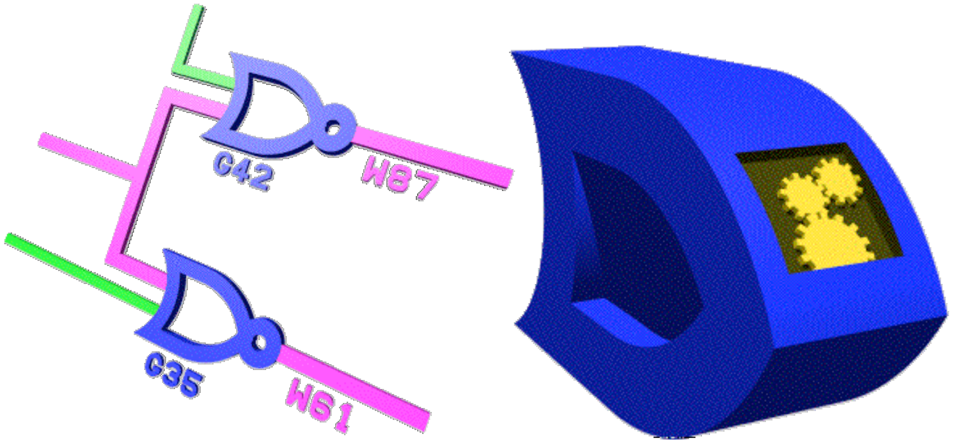
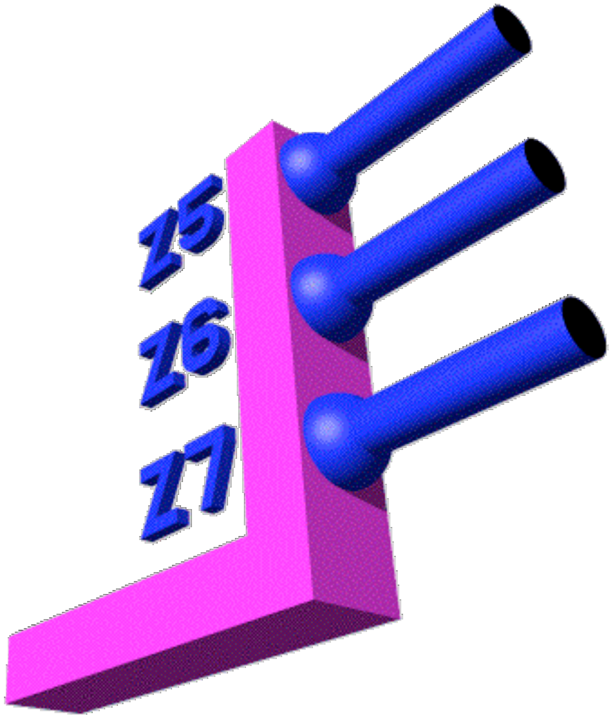


Creating Virtual Peripheral Devices in a Digital Circuit Simulator Using Tcl/Tk



Jeffery P. Hansen

Institute for Complex Engineered Systems
Carnegie Mellon University

October 26, 2005

Outline

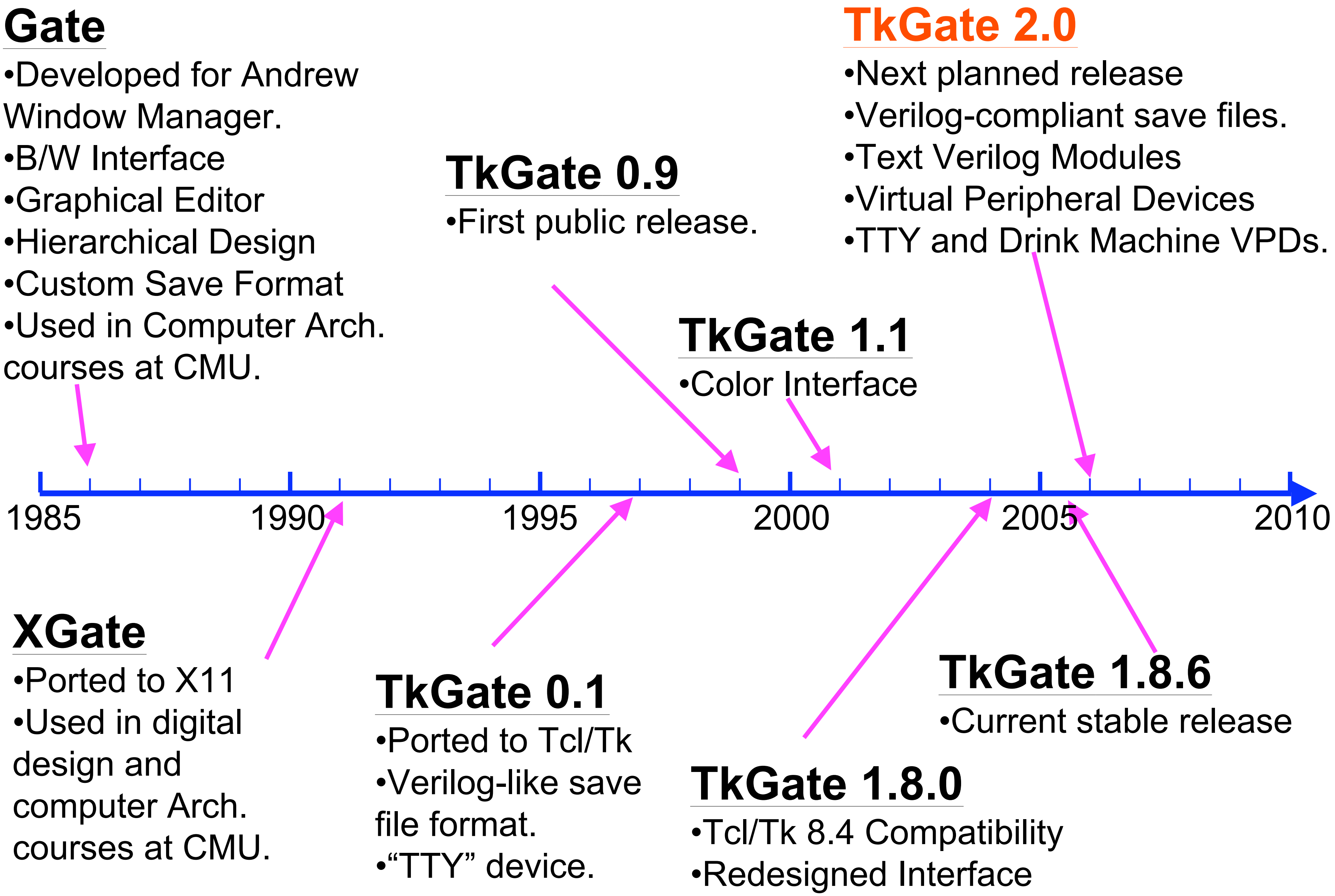
- Overview of TkGate
- Overview of Verilog
- Virtual Peripheral Devices (VPDs)
- VPD Tcl-Side
- VPD Verilog-Side
- Implementation
- Conclusions

TkGate 2.0 Overview

- **Interface Features**
  - Hybrid Tcl/Tk and Xlib/C based interface.
    - Interface comprised of about 64K lines of C, 25K lines of Tcl/Tk.
    - Separate simulator is about 31K lines of C.
  - Multi-lingual interface with support for Catalan, English, French, German, Japanese, Spanish, Czech and Welsh.
  - Design entry of both graphical and text/Verilog modules.
  - User-defined device symbols.
  - Support tools include a microcode/macrocode compiler.

- **Simulation Features**
  - Dynamically loadable Verilog scripts.
  - Graphical display of simulation results.
  - Breakpoints, single-step and clock-step simulator control.
  - Interactive Virtual Peripheral Devices (VPD)
    - Uses Tcl and Verilog to design interactive components.
    - Tcl-Side and Verilog-side APIs facilitate communication.

History of TkGate



Gate
- Developed for Andrew Window Manager.
- B/W Interface
- Graphical Editor
- Hierarchical Design
- Custom Save Format
- Used in Computer Arch. courses at CMU.

XGate
- Ported to X11
- Used in digital design and computer Arch. courses at CMU.

TkGate 0.1
- Ported to Tcl/Tk
- Verilog-like save file format.
- “TTY” device.

TkGate 0.9
- First public release.

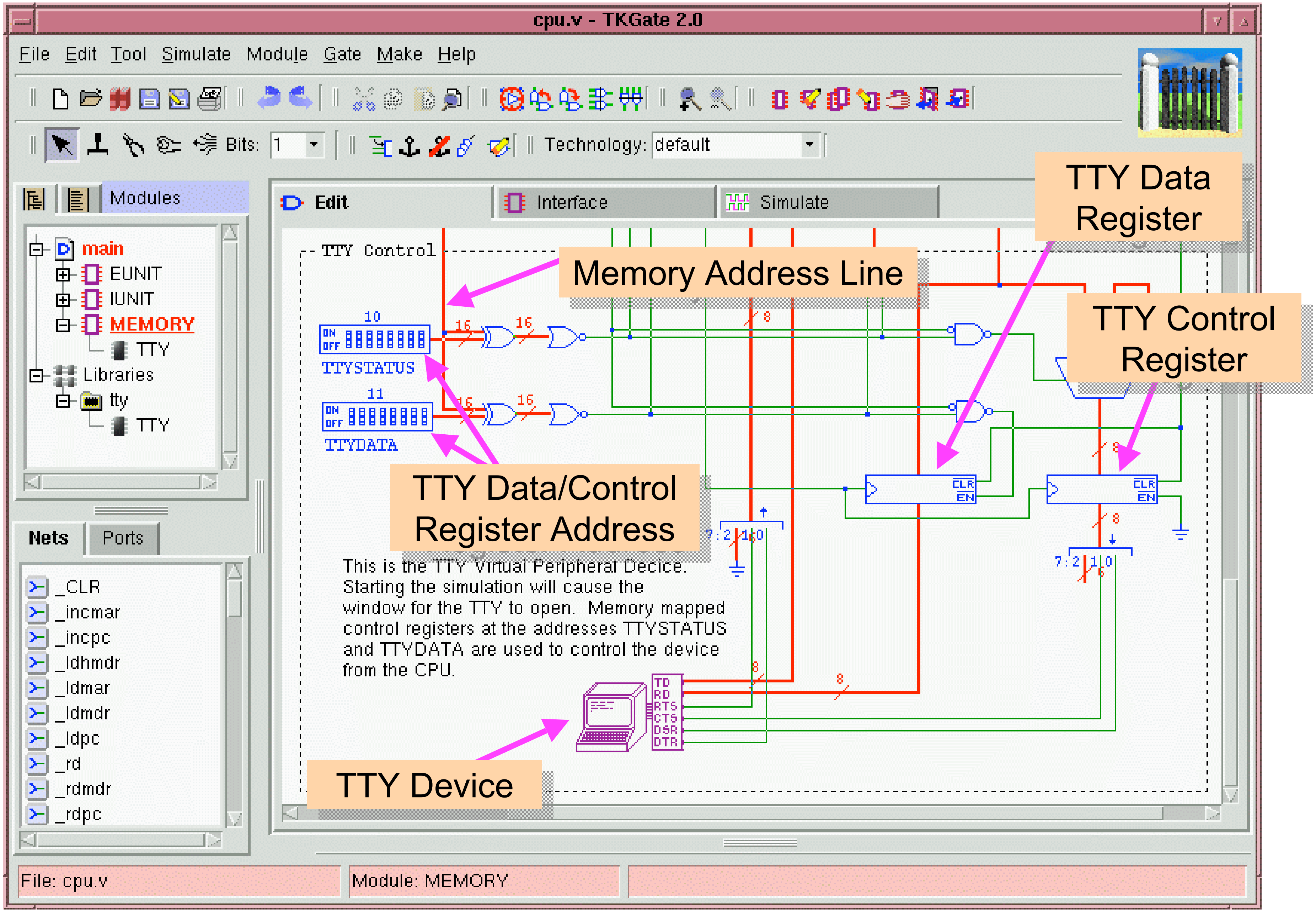
TkGate 1.1
- Color Interface

TkGate 1.8.0
- Tcl/Tk 8.4 Compatibility
- Redesigned Interface

TkGate 1.8.6
- Current stable release

TkGate 2.0
- Next planned release
- Verilog-compliant save files.
- Text Verilog Modules
- Virtual Peripheral Devices
- TTY and Drink Machine VPDs.

TkGate Main Window



TTY Data Register

Memory Address Line

TTY Control Register

TTY Data/Control Register Address

This is the TTY Virtual Peripheral Decice. Starting the simulation will cause the window for the TTY to open. Memory mapped control registers at the addresses TTYSTATUS and TTYDATA are used to control the device from the CPU.

TTY Device

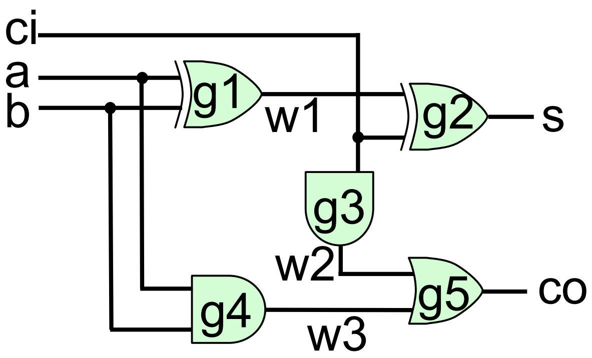
Verilog

- **What is Verilog?**
  - Hardware description language used to describe hardware designs.
  - Widely used industry standard for EDA (Electronic Design Automation) tools.
- **In what types of applications is Verilog used?**
  - Synthesis
  - Simulation
  - Formal verification
- **What do Verilog designs look like?**
  - Modular hierarchical design
  - High degree of parallelism
  - Multiple levels of abstraction including
    - Structural – Module described as collection of connected blocks.
    - Behavioral – Module described using C-like statements.

Verilog Examples

Structural Verilog

```
module add(s,co,a,b,ci)
input a,b,ci;
output s,co;

  xor g1 (w1,a,b);
  xor g2 (s,w1,ci);
  and g3 (w2,w1,ci);
  and g4 (w3,a,b);
  or g5 (co,w2,w3);

endmodule
```

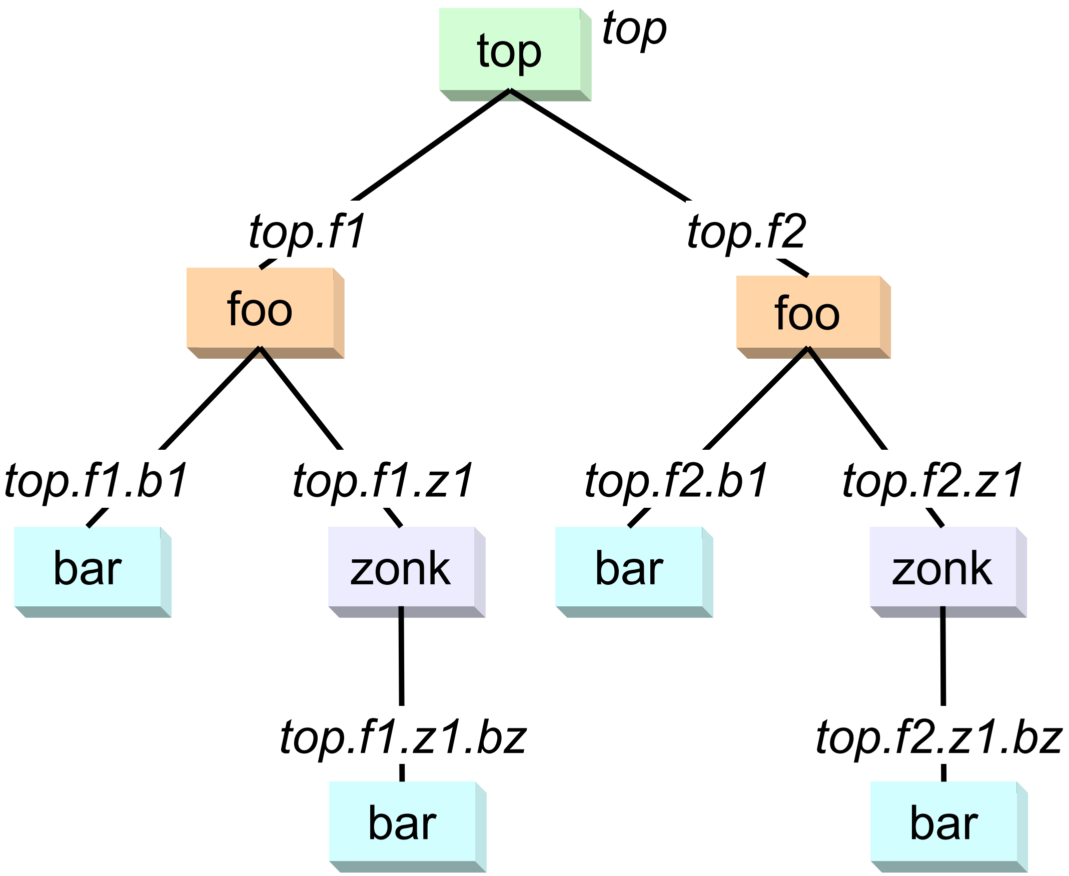


Behavioral Verilog

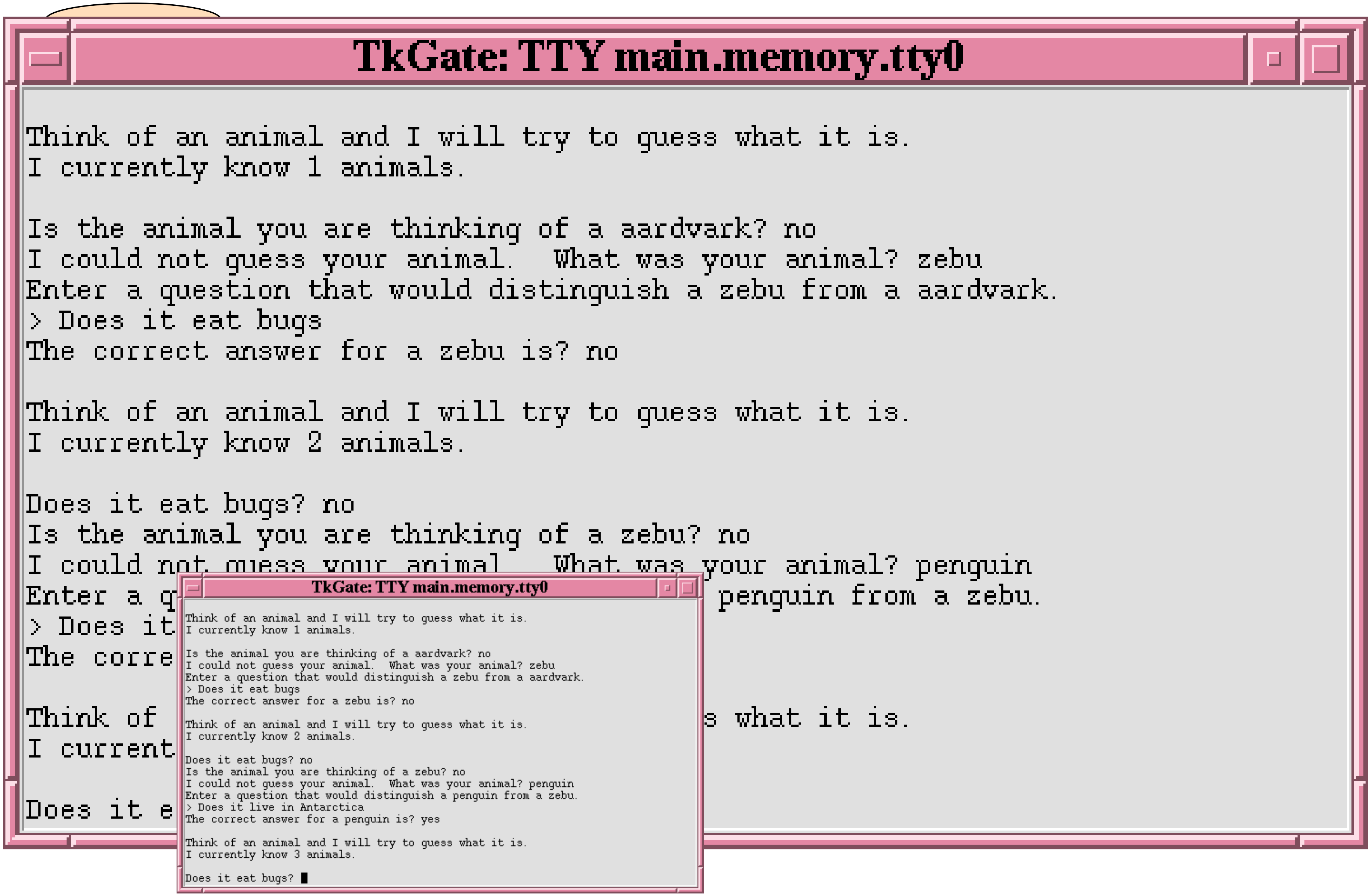
```
module count(state,ck,clr);
output [3:0] state;
reg [3:0] state;
input ck,clr;

  initial
    state = 4'h0;

  always @(posedge ck)
    state = state + 4'h1;

  always @(clr)
    if (clr)
      state = 4'h0;

endmodule
```

Verilog Design Hierarchy

```
module top;
  foo f1;
  foo f2;
endmodule

module foo;
  bar b1;
  zonk z1;
endmodule

module bar;
endmodule

module zonk;
  bar bz;
endmodule
```



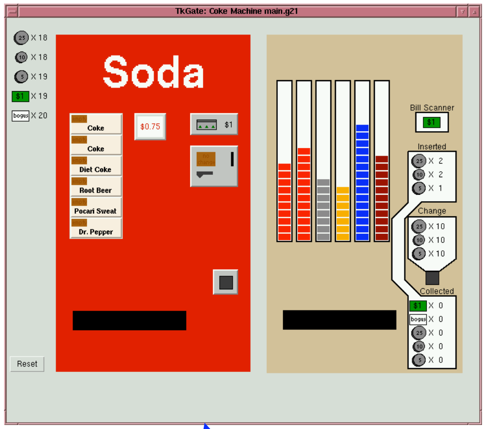
TTY Virtual Peripheral Device

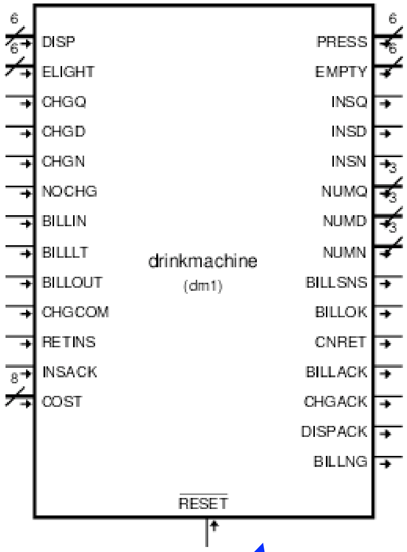


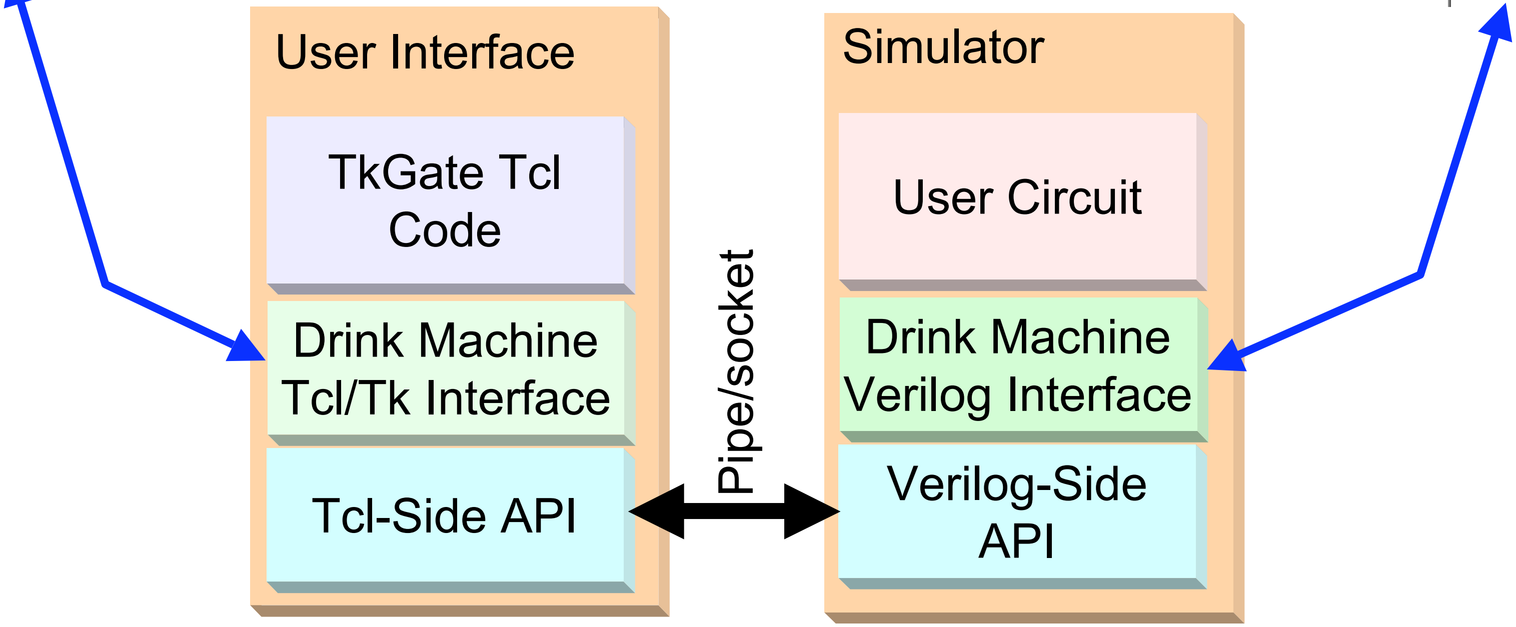
```
Think of an animal and I will try to guess what it is.
I currently know 1 animals.

Is the animal you are thinking of a aardvark? no
I could not guess your animal.  What was your animal? zebu
Enter a question that would distinguish a zebu from a aardvark.
> Does it eat bugs
The correct answer for a zebu is? no

Think of an animal and I will try to guess what it is.
I currently know 2 animals.

Does it eat bugs? no
Is the animal you are thinking of a zebu? no
I could not guess your animal.  What was your animal? penguin
Enter a question that would distinguish a penguin from a zebu.
> Does it live in Antarctica
The correct answer for a penguin is? yes

Think of an animal and I will try to guess what it is.
I currently know 3 animals.

Does it eat bugs? █
```

Drink Machine VPD

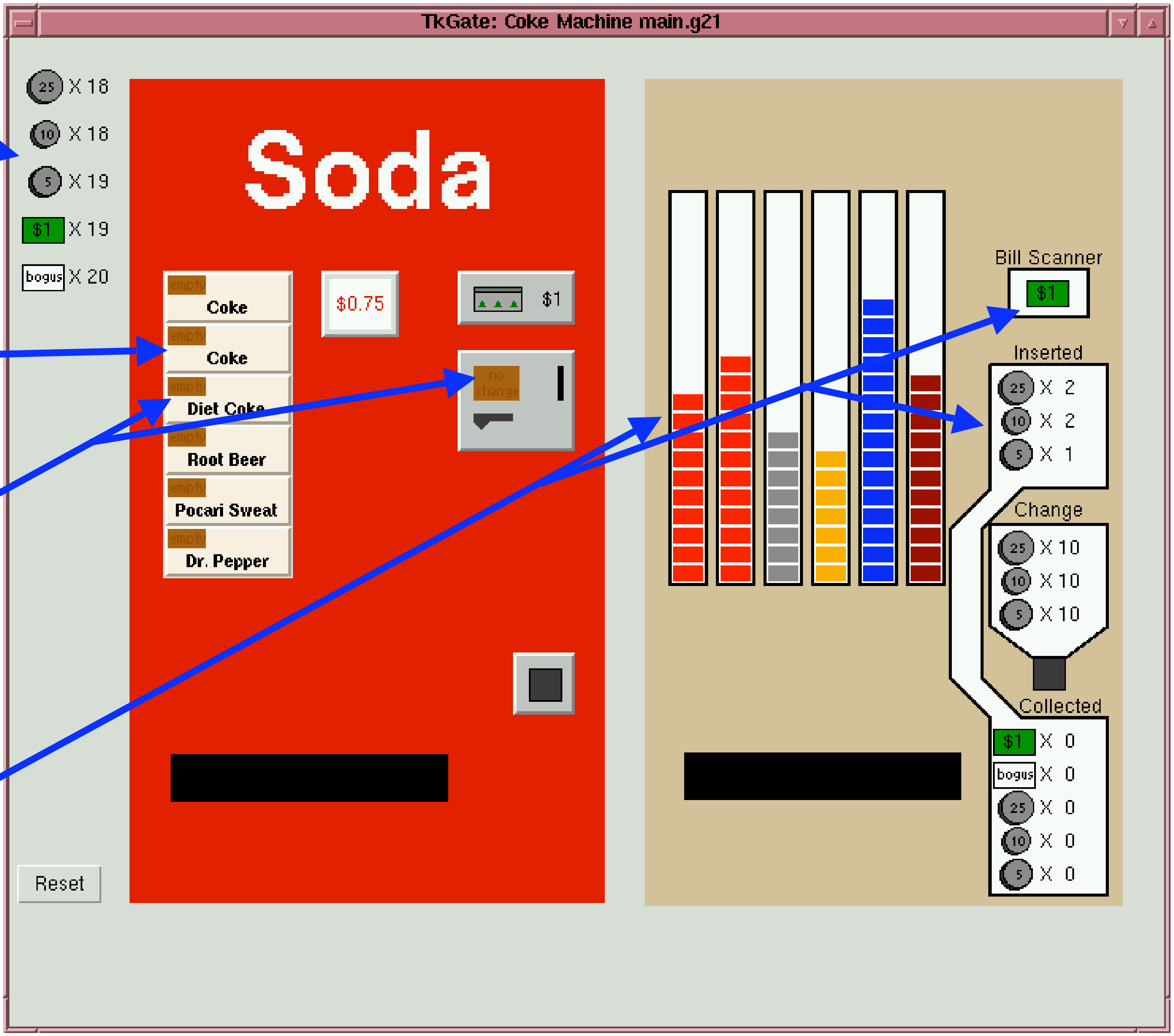
- **VPD Implementer**
  - Writes Tcl script implementing behavior.
  - Writes Verilog stub-module encapsulating interface.
- **VPD Client**
  - Loads library with desired VPD.
  - Creates one or more instances.
  - Creates control logic.







Drink Machine Window



Drag and drop coins/bills to insert into machine.

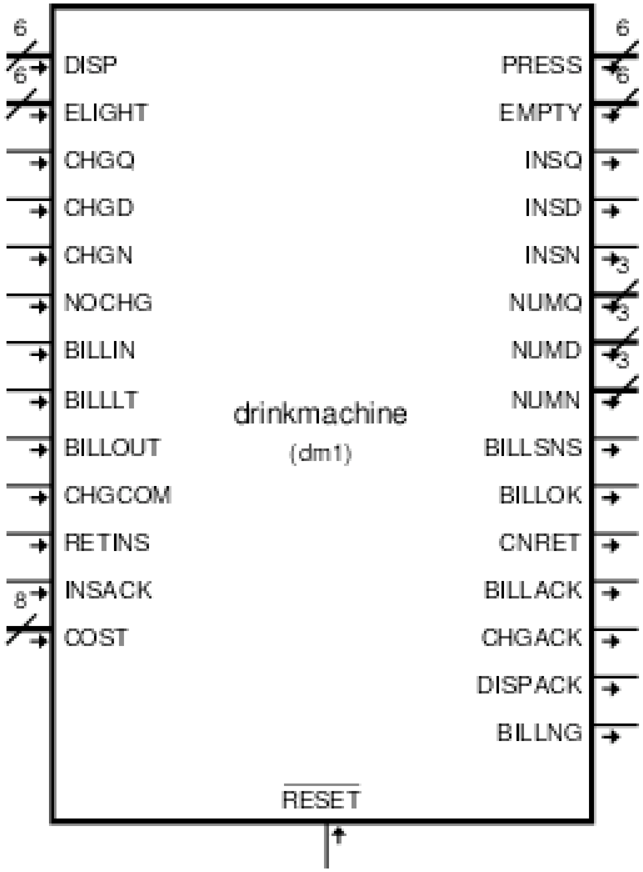
Push buttons to make drink selections.

User circuit can activate “empty” and “no change” lights.

Internal view shows inserted coins/bills and drink columns

Including a VPD in the User’s Design

- Users interact with VPD as a normal module.
  - Create one or more instances in user module.
  - Interact though ports on VPD module.
- Users can create multiple instances of a VPD.
  - Each VPD is assigned a unique instance name from the Verilog hierarchy.
  - Verilog instance name is used as VPD instance identifier in the Tcl script.

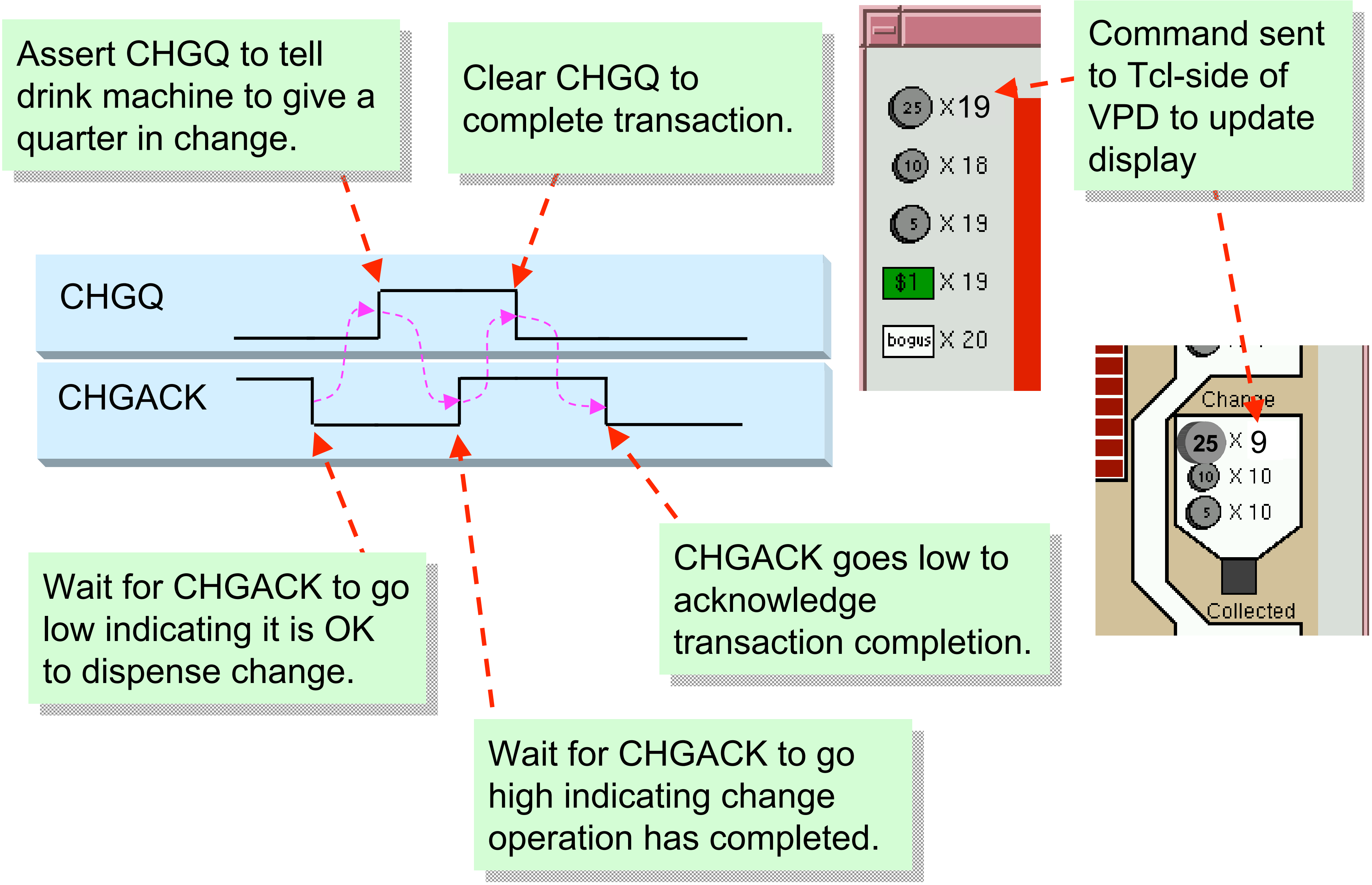


```
module top;
  wire [5:0] PRESS1, PRESS2;
  wire NOCHG1, NOCHG2;
  ...
  drinkmachine dm1(..., PRESS1, NOCHG1, ...);
  drinkmachine dm2(..., PRESS2, NOCHG2, ...);
  ...
endmodule
```

top.dm1

top.dm2

Example Interaction



Assert CHGQ to tell drink machine to give a quarter in change.

Clear CHGQ to complete transaction.

Command sent to Tcl-side of VPD to update display

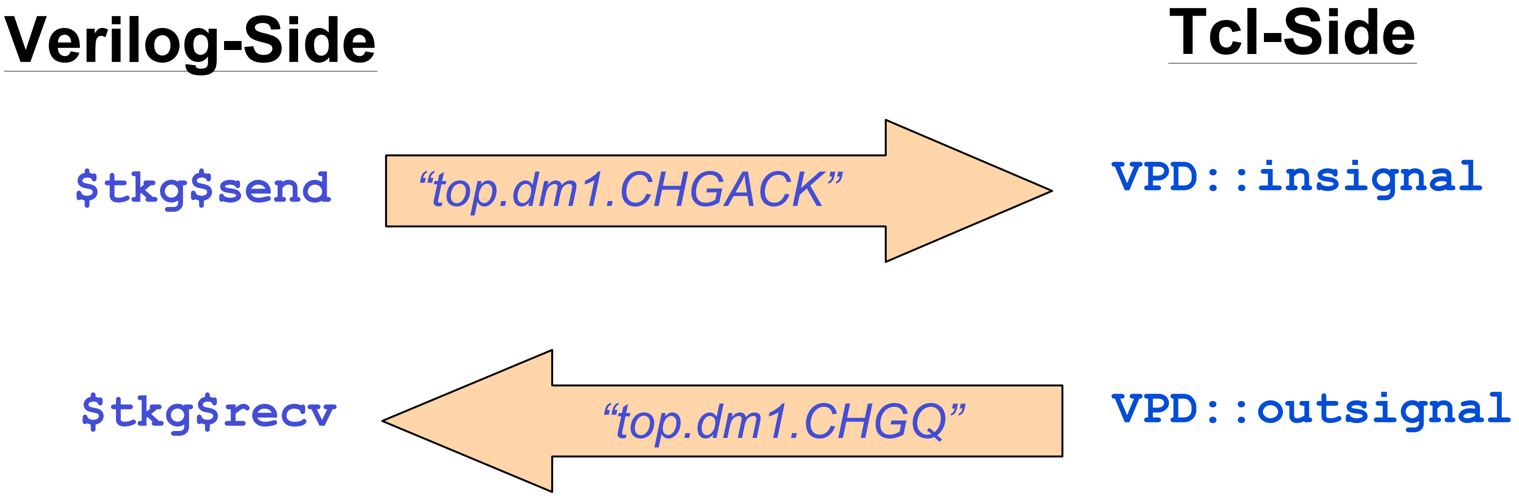
Wait for CHGACK to go low indicating it is OK to dispense change.

CHGACK goes low to acknowledge transaction completion.

Wait for CHGACK to go high indicating change operation has completed.

Named Channels

- Created dynamically as needed.
- Implemented as queue that can be accessed from both Tcl and Verilog.
- Objects on queue are arbitrary bit-sized values.
- One named channel used for each signal passing between Tcl-Side and Verilog-Side of implementation.



Verilog-Side

Tcl-Side

`$tkg$send` *"top.dm1.CHGACK"* `VPD::insignal`

`$tkg$recv` *"top.dm1.CHGQ"* `VPD::outsignal`

Tcl-Side API Highlights

`VPD::register` *name*

Register *name* as a virtual peripheral device. Registered VPDs can be posted using the $tkg$post() task from Verilog.

`VPD::newtoplevel -title` *title* **`-shutdowncommand`** *command*

Create a new top-level window for use as a VPD. Top-level windows created with this command will be given the specified title. When simulation mode is terminated, the window will be destroyed and the specified shut-down command will be invoked.

`VPD::insignal` *chan* **`[-variable`** *variable***`]`** **`[-command`** *command***`]`**

Associate an input channel with a Tcl variable and/or command. When data from the simulator is sent on the named channel, the registered Tcl variable will be set and/or the registered command will be called.

`VPD::outsignal` *chan variable*

Associate an output channel with a Tcl variable. When a value is assigned to the registered variable, data will be sent

Verilog-Side API Highlights

`$tkg$exec(`*format*`, `p_1`, ..., `p_n`)`

Construct a Tcl command and execute it. Executed commands are subject to the current security policy.

`$tkg$post(`*vpdname*`, `*instname*`, `*p1*`, ..., `*pn*`)`

Call the "post" procedure for a registered VPD passing the specified argument list. The instance name and one or more optional parameters are passed as arguments to the "*vpdname*`::post`" command.

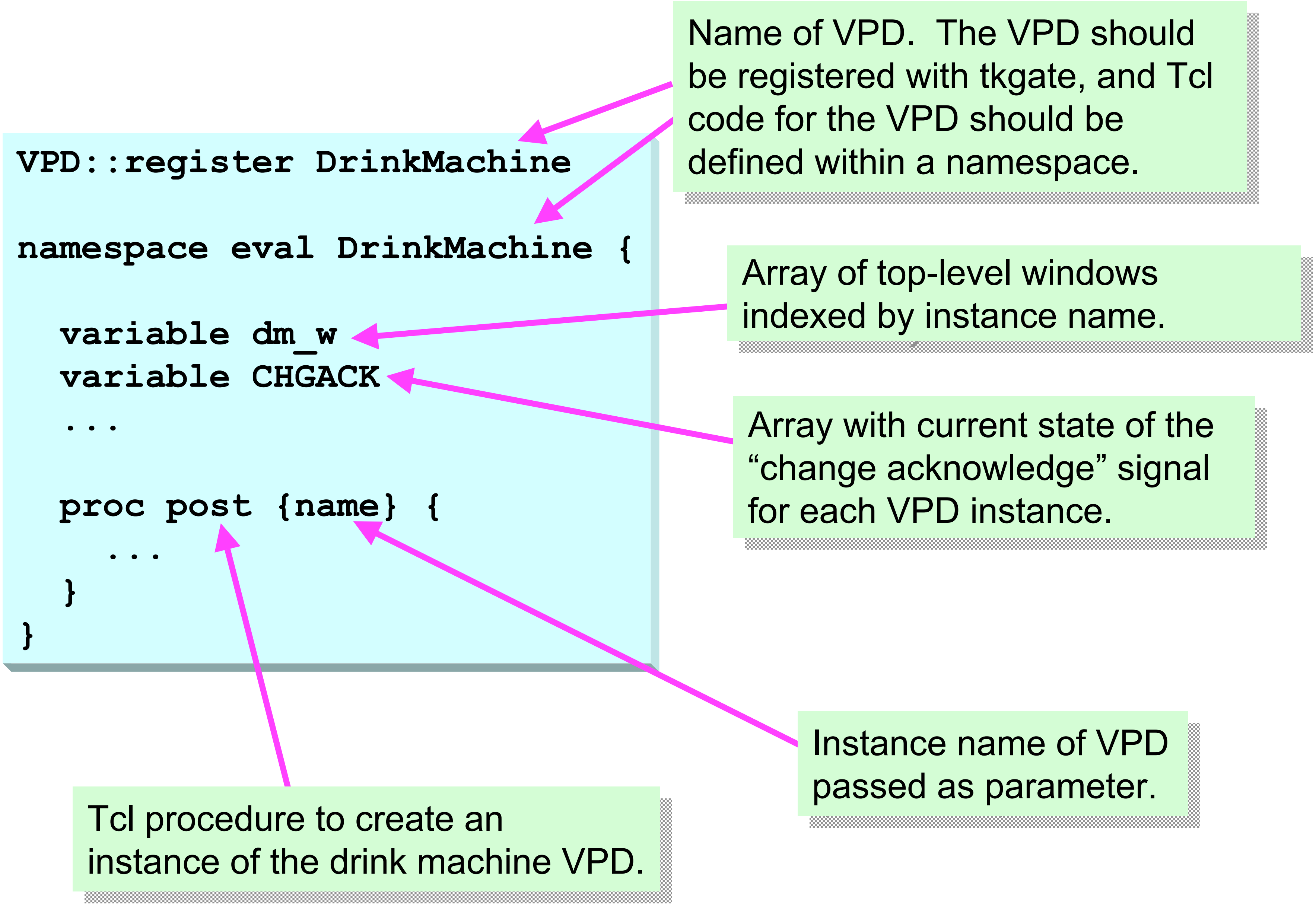
`$tkg$send(`*name*`, `*data*`)`

Send data on a named channel.

`$tkg$recv(`*name*`)`

Receive data on a named channel. Blocks if no data is available in the queue.

Drink Machine VPD Overview



```
VPD::register DrinkMachine

namespace eval DrinkMachine {

  variable dm_w
  variable CHGACK
  ...

  proc post {name} {
    ...
  }
}
```

Name of VPD. The VPD should be registered with tkgate, and Tcl code for the VPD should be defined within a namespace.

Array of top-level windows indexed by instance name.

Array with current state of the “change acknowledge” signal for each VPD instance.

Instance name of VPD passed as parameter.

Tcl procedure to create an instance of the drink machine VPD.

“post” Proceedure (Creating Window)

Variable to store name of window for VPD instance

VPD instance name (e.g., “top.dm1”)

Window for VPD will be labeled “Vending Machine top.dm1”

```
set dm_w($name) [VPD::newtoplevel \
    -title "Vending Machine $name" \
    -shutdowncommand "DrinkMachine::unpost $name"]
```

The “unpost” procedure for this VPD will be invoked when the simulation is terminated.

“post” Procedure (Registering Output Signals)

Register a signal from the Tcl-side to the Verilog-side.

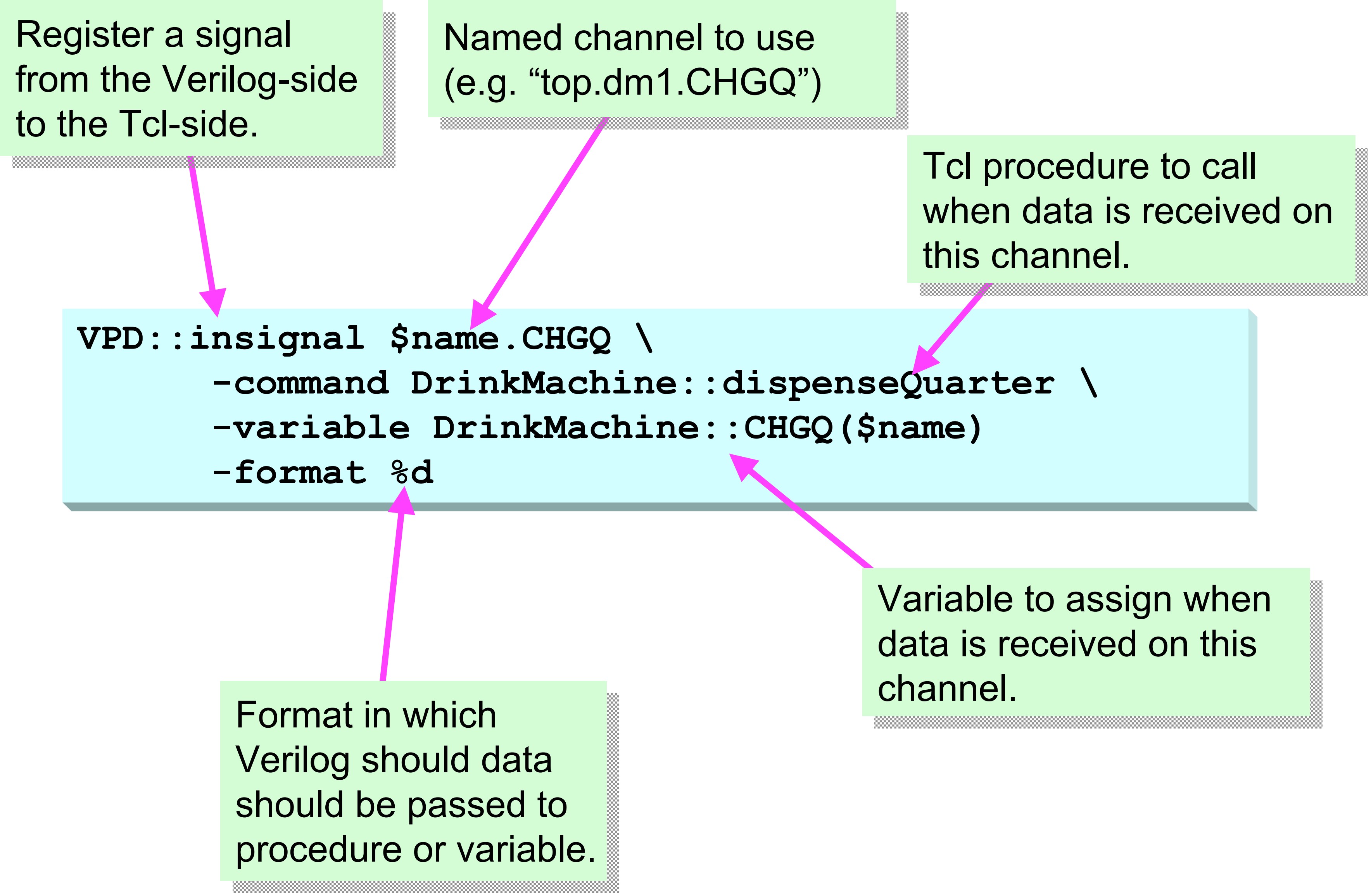
Named channel to use (e.g. “top.dm1.CHGACK”)

Tcl variable to attach to the named channel.

```
VPD::outsignal $name.CHGACK DrinkMachine::CHGACK($name)

set CHGACK($name) 0
```

Send a “0” to the Verilog-side of the implementation.

“post” Procedure (Registering Input Signals)



Register a signal from the Verilog-side to the Tcl-side.

Named channel to use (e.g. “top.dm1.CHGQ”)

Tcl procedure to call when data is received on this channel.

```
VPD::insignal $name.CHGQ \
      -command DrinkMachine::dispenseQuarter \
      -variable DrinkMachine::CHGQ($name)
      -format %d
```

Variable to assign when data is received on this channel.

Format in which Verilog should data should be passed to procedure or variable.

Verilog-Side VPD Definition

Module port declarations

```
module drinkmachine(..., CHGQ, CHGACK, ...);
output CHGACK;
reg CHGACK;
input CHGQ;

  initial
    $tkg$post("DrinkMachine","%m");

  always #10
    CHGACK = $tkg$recv("%m.CHGACK");

  always @ (CHGQ)
    $tkg$send("%m.CHGQ",CHGQ);

  ...
endmodule
```

Port type declarations

Create VPD window

Receive data from Tcl-side

Send data to Tcl-side

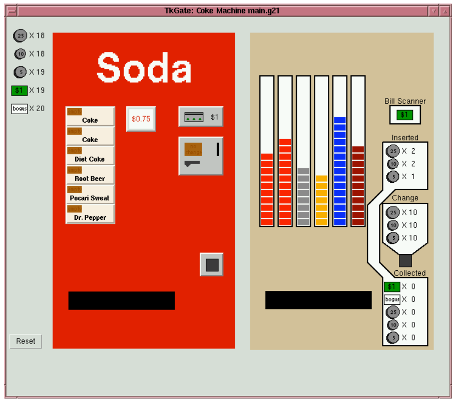
Initialization

Verilog “execute once” statement.

Type name of VPD to post.

Name of VPD instance. “%m” replace with instance name (e.g., “top.dm1”).

```
initial
   $tkg$post("DrinkMachine","%m");
```



VPD::newtoplevel

Simulator

“go *(start simulation)*”

“post DrinkMachine top.dm1”

Interface

Tcl->Verilog Dataflow

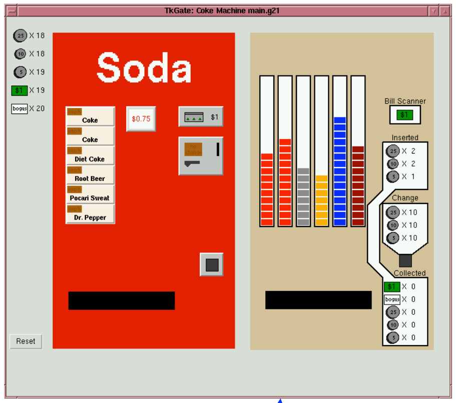
Verilog “infinite loop” statement.

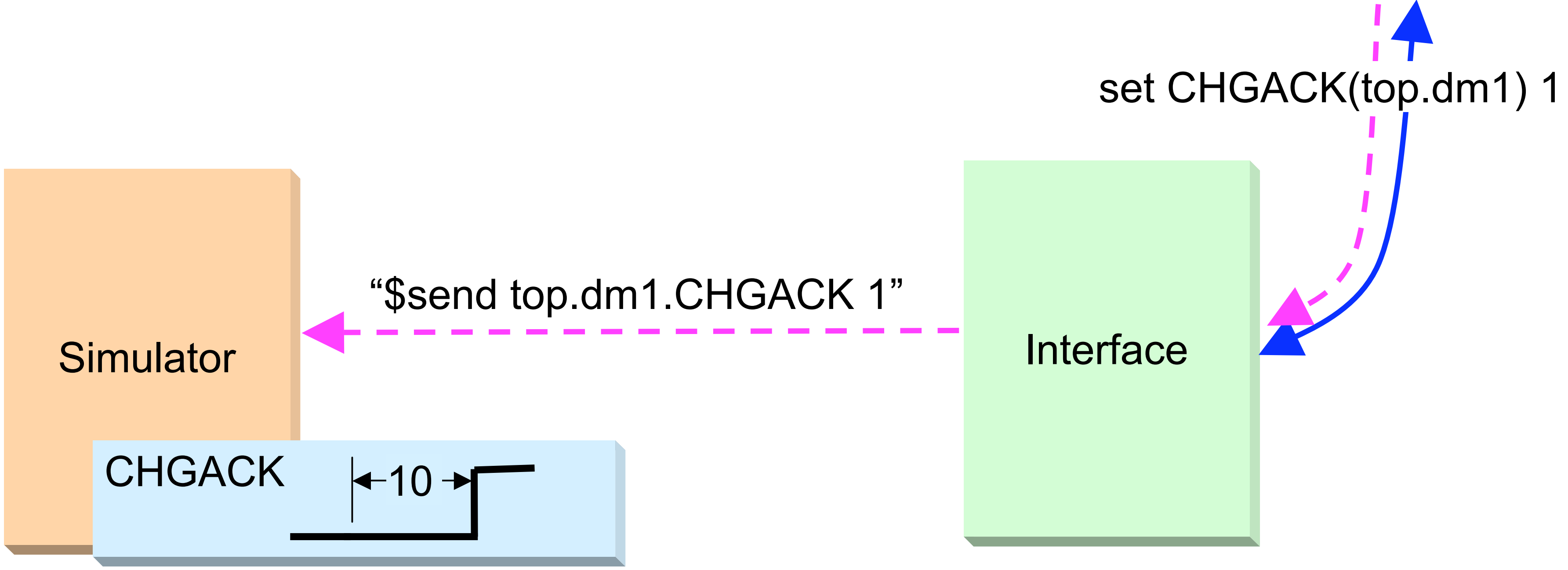
Delay

Verilog signal to update.

Named channel to read.

```
always #10
  CHGACK = $tkg$recv("%m.CHGACK");
```




set CHGACK(top.dm1) 1

“$send top.dm1.CHGACK 1”

Simulator

Interface

CHGACK

10

Verilog->Tcl Dataflow

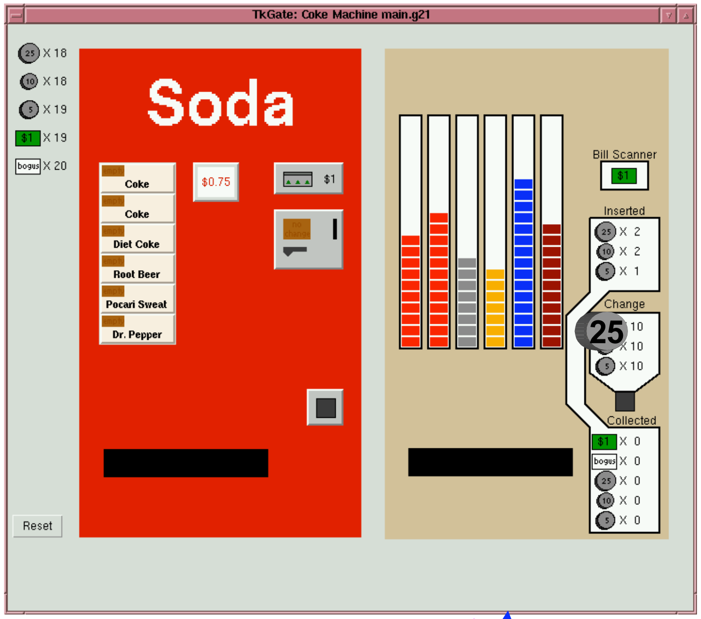
Verilog "infinite loop" statement.

Block until signal changes.

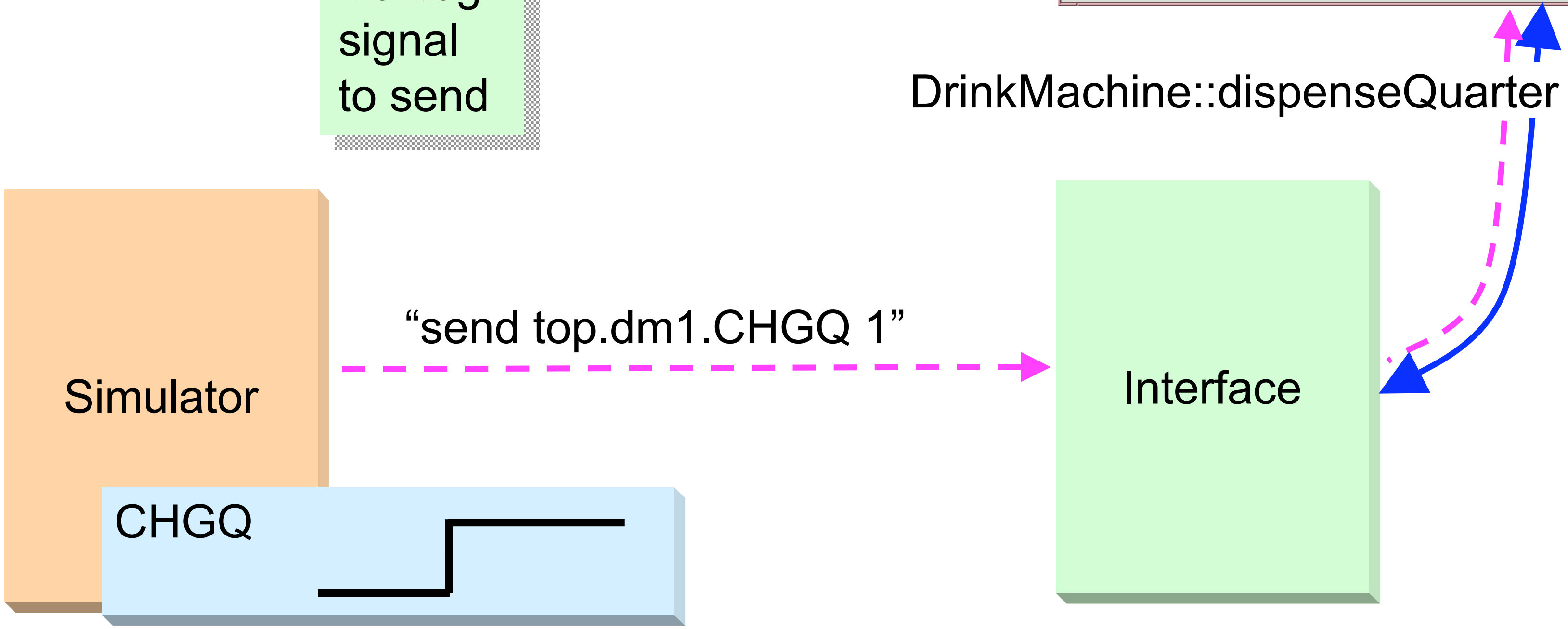
Named channel to use

```
always @(CHGQ)
  $tkg$send("%m.CHGQ",CHGQ);
```

Verilog signal to send

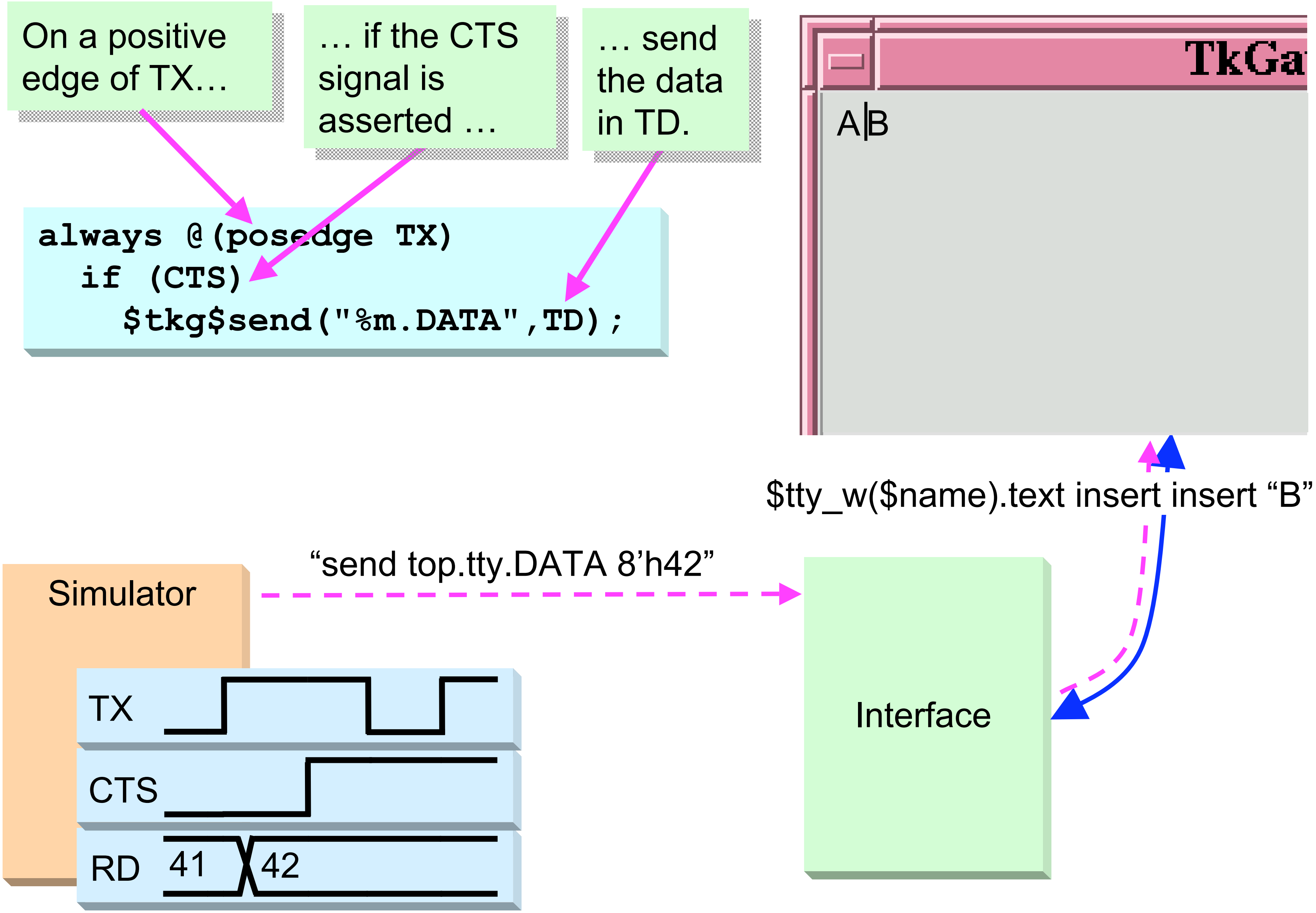


DrinkMachine::dispenseQuarter top.dm1



"send top.dm1.CHGQ 1"

Alternate Dataflow Styles



On a positive edge of TX…

… if the CTS signal is asserted …

… send the data in TD.

```
always @(posedge TX)
   if (CTS)
      $tkg$send("%m.DATA",TD);
```

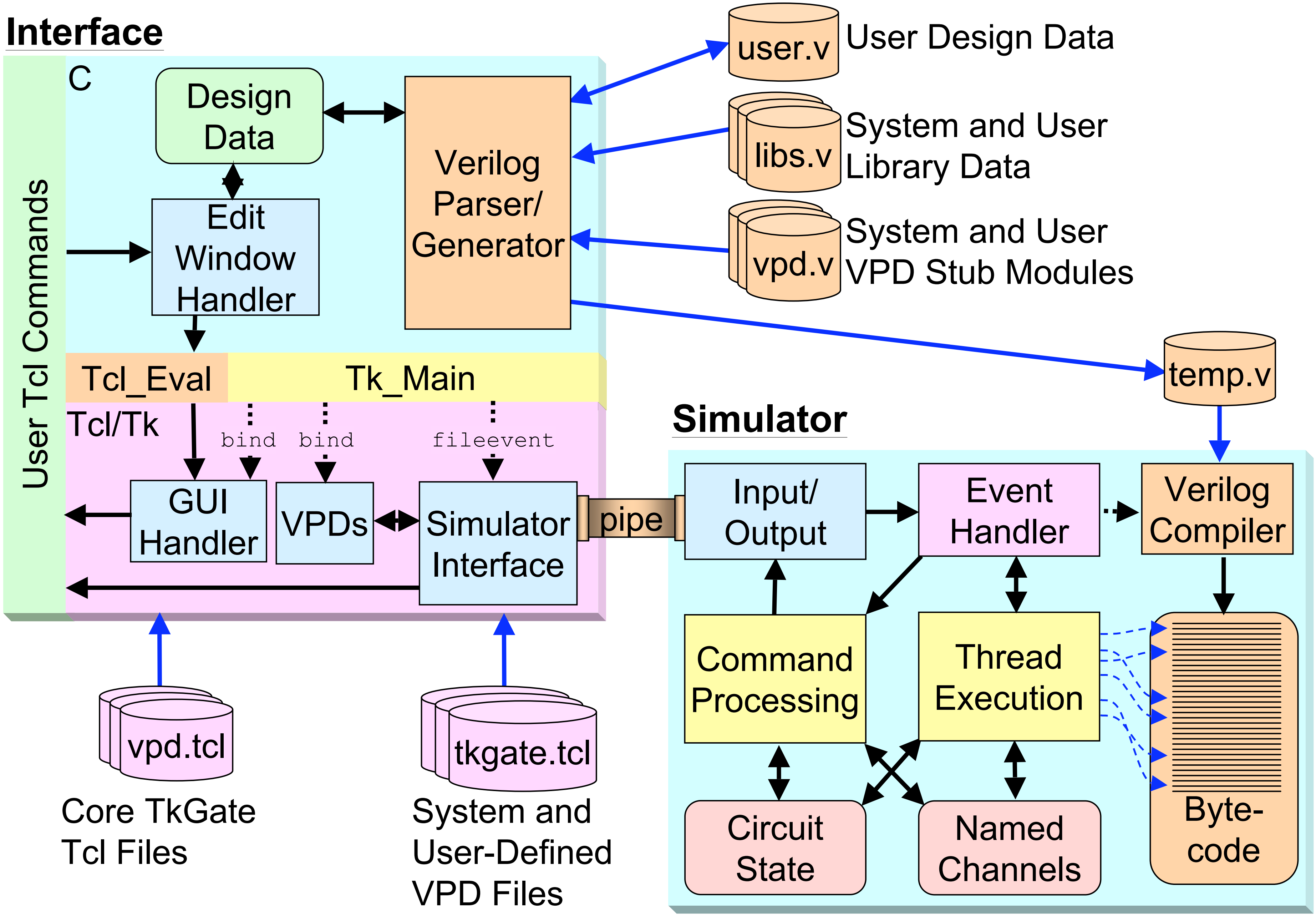
$tty_w($name).text insert insert “B”

“send top.tty.DATA 8’h42”

Simulator

Interface

TkGate Architecture



Interface

User Tcl Commands

C

Design Data

Edit Window Handler

Verilog Parser/ Generator

user.v

User Design Data

libs.v

System and User Library Data

vpd.v

System and User VPD Stub Modules

temp.v

Tcl_Eval

Tk_Main

Tcl/Tk

`bind` `bind` `fileevent`

GUI Handler

VPDs

Simulator Interface

pipe

vpd.tcl

Core TkGate Tcl Files

tkgate.tcl

System and User-Defined VPD Files

Simulator

Input/ Output

Event Handler

Verilog Compiler

Command Processing

Thread Execution

Byte-code

Circuit State

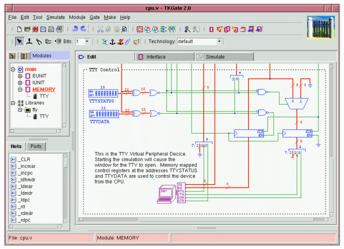
Named Channels

Conclusion

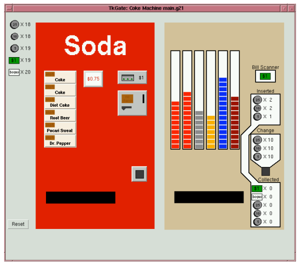
- Virtual Peripheral Devices
  - Mechanism to create interactive devices.
  - Devices appear as ordinary module to user design.
  - Signals on Verilog module translated into Tcl commands in interface.
  - Named channels used to communicate between Tcl and Verilog side.
- Implementation
  - Tcl/Tk Graphical Interface
    - Defined in unique namespace.
    - Provide "post" method to create window.
    - Bind Tcl variables and commands to named channels.
  - Verilog Stub Module
    - Use Verilog "`initial`" to invoked the VPD "post" command.
    - Use Verilog "`always`" to link Verilog variables to Tcl variables.
    - More complex multi-variable interactions can also be implemented.

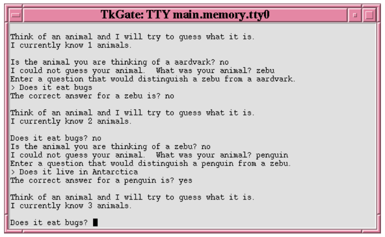
http://www.tkgate.org

TkGate Architecture

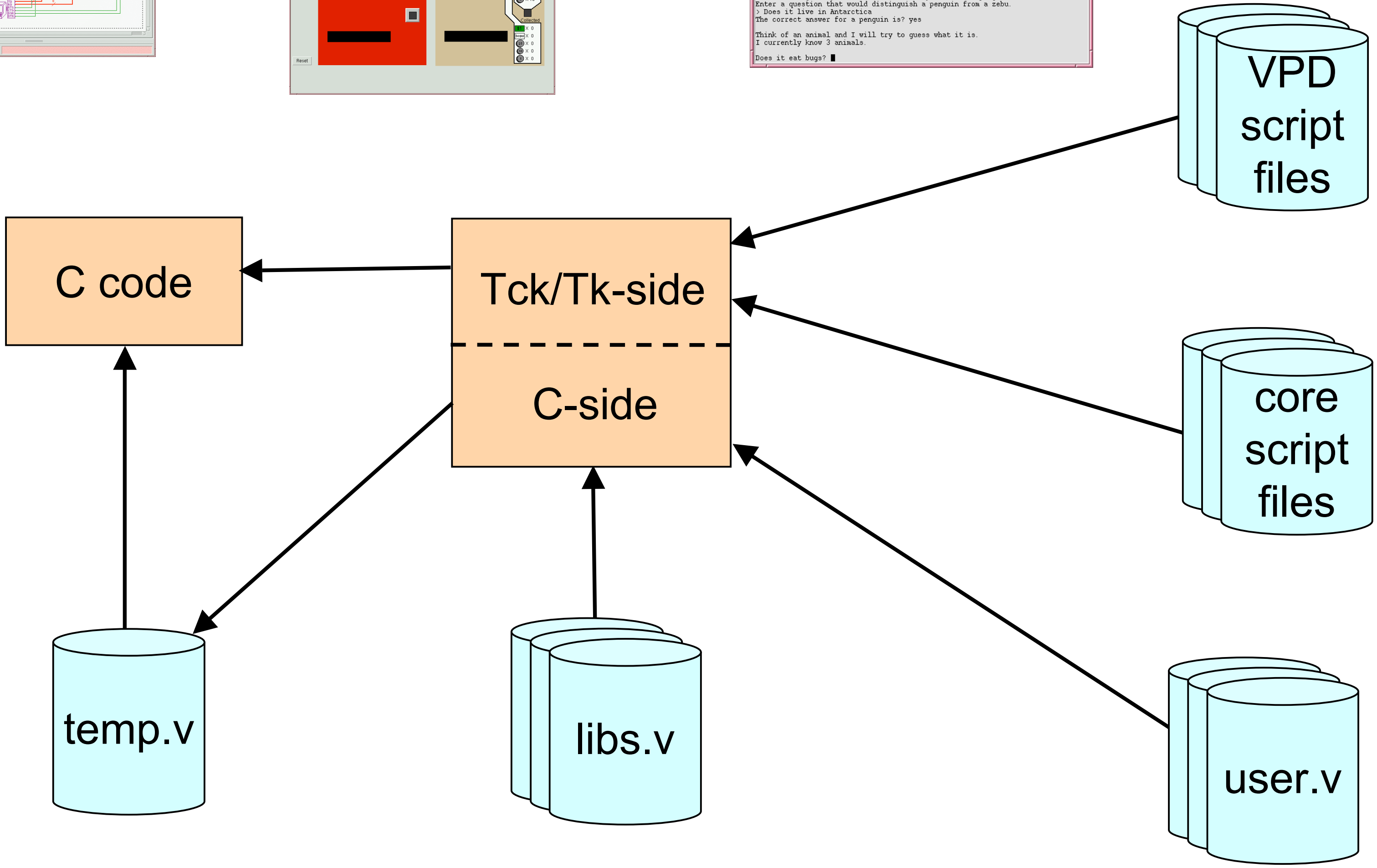


This is the TTY Virtual Peripheral Decice. Starting the simulation will cause the window for the TTY to open. Memory mapped control registers at the addresses TTYSTATUS and TTYDATA are used to control the device from the CPU.



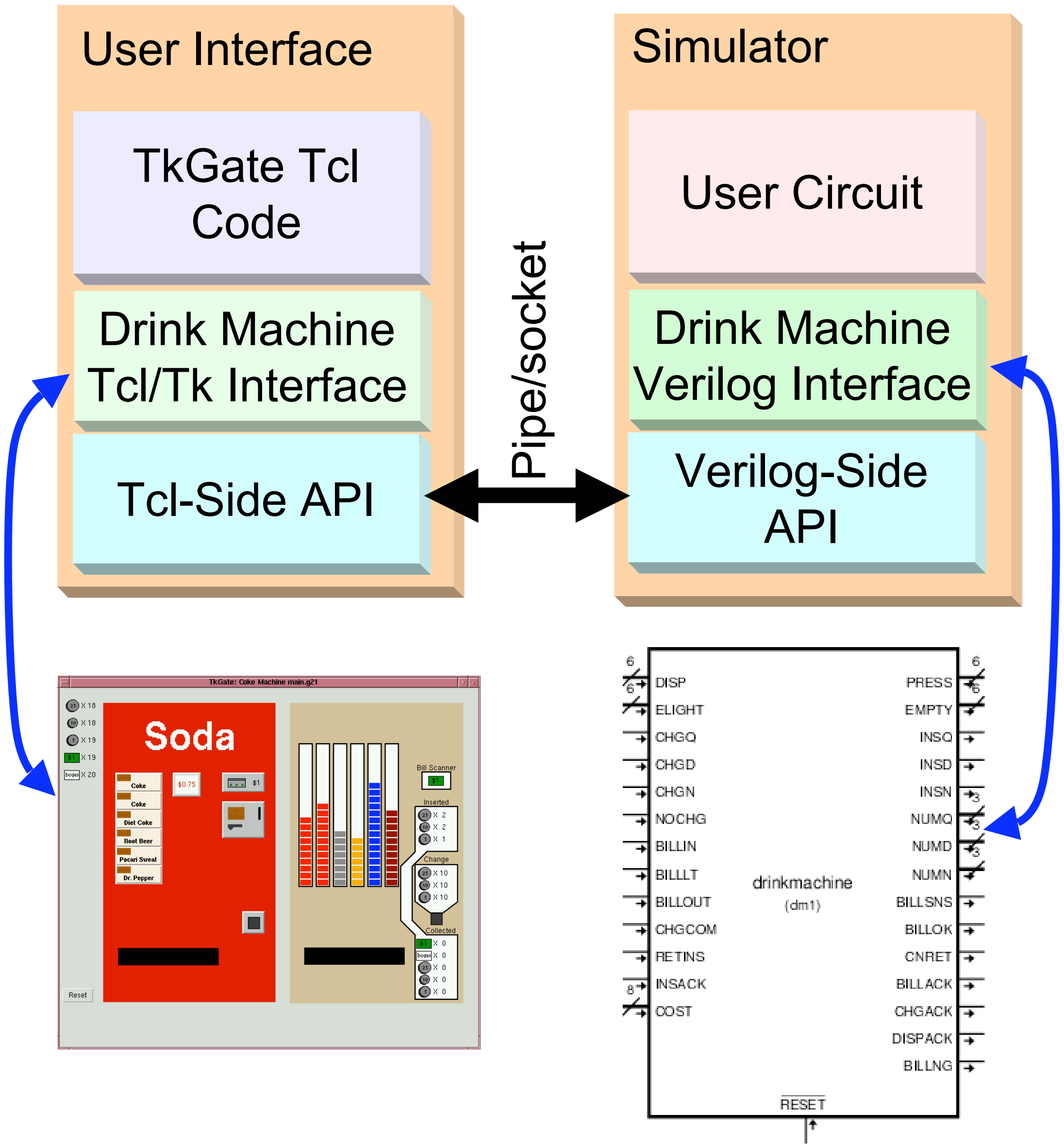


```
Think of an animal and I will try to guess what it is.
I currently know 1 animals.

Is the animal you are thinking of a aardvark? no
I could not guess your animal.  What was your animal? zebu
Enter a question that would distinguish a zebu from a aardvark.
> Does it eat bugs
The correct answer for a zebu is? no

Think of an animal and I will try to guess what it is.
I currently know 2 animals.

Does it eat bugs? no
Is the animal you are thinking of a zebu? no
I could not guess your animal.  What was your animal? penguin
Enter a question that would distinguish a penguin from a zebu.
> Does it live in Antarctica
The correct answer for a penguin is? yes

Think of an animal and I will try to guess what it is.
I currently know 3 animals.

Does it eat bugs? █
```



Drink Machine VPD

- **VPD Implementer**
  - Writes Tcl script implementing behavior.
  - Writes Verilog stub-module encapsulating interface.
- **VPD Client**
  - Loads library with desired VPD.
  - Creates one or more instances.
  - Creates control logic.



User Interface

TkGate Tcl Code

Drink Machine Tcl/Tk Interface

Tcl-Side API

Pipe/socket

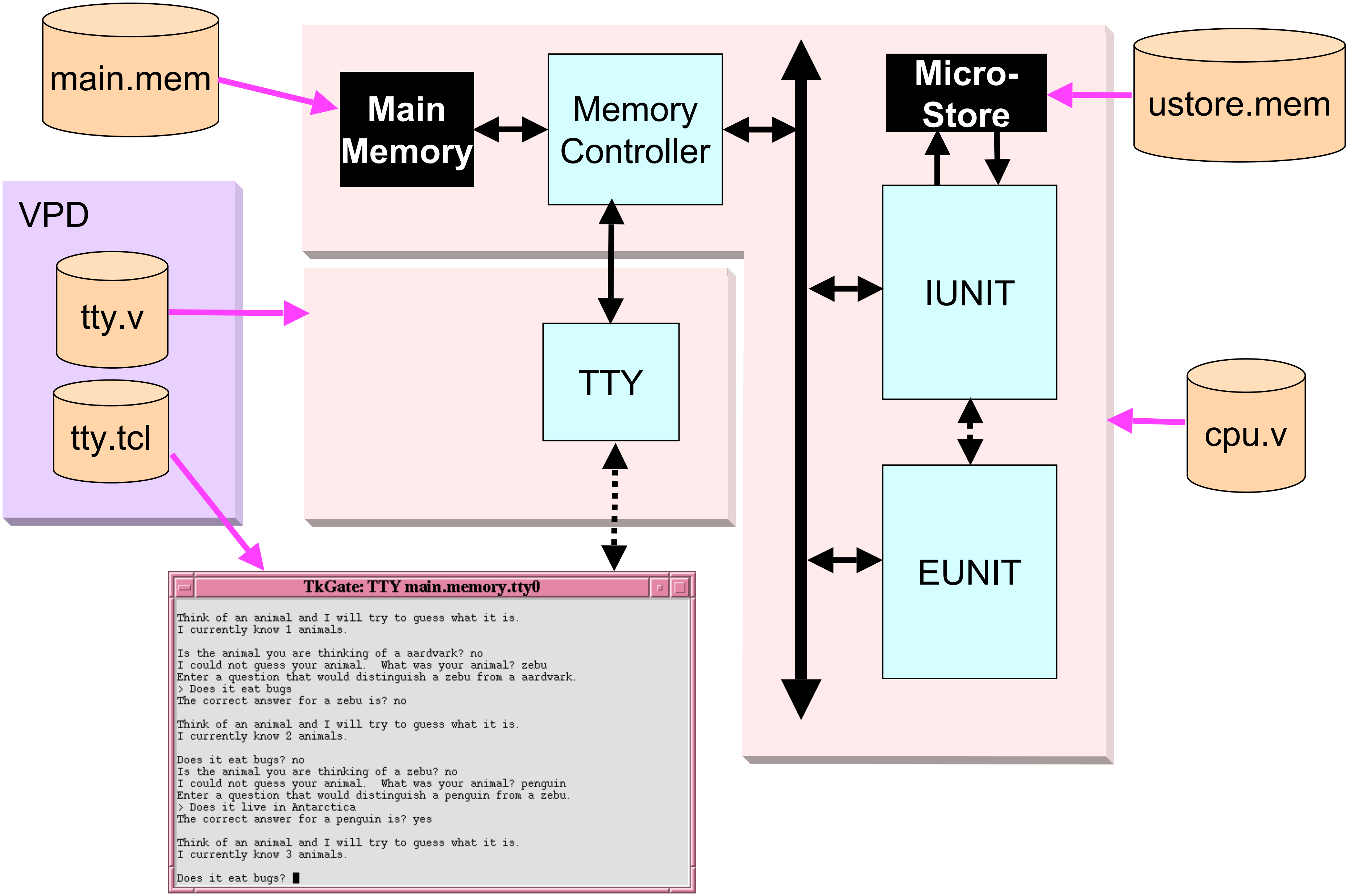
Simulator

User Circuit

Drink Machine Verilog Interface

Verilog-Side API

TTY Virtual Peripheral Device


main.mem

Main Memory

Memory Controller

Micro-Store

ustore.mem

VPD

tty.v

tty.tcl

TTY

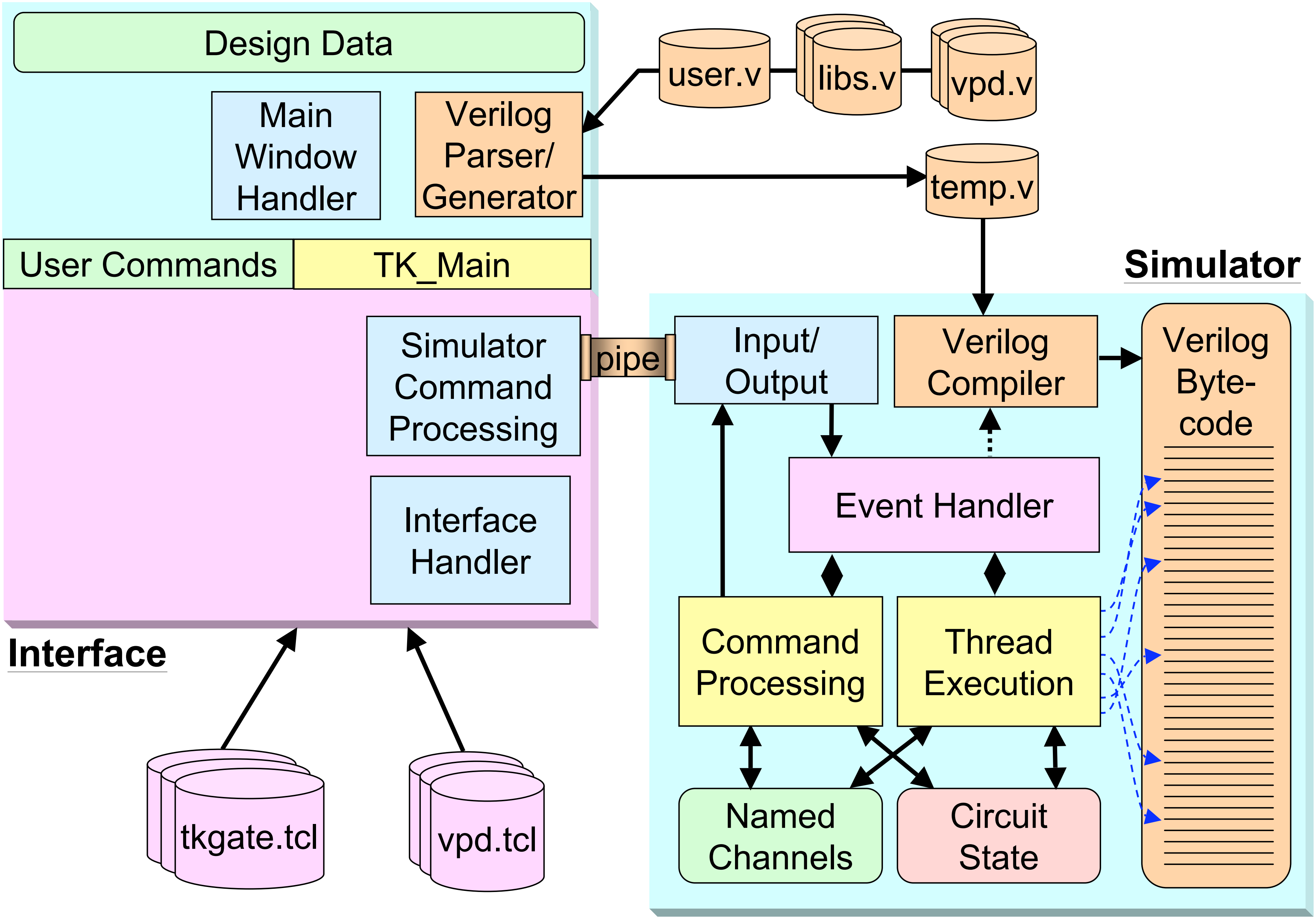
IUNIT

EUNIT

cpu.v

```
Think of an animal and I will try to guess what it is.
I currently know 1 animals.

Is the animal you are thinking of a aardvark? no
I could not guess your animal.  What was your animal? zebu
Enter a question that would distinguish a zebu from a aardvark.
> Does it eat bugs
The correct answer for a zebu is? no

Think of an animal and I will try to guess what it is.
I currently know 2 animals.

Does it eat bugs? no
Is the animal you are thinking of a zebu? no
I could not guess your animal.  What was your animal? penguin
Enter a question that would distinguish a penguin from a zebu.
> Does it live in Antarctica
The correct answer for a penguin is? yes

Think of an animal and I will try to guess what it is.
I currently know 3 animals.

Does it eat bugs? █
```

TkGate Architecture


Design Data
Main Window Handler
Verilog Parser/ Generator
user.v
libs.v
vpd.v
temp.v
User Commands
TK_Main
Simulator
Simulator Command Processing
pipe
Input/ Output
Verilog Compiler
Verilog Byte-code
Interface Handler
Event Handler
Interface
Command Processing
Thread Execution
Named Channels
Circuit State
tkgate.tcl
vpd.tcl