Verilator creates the following files in the output directory:

{prefix}.mk // Make include file for compiling

{prefix}\_classes.mk // Make include file with class names

For -cc and -sc mode, it also creates:

{prefix}.cpp // Top level C++ file

{prefix}.h // Top level header

////////////

{prefix}{each\_verilog\_module}.cpp // Lower level internal C++ files

{prefix}{each\_verilog\_module}.h // Lower level internal header files

VL\_ST\_SIG8(VHW1::\_\_Vtable1\_MuxOut[64],0,0);

Body

Always at HW1.v:11

C++ language

::

當宣告為一般型態時 MyClass CA，就使用點(.)來存取Class中的成員：  
MyClass CA;  
CA.add(0);       // 左邊為 Class變數

當宣告為指標型態時 MyClass \* CA，就使用箭頭(->)來存取Class中的成員：  
MyClass \* pCA; //pCA屬於MyClass的類別  
pCA->add(0);    // 左邊為 Class指標

雙冒號(::)只用在Class成員函式或Class成員變數中：

#if defined(或者是ifdef)<标识符(条件)>

<程序段1>

#endif

**正式:**

1. 程式架構 🡨

* 弄清變數等

1. 針對邏輯部分分析

一樣的格式

|  |  |  |
| --- | --- | --- |
| VL\_SIG8(\_\_Vtableidx1,5,0); | \_\_Vtableidx1 | 5,0 |
| static VL\_ST\_SIG8(\_\_Vtable1\_MuxOut[64],0,0); | \_\_Vtable1\_MuxOut[64] | 0,0 |

|  |  |  |
| --- | --- | --- |
| (IData)(vlTOPp->C) | 推測是叫用C之值 | Input，應為wire |
| VL\_RAND\_RESET\_I | 決定幾個bits |  |
| \_\_Vtableidx1 | 應為暫存所有腳位 |  |
| static VL\_ST\_SIG8(\_\_Vtable1\_MuxOut[64],0,0); |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

架構:

vlTOPp:A,B,C,D,Sel,MuxOut, \_\_Vtableidx1(應為暫存的)

---在生成的函數中禁用“ this-”引用，而是由verilator將從'vltopp-'開始生成絕對引用。

總架構:

void VHW1::\_\_Vconfigure:

1. 配置，first哪裡來的?

void VHW1::eval:

1. 當值改變時叫用
2. 關鍵字: vlSymsp, \_\_Vchange
3. Verilator has an important difference from an event based simulator; signal values that are changed by the VPI will not immediately propagate their values, instead the top level header file's eval() method must be called. Normally this would be part of the normal evaluation (i.e. the next clock edge), not as part of the value change. This makes the performance of VPI routines extremely fast compared to event based simulators, but can confuse some test-benches that expect immediate propagation.

void VHW1::\_eval\_initial\_loop:

1. Settle某個東西
2. 關鍵字: vlSymsp

VL\_INLINE\_OPT void VHW1::\_combo\_\_TOP\_\_1:

1. 前面的VL\_INLINE\_OPT不知為何。
2. 把ABCDSel都收進MuxOut中(不知有沒有運算過程，應該沒有)。
3. 關鍵字: vlTOPp，幾乎是所有變數的母

\*\*\*以下為相像程式

void VHW1::\_eval:

1. 和前面的eval很像
2. 以下幾個名字相像的def中:

VHW1\* \_\_restrict vlTOPp VL\_ATTR\_UNUSED = vlSymsp->TOPp;

不斷出現，可能為重要

1. vlTOPp->\_combo\_\_TOP\_\_1(vlSymsp);

void VHW1::\_eval\_initial:

1. 甚麼也沒有

void VHW1::final:

1. 可能是跟變數的設置有關

void VHW1::\_eval\_settle:

1. 與eval的相像一模一樣

目前的最後一個:

VL\_INLINE\_OPT QData VHW1::\_change\_request:

1. 那麼多個空格是class中的class嗎??
2. 將某個值設為false
3. 關鍵字: QData

void VHW1::\_ctor\_var\_reset:

1. ctor\_var\_reset程式最前面有被引用過\
2. 各個腳位的大小

**規則:**

1. VL\_DEBUG\_IF應該是除錯用的
2. ……

Always中:

將所有腳位全部or起來，有shift，故排起來像是:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | Sel[1] | Sel[0] |

全部or起來後變成像是100110存入(暫存器)\_\_Vtableidx1，然後將\_\_Vtable1\_MuxOut中暫存器的位置叫出來存入MuxOut的位置。

**困境:**

1. \_\_Vtable1\_MuxOut 的Vtable1網路上竟沒有結果

**可以嘗試:**

* 1. 搜尋VL\_DEBUG\_IF

目標:

1. \_\_Vtable1\_MuxOut意義
2. 與其他檔案比對
3. 釐清邏輯

關於其他檔案:

1. Syms系列:函式、變數定義內容方面沒啥內容
2. VHW1.mk沒看得懂的內容，內容不長
3. VHW1.h僅提及各函式名稱

\

11/8

|  |
| --- |
| /// Init time only, so slow is fine |
| extern IData VL\_RAND\_RESET\_I(int obits); ///< Random reset a signal  ///////extern 表明VL\_RAND\_RESET在其他地方會被定義 | |
| ///////and then  typedef vluint32\_t IData; ///< Verilated pack data, 17-32 bits  **Verilog unsigned integer (?** | |

一、repeat.cpp檔中Vcount\_\_repeat\_\_::\_combo\_\_TOP\_\_1邏輯部分

vlTOPp->NUMBER = 0U;

if ((0x8000U & (IData)(vlTOPp->I))) { //全部為0 //if I

vlTOPp->NUMBER = (0x1fU & ((IData)(1U) + (IData)(vlTOPp->NUMBER)));

} //跳過

if ((1U & (IData)(vlTOPp->I))) { //意思應該是0001，叫出I[0]

vlTOPp->NUMBER = (0x1fU & ((IData)(1U) + (IData)(vlTOPp->NUMBER)));

}

if ((2U & (IData)(vlTOPp->I))) { //0010

vlTOPp->NUMBER = (0x1fU & ((IData)(1U) + (IData)(vlTOPp->NUMBER)));

}

……

說明:

先宣告NUMBER = 0

沒有宣告J，因為不需要(?

開始repeat:

…重複16次，(因為length=16)，從I的第一個bit開始確認是否為1或0，2,3,4,5,6…15，是的話就加一

vlTOPp->NUMBER = (0x1fU & ((IData)(1U) + (IData)(vlTOPp->NUMBER)));

就是NUMBER = NUMBER + 1;

疑問: j = LENGTH-1; 所以是從15開始加1上去?是I[15]?

二、VHW1.cpp說明

先存成一個表，之後再處理處理MuxOut

上次說的DCBASS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| D | C | B | A | Sel[1] | Sel[0] |
| 0 | 1 | 1 | 0 | 1 | 1 |

可能case太多，採用窮舉法

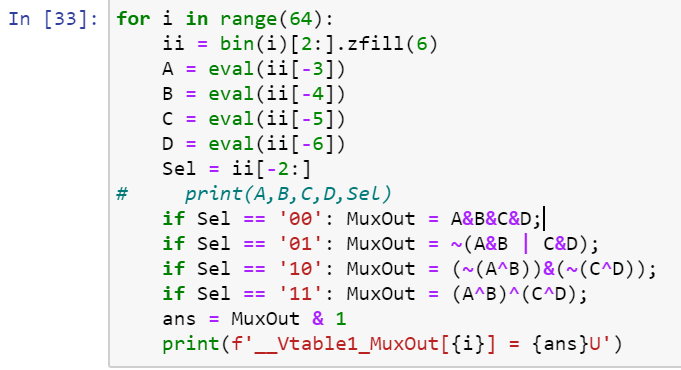
驗證:

(如上:011011，Sel = 2b11，

(0^1)^(1^0) = 0



|  |  |
| --- | --- |
| python | cpp |
| \_\_Vtable1\_MuxOut[0] = 0U  \_\_Vtable1\_MuxOut[1] = 1U  \_\_Vtable1\_MuxOut[2] = 1U  \_\_Vtable1\_MuxOut[3] = 0U  \_\_Vtable1\_MuxOut[4] = 0U  \_\_Vtable1\_MuxOut[5] = 1U  \_\_Vtable1\_MuxOut[6] = 0U  \_\_Vtable1\_MuxOut[7] = 1U  \_\_Vtable1\_MuxOut[8] = 0U  \_\_Vtable1\_MuxOut[9] = 1U  \_\_Vtable1\_MuxOut[10] = 0U  \_\_Vtable1\_MuxOut[11] = 1U  \_\_Vtable1\_MuxOut[12] = 0U  \_\_Vtable1\_MuxOut[13] = 0U  \_\_Vtable1\_MuxOut[14] = 1U  \_\_Vtable1\_MuxOut[15] = 0U  \_\_Vtable1\_MuxOut[16] = 0U  \_\_Vtable1\_MuxOut[17] = 1U  \_\_Vtable1\_MuxOut[18] = 0U  \_\_Vtable1\_MuxOut[19] = 1U  \_\_Vtable1\_MuxOut[20] = 0U  \_\_Vtable1\_MuxOut[21] = 1U  \_\_Vtable1\_MuxOut[22] = 0U  \_\_Vtable1\_MuxOut[23] = 0U  \_\_Vtable1\_MuxOut[24] = 0U  \_\_Vtable1\_MuxOut[25] = 1U  \_\_Vtable1\_MuxOut[26] = 0U  \_\_Vtable1\_MuxOut[27] = 0U  \_\_Vtable1\_MuxOut[28] = 0U  \_\_Vtable1\_MuxOut[29] = 0U  \_\_Vtable1\_MuxOut[30] = 0U  \_\_Vtable1\_MuxOut[31] = 1U  \_\_Vtable1\_MuxOut[32] = 0U  \_\_Vtable1\_MuxOut[33] = 1U  \_\_Vtable1\_MuxOut[34] = 0U  \_\_Vtable1\_MuxOut[35] = 1U  \_\_Vtable1\_MuxOut[36] = 0U  \_\_Vtable1\_MuxOut[37] = 1U  \_\_Vtable1\_MuxOut[38] = 0U  \_\_Vtable1\_MuxOut[39] = 0U  \_\_Vtable1\_MuxOut[40] = 0U  \_\_Vtable1\_MuxOut[41] = 1U  \_\_Vtable1\_MuxOut[42] = 0U  \_\_Vtable1\_MuxOut[43] = 0U  \_\_Vtable1\_MuxOut[44] = 0U  \_\_Vtable1\_MuxOut[45] = 0U  \_\_Vtable1\_MuxOut[46] = 0U  \_\_Vtable1\_MuxOut[47] = 1U  \_\_Vtable1\_MuxOut[48] = 0U  \_\_Vtable1\_MuxOut[49] = 0U  \_\_Vtable1\_MuxOut[50] = 1U  \_\_Vtable1\_MuxOut[51] = 0U  \_\_Vtable1\_MuxOut[52] = 0U  \_\_Vtable1\_MuxOut[53] = 0U  \_\_Vtable1\_MuxOut[54] = 0U  \_\_Vtable1\_MuxOut[55] = 1U  \_\_Vtable1\_MuxOut[56] = 0U  \_\_Vtable1\_MuxOut[57] = 0U  \_\_Vtable1\_MuxOut[58] = 0U  \_\_Vtable1\_MuxOut[59] = 1U  \_\_Vtable1\_MuxOut[60] = 1U  \_\_Vtable1\_MuxOut[61] = 0U  \_\_Vtable1\_MuxOut[62] = 1U  \_\_Vtable1\_MuxOut[63] = 0U | \_\_Vtable1\_MuxOut[0] = 0U;  \_\_Vtable1\_MuxOut[1] = 1U;  \_\_Vtable1\_MuxOut[2] = 1U;  \_\_Vtable1\_MuxOut[3] = 0U;  \_\_Vtable1\_MuxOut[4] = 0U;  \_\_Vtable1\_MuxOut[5] = 1U;  \_\_Vtable1\_MuxOut[6] = 0U;  \_\_Vtable1\_MuxOut[7] = 1U;  \_\_Vtable1\_MuxOut[8] = 0U;  \_\_Vtable1\_MuxOut[9] = 1U;  \_\_Vtable1\_MuxOut[10] = 0U;  \_\_Vtable1\_MuxOut[11] = 1U;  \_\_Vtable1\_MuxOut[12] = 0U;  \_\_Vtable1\_MuxOut[13] = 0U;  \_\_Vtable1\_MuxOut[14] = 1U;  \_\_Vtable1\_MuxOut[15] = 0U;  \_\_Vtable1\_MuxOut[16] = 0U;  \_\_Vtable1\_MuxOut[17] = 1U;  \_\_Vtable1\_MuxOut[18] = 0U;  \_\_Vtable1\_MuxOut[19] = 1U;  \_\_Vtable1\_MuxOut[20] = 0U;  \_\_Vtable1\_MuxOut[21] = 1U;  \_\_Vtable1\_MuxOut[22] = 0U;  \_\_Vtable1\_MuxOut[23] = 0U;  \_\_Vtable1\_MuxOut[24] = 0U;  \_\_Vtable1\_MuxOut[25] = 1U;  \_\_Vtable1\_MuxOut[26] = 0U;  \_\_Vtable1\_MuxOut[27] = 0U;  \_\_Vtable1\_MuxOut[28] = 0U;  \_\_Vtable1\_MuxOut[29] = 0U;  \_\_Vtable1\_MuxOut[30] = 0U;  \_\_Vtable1\_MuxOut[31] = 1U;  \_\_Vtable1\_MuxOut[32] = 0U;  \_\_Vtable1\_MuxOut[33] = 1U;  \_\_Vtable1\_MuxOut[34] = 0U;  \_\_Vtable1\_MuxOut[35] = 1U;  \_\_Vtable1\_MuxOut[36] = 0U;  \_\_Vtable1\_MuxOut[37] = 1U;  \_\_Vtable1\_MuxOut[38] = 0U;  \_\_Vtable1\_MuxOut[39] = 0U;  \_\_Vtable1\_MuxOut[40] = 0U;  \_\_Vtable1\_MuxOut[41] = 1U;  \_\_Vtable1\_MuxOut[42] = 0U;  \_\_Vtable1\_MuxOut[43] = 0U;  \_\_Vtable1\_MuxOut[44] = 0U;  \_\_Vtable1\_MuxOut[45] = 0U;  \_\_Vtable1\_MuxOut[46] = 0U;  \_\_Vtable1\_MuxOut[47] = 1U;  \_\_Vtable1\_MuxOut[48] = 0U;  \_\_Vtable1\_MuxOut[49] = 0U;  \_\_Vtable1\_MuxOut[50] = 1U;  \_\_Vtable1\_MuxOut[51] = 0U;  \_\_Vtable1\_MuxOut[52] = 0U;  \_\_Vtable1\_MuxOut[53] = 0U;  \_\_Vtable1\_MuxOut[54] = 0U;  \_\_Vtable1\_MuxOut[55] = 1U;  \_\_Vtable1\_MuxOut[56] = 0U;  \_\_Vtable1\_MuxOut[57] = 0U;  \_\_Vtable1\_MuxOut[58] = 0U;  \_\_Vtable1\_MuxOut[59] = 1U;  \_\_Vtable1\_MuxOut[60] = 1U;  \_\_Vtable1\_MuxOut[61] = 0U;  \_\_Vtable1\_MuxOut[62] = 1U;  \_\_Vtable1\_MuxOut[63] = 0U; |



參考網站:

<https://www.veripool.org/projects/verilator/wiki/Manual-verilator>

問題<https://www.veripool.org/boards/2/topics/2747-Verilator-Wait-on-rising-edge-from-c->

<http://manpages.ubuntu.com/manpages/bionic/man1/verilator.1.html>