[7], APA[7], BPB[7], CPC[7]);
  nand I8(I[8], E[8], APA[8], BPB[8], CPC[8]);
endmodule /* EncodeChan */
/*************************************************************************/
module DecodeChan(I, Chan);

  input[8:0] I;
  output[3:0] Chan;
wire Iand, Ib8, Ib1, Ib2, Ib3, Ib5, I56, I245, I3456, I1256;
and Iandgate(Iand, I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]);
  not Ib8gate(Ib8, I[8]);
  nor Chan3(Chan[3], Iand, Ib8);
not Ib1gate(Ib1, I[1]);
  not Ib2gate(Ib2, I[2]);
  not Ib3gate(Ib3, I[3]);
  not Ib5gate(Ib5, I[5]);

  nand I56gate(I56, Ib5, I[6]);
  nand I245gate(I245, Ib2, I[4], I[5]);
  nand I3456gate(I3456, Ib3, I[4], I[5], I[6]);
  nand I1256gate(I1256, Ib1, I[2], I[5], I[6]);
nand Chan2(Chan[2], I[4], I[6], I[7], I56);
  nand Chan1(Chan[1], I[6], I[7], I245, I3456);
  nand Chan0(Chan[0], I[7], I56, I1256, I3456);
endmodule /* DecodeChan */
```

ОПИСАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ VERILOG ИЕРАРХИЧЕСКОЙ ПОВЕДЕНЧЕСКОЙ МОДЕЛИ c432

```
/*************************************************************************
 *                                                                       *
 *  VERILOG BEHAVIORAL DESCRIPTION OF THE ISCAS-85 BENCHMARK CIRCUIT c432   *
 *                                                                       *
 *  Function: 27-channel interrupt controller                        *
 *                                                                       *
 *  Written by: Mark C. Hansen                                        *
 *                                                                       *
 *  Last modified: Oct 17, 1997                                      *
 *                                                                       *
 *************************************************************************/
module Circuit432 (in4, in17, in30, in43, in56, in69, in82, in95, in108,
            in1, in11, in24, in37, in50, in63, in76, in89, in102,
            in8, in21, in34, in47, in60, in73, in86, in99, in112,
            in14, in27, in40, in53, in66, in79, in92, in105, in115,
            out223, out329, out370,
            out421, out430, out431, out432);
input       in4, in17, in30, in43, in56, in69, in82, in95, in108,
```

```
            in1, in11, in24, in37, in50, in63, in76, in89, in102,
            in8, in21, in34, in47, in60, in73, in86, in99, in112,
            in14, in27, in40, in53, in66, in79, in92, in105, in115;
  output       out223, out329, out370,
            out421, out430, out431, out432;
wire [8:0]    A, B, C, E;
  wire         PA, PB, PC;
  wire [3:0]    Chan;
assign
    E[8:0] = { in4, in17, in30, in43, in56, in69, in82, in95, in108 },
    A[8:0] = { in1, in11, in24, in37, in50, in63, in76, in89, in102 },
    B[8:0] = { in8, in21, in34, in47, in60, in73, in86, in99, in112 },
    C[8:0] = { in14, in27, in40, in53, in66, in79, in92, in105, in115 },
    PA = out223,
    PB = out329,
    PC = out370,
    Chan[3:0] = { out421, out430, out431, out432 };

  TopLevel432b Ckt432 (E, A, B, C, PA, PB, PC, Chan);
endmodule /* Circuit432 */
/*************************************************************************/
module TopLevel432b (E, A, B, C, PA, PB, PC, Chan);
input[8:0]          E, A, B, C;
  output            PA, PB, PC;
  output[3:0] Chan;
  wire[8:0] X1, X2, I;
PriorityA M1(E, A, PA, X1);
  PriorityB M2(E, X1, B, PB, X2);
  PriorityC M3(E, X1, X2, C, PC);
  EncodeChan M4(E, A, B, C, PA, PB, PC, I);
  DecodeChan M5(I, Chan);
endmodule /* TopLevel432b */
/*************************************************************************/
module PriorityA(E, A, PA, X1);
input[8:0] E, A;
  output    PA;
  output[8:0] X1;

  wire [8:0] Ab, EAb;
assign Ab = ~A;
  assign EAb = ~(Ab & E);
  assign PA = ~&EAb;
  assign X1 = EAb ^ {9{PA}};
endmodule /* PriorityA */
/*************************************************************************/
module PriorityB(E, X1, B, PB, X2);
input[8:0] E, X1, B;
  output    PB;
  output[8:0] X2;

  wire [8:0] Eb, EbB, XEB;
assign Eb = ~E;
  assign EbB = ~(Eb | B);
  assign XEB = ~(X1 & EbB);
  assign PB = ~&XEB;
  assign X2 = XEB ^ {9{PB}};
endmodule /* PriorityB */
```

```
/*****************************************************************************/
module PriorityC(E, X1, X2, C, PC);
input[8:0] E, X1, X2, C;
  output     PC;

  wire [8:0] Eb, EbC, XEC;
assign Eb = ~E;
  assign EbC = ~(Eb | C);
  assign XEC = ~(X1 & X2 & EbC);
  assign PC = ~&XEC;
endmodule /*PriorityC */
/*****************************************************************************/
module EncodeChan(E, A, B, C, PA, PB, PC, I);
input[8:0] E, A, B, C;
  input PA, PB, PC;
  output[8:0] I;
wire [8:0] APA, BPB, CPC;
assign APA = ~(A & {9{PA}});
  assign BPB = ~(B & {9{PB}});
  assign CPC = ~(C & {9{PC}});
  assign I = ~(E & APA & BPB & CPC);
endmodule /* EncodeChan */
/*****************************************************************************/
module DecodeChan(I, Chan);

  input[8:0] I;
  output[3:0] Chan;
wire Iand, I8b, I1b, I2b, I3b, I5b, I56, I245, I3456, I1256;
assign I8b = ~I[8];
  assign Iand = &I[7:0];
  assign Chan[3] = ~(I8b | Iand);
assign I1b = ~I[1];
  assign I2b = ~I[2];
  assign I3b = ~I[3];
  assign I5b = ~I[5];
assign I56 = ~(I5b & I[6]);
  assign I245 = ~(I2b & I[4] & I[5]);
  assign I3456 = ~(I3b & I[4] & I[5] & I[6]);
  assign I1256 = ~(I1b & I[2] & I[5] & I[6]);
assign Chan[2] = ~(I[4] & I[6] & I[7] & I56);
  assign Chan[1] = ~(I[6] & I[7] & I245 & I3456);
  assign Chan[0] = ~(I[7] & I56 & I1256 & I3456);
endmodule /* DecodeChan */
```

ОПИСАНИЕ ПОЛНОГО ФУНКЦИОНАЛЬНОГО ТЕСТА

```
111111111011111111011111111011111111
111111111011111111011111111101111111
111111111101111111101111111110111111
111111111110111111110111111111011111
111111111111011111111011111111101111
111111111111101111111101111111110111
111111111111110111111110111111111011
111111111111111011111111011111111101
111111111111111101111111101111111110
111111111111111110100111111101111111
111111111111111111010111111010000000
111111111111111111110110101111111111
```

11111111111111111111010111111111111
11111111111111111111100111111111111
11111111111111111111110111111111111
11111111111111111111111011111111111
11111111111111111111111101111111111
11111111111111111111111110111111111
11111111111111111111111111011111111
11111111111111111111111111101111111
11111111111111111111111111110111111
11111111111111111111111111111011111
11111111111111111111111111111101111
11111111111111111111111111111110111
11111111111111111111111111111111011
10000001010000001010000001010000000
01111110101111110101111101011111100
11111111111101101100010010000010010
11111111111110100111111111111111111
11111111101111111100000000100000000
11111111100111111111111111001111111
00000000000000000000000000000000000