```
/* vpiInverter.c

Author      A.D.Beckett
Group       Structured Custom, Cadence Design Systems Ltd.
Machine     SUN
Date        Jul 22, 1998
Modified
By

Uses VPI callbacks to implement a simple inverter

If compiled with -DWITHDELAY, includes a 5 unit delay
on output of inverter.

Inverter is "instantiated" in design by invoking:

$myinverter(outputReg,inputNet);

Note that this code doesn't really do any argument checking.

************************************************

*/

#include   <stdio.h>
#include   <stdlib.h>
#include   " vpi_user.h"
#include   " veriuser.h"

/*************************************************************
*                                                            *
*                static void check_error(where)              *
*                         char *where;                       *
*                                                            *
*  General purpose function for checking to see whether there  *
*     has been a vpi error, and printing it if there has.    *
* The argument is a string to say where the problem occurred.  *
*                                                            *
*************************************************************/

static   void  check_error(where)
char  *where;
{
    s_vpi_error_info error_info;

     if (vpi_chk_error(&error_info)) {
        vpi_printf("    An error occurred in %s\n",where);
        vpi_printf("    Level: %d state: %d %s%s%s%s\n",
                error_info.level, error_info.state,
                "code:",error_info.code,"      product:", error_info.product);
        vpi_printf("    Message: %s\n",error_info.message);
        if (error_info.file)
                vpi_printf("    LineNo: %d File: %s\n",
                    error_info.line, error_info.file);
        }
}


/*************************************************************
*                                                            *
*                  vpiHandle call_vpi_scan(itr)              *
*                         vpiHandle itr;                     *
*                                                            *
*           Error checking wrapper around vpi_scan()         *
*                                                            *
*************************************************************/
```

```
vpiHandle call_vpi_scan(itr)
vpiHandle itr;
{
    vpiHandle retVal = vpi_scan(itr);
    check_error("     vpi_scan");
     return (retVal);
}

/*************************************************************
*                                                            *
*             vpiHandle call_vpi_iterate(type,ref)           *
*                          int type;                         *
*                        vpiHandle ref;                      *
*                                                            *
*          Error checking wrapper around vpi_iterate.        *
*                                                            *
*************************************************************/

vpiHandle call_vpi_iterate(type,ref)
int  type;
vpiHandle ref;
{
    vpiHandle retVal = vpi_iterate(type,ref);
    check_error("     vpi_iterate");
     return (retVal);
}

int  vpiInverterCB(callbackData)
p_cb_data callbackData;
{
    s_vpi_value outVal;
#ifdef  WITHDELAY
    s_vpi_time outDelay;
#endif

    outVal.format=vpiBinStrVal;
     /* if the input is not "0" */
     if (strcmp(callbackData->value->value.str,"            0"))
        outVal.value.str="     0";
     else
        outVal.value.str="     1";

#ifdef  WITHDELAY
    outDelay.type=vpiScaledRealTime;
    outDelay.real=5.0;
    vpi_put_value(*((vpiHandle *)callbackData->user_data),
        &outVal, &outDelay, vpiInertialDelay);
#else
    vpi_put_value(*((vpiHandle *)callbackData->user_data),
        &outVal,    NULL , vpiNoDelay);
#endif
}

/*************************************************************
*                                                            *
*                      int vpiInverter()                     *
*                                                            *
*       Set up the value change callback, monitoring the     *
*     input node. Note that this doesn't really do enough    *
*                           checking!                        *
*                                                            *
*************************************************************/

int  vpiInverter()
{
```

```
    vpiHandle taskHandle,argIter,inHandle,outHandle;
    s_cb_data callbackData;

    taskHandle = vpi_handle(vpiSysTfCall,           NULL );
    check_error("     vpi_handle");

     /* process the arguments */
    argIter=call_vpi_iterate(vpiArgument,taskHandle);
     if (argIter) {
        /* get the first argument - the output register */
        outHandle=call_vpi_scan(argIter);
        if (outHandle) {
             /* get the second argument - the input net */
            inHandle=call_vpi_scan(argIter);
             if (inHandle) {
                callbackData.reason=cbValueChange;
                callbackData.obj=inHandle;
                /* don't need the time, so set to NULL */
                callbackData.time=     NULL ;
                /* allocate the space for the value, so the callback can see
                   the current value - represented as a string */
                callbackData.value=(p_vpi_value) malloc(           sizeof (s_vpi_va
                callbackData.value->format=vpiBinStrVal;
                /* store the output handle in an area pointed to by the
                   user_data. Could probably store it directly in the
                   user_data, but that's assuming that a char * is the
                   same size as a vpiHandle */
                callbackData.user_data=(       char  *) malloc(   sizeof (vpiHandle)
                *((vpiHandle *) callbackData.user_data)=outHandle;
                /* the callback function */
                callbackData.cb_rtn=vpiInverterCB;
                vpi_register_cb(&callbackData);
                /* since we've not iterated to the end, free the
                   iterator */
                vpi_free_object(argIter);
                }
            }
        }

     return  0;
}

/*************************************************************
*                                                            *
*                   Register the system task                 *
*                                                            *
*************************************************************/

void  register_my_systfs()
{
  s_vpi_systf_data task_data_s;
  p_vpi_systf_data task_data_p = &task_data_s;

  task_data_p->type = vpiSysTask;
  task_data_p->tfname = "       $myinverter";
  task_data_p->calltf = (       int (*)()) vpiInverter;
  task_data_p->compiletf = 0L;
  vpi_register_systf(task_data_p);
}

/* Enter register_my_systfs() in the startup array. */
void  (*vlog_startup_routines[])() = {register_my_systfs, 0};
```