**High-Level Model of c3540**

**Statistics:** 50 inputs; 22 outputs; 1669 gates

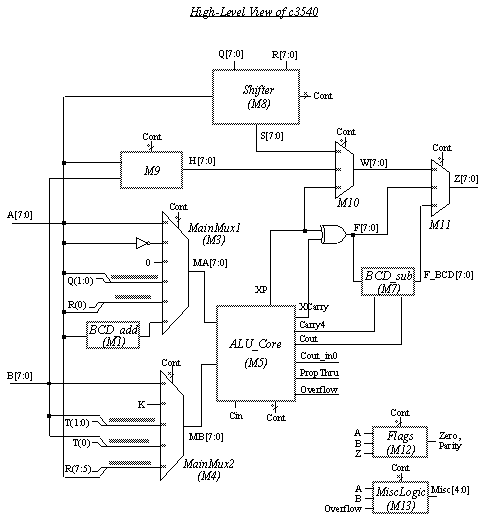
**Function:** 8-bit ALU with binary and BCD arithmetic, and logic and shift operations.

This benchmark is an 8-bit ALU that can perform binary and BCD arithmetic operations as well as logic and shift operations. Logic operations are intermixed with arithmetic ones, much as in the TTL 74181. BCD addition is done via a two's-complement adder by adding 6 to both digits of the first operand, and then subtracting 6 from the digits of the result if they do not generate a carry. A total of 14 control inputs are used for multiplexing and masking data inputs. Most multiplexers in this circuit have an odd-number of inputs, e.g, 3 and 5, and their selection is different between lower and upper digits (4 bits). The selection mechanism of M4 (output MB) is even more complicated; see below for details. The largest module is M5 (ALU\_Core), which consists of two 4-bit CLAs. Module M8 (Shifter) can shift the input bus A by 1 to 8 bits in either direction. Parity and zero flags are generated by module M12 (Flags) using the input buses A, B and the output bus Z; see the relevant figures or the Verilog model for their exact definition. Various logic functions of A and B are calculated by module M13 which does not have an apparent high-level structure.

[**Inputs/Outputs vs. Netlist Numbers**](http://web.eecs.umich.edu/%7Ejhayes/iscas.restore/c3540/c3540.html#pgfId=934023)

**Models:**

* I. Original ISCAS gate-level netlist
  + [in ISCAS-89 format](http://web.eecs.umich.edu/%7Ejhayes/iscas.restore/c3540/c3540.isc)
  + [in Verilog](http://web.eecs.umich.edu/%7Ejhayes/iscas.restore/c3540/c3540gate.v)
* II. [Verilog hierarchical netlist](http://web.eecs.umich.edu/%7Ejhayes/iscas.restore/c3540/c3540high.v) (functionally equivalent to I)
* III. [Verilog flat netlist](http://web.eecs.umich.edu/%7Ejhayes/iscas.restore/c3540/flat3540.v) (flat version of II; functionally equivalent to I, but with minor structural differences)



**Modules M1 (BCD\_add) and M7 (BCD\_sub)**

In order to perform BCD addition with a two's complement adder, module M1 adds 6 to each digit of the input bus, and module M7 subtracts 6 from each digit of the result if there is no carry from that digit.

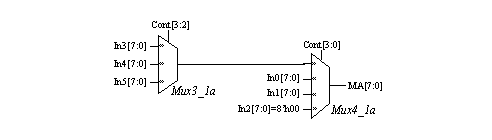
The logic equations for the lower digit of M1 (upper digit is similar):

* Out[3] = In[3] + In[2] + In[1]
* Out[2] = !(In[2] In[1])
* Out[1] = !In[1]
* Out[0] = In[0]

The logic equations for the lower digit of M7 (upper digit is similar):

* Out[3] = In[3].Carry4 + In[3]. In[2] . In[1]. !Carry4
* Out[2] = In[2] In[1]. !Carry4
* Out[1] = In[1] !Carry4
* Out[0] = In[0]

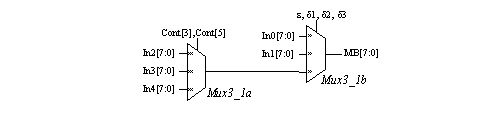
**Module M3 (MainMux1)**



This module consists of two cascaded multiplexers, as shown above. The control signals Cont[3:0] determine the select inputs of M3. Notice that selection is different between the lower and the upper digits of the output.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cont3[3:0] | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| MA | MA[3:0] | In0 | In0 | In0 | In3 | In0 | In0 | In1 | In5 | In2 | In0 | In2 | In4 | In2 | In5 | In1 | In5 |
| MA[7:4] | In0 | In0 | In0 | In3 | In2 | In0 | In1 | In5 | In0 | In0 | In0 | In4 | In2 | In5 | In1 | In5 |

**Module M4 (MainMux2)**



Like M3, Module M4 is made up of two cascaded multiplexers. However, it's select logic is more involved. The control signals Cont[6:3] and Cont[1:0] are decoded into the following select signals:

* CHi = Cont[0]. Cont[1]. !(Cont[3]. Cont[5])
* CLo1 = !Cont[0]. !Cont[5]. Cont[6]
* CLo2 = !Cont[0]. Cont[6]
* CLo3 = !Cont[0]. (Cont[5]+Cont[6])

|  |  |  |  |
| --- | --- | --- | --- |
| Selected input | Mux output MB[7:0] | | |
| **MB[2:0]** | **MB[3]** | **MB[7:4]** |
| In0 | !CHi. !CLo1 | !CHi. !CLo2 | !CHi. !CLo3 |
| In1 | !CHi. CLo1 | !CHi. CLo2 | !CHi. CLo3 |
| In2 | CHi. !Cont[3]. !Cont[4] | CHi .!Cont[3]. !Cont[4] | CHi. !Cont[3]. !Cont[4] |
| In3 | CHi. !Cont[3]. Cont[4] | CHi. !Cont[3]. Cont[4] | CHi. !Cont[3]. Cont[4] |
| In4 | CHi. Cont[3] | CHi. Cont[3] | CHi. Cont[3] |

Notice that since In1 is connected to the single bit K input, all the bits of MB can be set to K (0 or 1).

**Module M5 (ALU\_Core)**

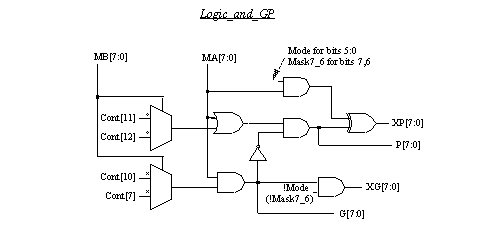
This is the largest module of the c3540 benchmark. The control inputs to this module are Cont[12:7] and Cont[2:0]. An internal signal called Mode determines whether a logic or an arithmetic operation is to be performed:

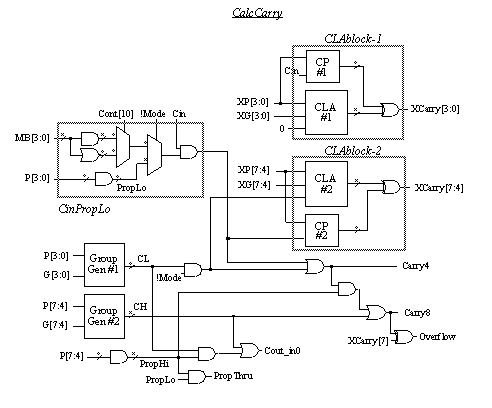
* Mode = !Cont[0]. Cont[1]. !Cont[2]. Cont[8]. Cont[9]

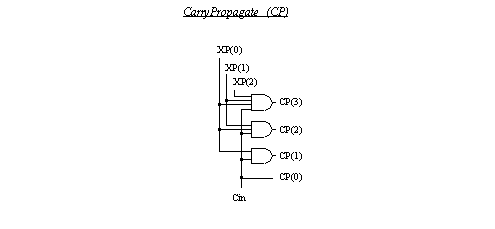
Mode is 1 for a logic operation, and 0 for an arithmetic operation. In the case of arithmetic operations, an additional control signal named Mask7\_6 is used to mask bits #7 and #6 of the MA bus:

* Mask7\_6 = !Cont[0]. Cont[1]. !Cont[2]. Cont[8]

As in the TTL circuit 74181, logic and arithmetic operations are intermixed. A block named Logic\_and\_GP computes both logic operations as well as the generate and propagate signals used for binary addition. The carry signals are computed by CalcCarry, and the final result of the ALU is obtained by XORing the carry signals with a modified propagate signal called XP.







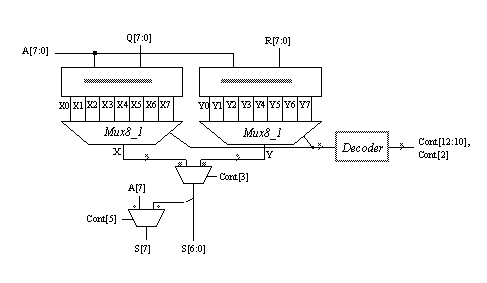
The set of logic and arithmetic operations performed by ALU\_Core is shown below. The table is valid for Cin=0. Logic 1 is added to the result if Cin=1 (this applies to arithmetic as well as logic operations.)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cont [11] | Cont[12] | Cont [10] | Cont[7] | Arithmetic Function  (XP xor XCarry) | Logic Function (XP) |
| 0 | 0 | 0 | 0 | MA | 0 |
| 0 | 0 | 0 | 1 | MA plus MA.MB | MA.MB |
| 0 | 0 | 1 | 0 | MA plus MA.!MB | MA.!MB |
| 0 | 0 | 1 | 1 | MA plus MA (shift left) | MA |
| 0 | 1 | 0 | 0 | MA+MB | !MA.MB |
| 0 | 1 | 0 | 1 | MA plus MB | MB |
| 0 | 1 | 1 | 0 | (MA+MB) plus MA.!MB | MA xor MB |
| 0 | 1 | 1 | 1 | (MA+MB) plus MA | MA+MB |
| 1 | 0 | 0 | 0 | MA+!MB | !MA.!MB |
| 1 | 0 | 0 | 1 | (MA+!MB) plus MA.MB | MA xnor MB |
| 1 | 0 | 1 | 0 | MA minus MB minus 1 | !MB |
| 1 | 0 | 1 | 1 | (MA+!MB) plus MA | MA+!MB |
| 1 | 1 | 0 | 0 | minus 1 (2's compl) | !MA |
| 1 | 1 | 0 | 1 | MA.MB minus 1 | !MA+MB |
| 1 | 1 | 1 | 0 | MA.!MB minus 1 | !MA+!MB |
| 1 | 1 | 1 | 1 | MA minus 1 | 1 |

Legend for operators:

* ! : complement
* + : logical OR
* . : logical AND
* xor : logical XOR
* xnor : logical XNOR
* plus : arithmetic add
* minus: arithmetic subtract

**Module M8 (Shifter)**

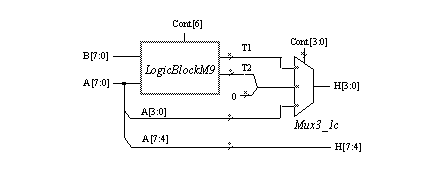


This module contains logic for shifting the input bus A by 1 to 8 bits in either direction. When shifting towards LSB (MSB), the empty bit positions are filled by a Q (R) bus. The Shifter decodes the control signals Cont[12:10], and Cont[2] into 8 signals, which are the select inputs for sixteen 8:1 multiplexers. Input selection for these multiplexers is shown below. The X and Y buses are fed into a 2:1 multiplexer controlled by Cont[3]. There is an additional multiplexer for bit #7 whose second input is A[7]; this can be used for shifting signed input data.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cont[2,10,11,12] | X | X[0] | X[1] | X[2] | X[3] | X[4] | X[5] | X[6] | X[7] |
| 9 | X0 | A[1] | A[2] | A[3] | A[4] | A[5] | A[6] | A[7] | Q[0] |
| 0-7,10 | X1 | A[2] | A[3] | A[4] | A[5] | A[6] | A[7] | Q[0] | Q[1] |
| 11 | X2 | A[3] | A[4] | A[5] | A[6] | A[7] | Q[0] | Q[1] | Q[2] |
| 12 | X3 | A[4] | A[5] | A[6] | A[7] | Q[0] | Q[1] | Q[2] | Q[3] |
| 13 | X4 | A[5] | A[6] | A[7] | Q[0] | Q[1] | Q[2] | Q[3] | Q[4] |
| 14 | X5 | A[6] | A[7] | Q[0] | Q[1] | Q[2] | Q[3] | Q[4] | Q[5] |
| 15 | X6 | A[7] | Q[0] | Q[1] | Q[2] | Q[3] | Q[4] | Q[5] | Q[6] |
| 8 | X7 | Q[0] | Q[1] | Q[2] | Q[3] | Q[4] | Q[5] | Q[6] | Q[7] |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cont[2,10,11,12] | Y | Y[0] | Y[1] | Y[2] | Y[3] | Y[4] | Y[5] | Y[6] | Y[7] |
| 9 | Y0 | R[7] | A[0] | A[1] | A[2] | A[3] | A[4] | A[5] | A[6] |
| 0-7,10 | Y1 | R[6] | R[7] | A[0] | A[1] | A[2] | A[3] | A[4] | A[5] |
| 11 | Y2 | R[5] | R[6] | R[7] | A[0] | A[1] | A[2] | A[3] | A[4] |
| 12 | Y3 | R[4] | R[5] | R[6] | R[7] | A[0] | A[1] | A[2] | A[3] |
| 13 | Y4 | R[3] | R[4] | R[5] | R[6] | R[7] | A[0] | A[1] | A[2] |
| 14 | Y5 | R[2] | R[3] | R[4] | R[5] | R[6] | R[7] | A[0] | A[1] |
| 15 | Y6 | R[1] | R[2] | R[3] | R[4] | R[5] | R[6] | R[7] | A[0] |
| 8 | Y7 | R[0] | R[1] | R[2] | R[3] | R[4] | R[5] | R[6] | R[7] |

**Module M9**

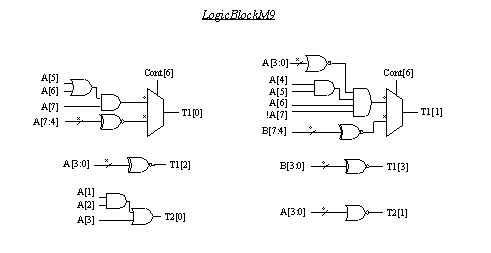


This is a relatively small module that appears to calculate some special-purpose logic functions of the input buses A and B. Five control inputs to this module are Cont[6] and Cont[3:0]. The select inputs of the 3:1 multiplexer Mux3\_1c are calculated by

* CHi = !(Cont[0]. !Cont[1]. Cont[2]. Cont[3])
* CLo = !(Cont[0]. !Cont[1]. Cont[2]. !Cont[3])

|  |  |  |
| --- | --- | --- |
| Select inputs | | Mux output H[3:0] |
| CHi | **CLo** |
| 0 | 0 | (not possible) |
| 0 | 1 | T1[3:0] |
| 1 | 0 | { 2'b00, T2[1:0] } |
| 1 | 1 | A[3:0] |

The details of LogicBlockM9 are shown below. It seems to calculate a non-standard function. It may be some type of code translation or encryption. The authors will be glad to know if you have any insight into the function of this particular block. Please contact us if you have any information.



**Module M10**

This module contains a set of eight 3:1 multiplexers of type Mux3\_1c. As with the other multiplexers, the lower and upper digits of the output are selected differently. The control signals Cont[7],Cont[3:0] are decoded into three select signals CHi, CLo1, CLo2 as follows.

* CHi = !(Cont[0]. Cont[1]. (!Cont[2] + Cont[7]))
* CLo1 = Cont[1]+Cont[2]+Cont[3]
* CLo2 = Cont[1]+Cont[3]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Select inputs | | Mux output W[3:0] | Select inputs | | Mux output W[7:4] |
| CHi | **CLo1** | **CHi** | **CLo2** |
| 0 | 0 | (not possible) | 0 | 0 | (not possible) |
| 0 | 1 | S[3:0] | 0 | 1 | S[7:4] |
| 1 | 0 | XP[3:0] | 1 | 0 | XP[7:4] |
| 1 | 1 | H[3:0] | 1 | 1 | H[7:4] |

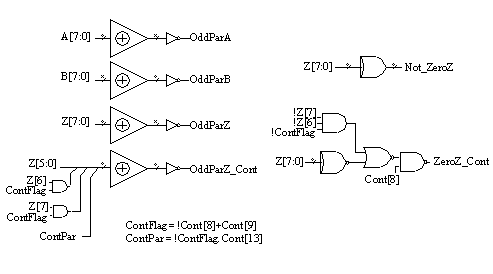
**Module M11**

This module consists of eight 3:1 multiplexers of type Mux3\_1c. Unlike M10, its lower and upper digits are selected by the same two signals which are decoded from the control signals Cont[6:5], Cont[2:0] as follows.

* CHi = Cont[0]. !(Cont[1]. !Cont[2]. Cont[6])
* CLo = !(Cont[0]. !Cont[1]. Cont[2]. !Cont[5])

|  |  |  |
| --- | --- | --- |
| Select inputs | | Mux output Z[7:0] |
| CHi | **CLo** |
| 0 | 0 | (not possible) |
| 0 | 1 | F[7:0] |
| 1 | 0 | F\_BCD[7:0] |
| 1 | 1 | W[7:0] |

**Module M12 (Flags)**



This module generates two zero flags and four parity outputs. As shown above, the parities are calculated from the input buses A and B, and the output bus Z. The zero flags are calculated from the output bus Z only. Three control inputs to M12 are Cont[13] and Cont[9:8]. Paralleling the type of masking used in ALU\_Core by Mask7\_6, this module can mask out Z[7:6] for the OddParZ\_Cont and ZeroZ\_Cont outputs using an internally generated ContFlag signal.

|  |  |
| --- | --- |
| Input | Netlist numbers |
| A[7:0] | 50, 58, 68, 77, 87, 97, 107, 116 |
| B[7:0] | 226, 232, 238, 244, 250, 257, 264, 270 |
| Q[7:0] | 124, 125, 128, 132, 137, 143, 150, 159 |
| R[7:0) | 283, 294, 303, 311, 317, 322, 326, 329 |
| T[1:0] | 222, 223 |
| Cin | 330 |
| K | 274 |
| Cont[13:0] | 2897, 200, 190, 179, 343, 213, 169, 45, 41, 1698, 33, 20, 13, 1 |

|  |  |
| --- | --- |
| Output | Netlist numbers |
| Z[7:0] | 375, 378, 381, 384, 387, 390, 393, 396 |
| OddParZ, OddParZ\_Cont | 402, 405 |
| OddParA | 351 |
| OddParB | 358 |
| NotZeroZ, ZeroZ\_Cont | 407, 409 |
| XCarry2, Cout\_in0 | 399, 369 |
| PropThru | 372 |
| MiscOuts[4:0] | 353, 355, 361, 364, 367 |