

Circuit Design and Simulation with VHDL

2nd edition

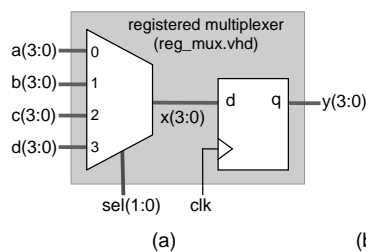
Volnei A. Pedroni

MIT Press, 2010

Book web: www.vhdl.us

Appendix B Quartus II Tutorial (Quartus II 9.1 sp1)

This tutorial is based on **Quartus II 9.1 sp1 Web Edition** (free at www.altera.com). The circuit used in the tutorial is the registered multiplexer of figure 1a, with the VHDL code of figure 1b.



(b)

```

1  -----
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  -----
5  ENTITY reg_mux IS
6  PORT (a, b, c, d: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
7        sel: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
8        clk: IN STD_LOGIC;
9        y: OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
10 END ENTITY;
11 -----
12 ARCHITECTURE reg_mux OF reg_mux IS
13     SIGNAL x: STD_LOGIC_VECTOR(3 DOWNTO 0);
14 BEGIN
15     x <= a WHEN sel="00" ELSE
16         b WHEN sel="01" ELSE
17         c WHEN sel="10" ELSE
18         d;
19     PROCESS (clk)
20     BEGIN
21         IF (clk'EVENT AND clk='1') THEN
22             y <= x;
23         END IF;
24     END PROCESS;
25 END ARCHITECTURE;
26 -----
27 -----

```

Figure 1

1. Starting a New Project

- Launch Quartus II.
- Create a new project by selecting **File > New Project Wizard**. The dialog of figure 2 will be opened.

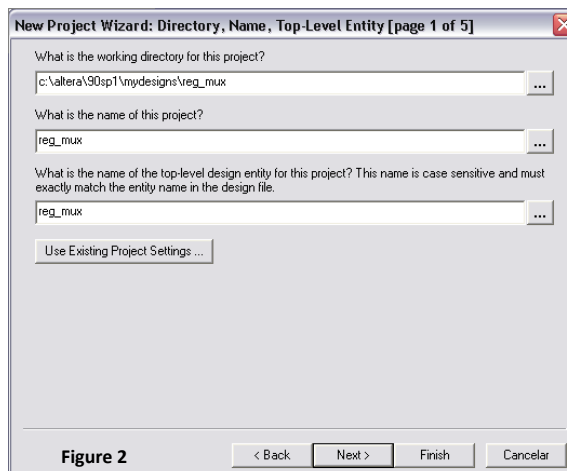

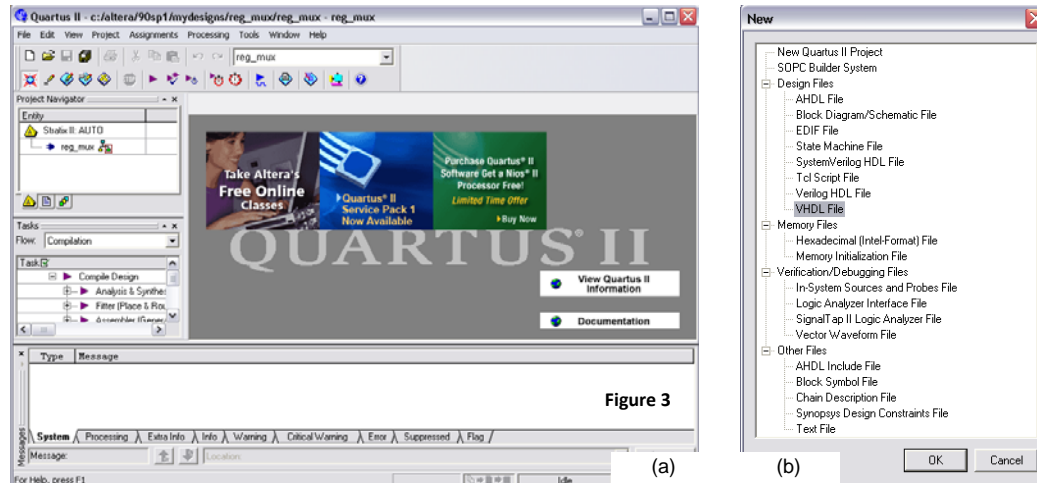


Figure 2

c) In the working directory field of figure 2, select the directory where all project files should be located (*reg_mux*, in this tutorial). If the directory does not exist yet, just type in the desired name (with the proper path, of course) and Quartus II will create it for you. Preferably, use the same name for the directory, the project, and the main VHDL entity.


d) In the project name field of figure 2, enter the desired project name (*reg_mux*). Note that the entity name field is automatically filled with the same name. Click **Finish** until the Project Navigator (figure 3a) is displayed.



e) Now enter the VHDL code. In case the code of figure 1b was already typed and saved in the work directory, proceed to Section 3. Otherwise, open the VHDL editor by clicking  or selecting **File > New**, which calls the dialog box of figure 3b. Select **VHDL File** and click **OK**. A blank page will be presented. Type the VHDL code and save it as *reg_mux.vhd*.



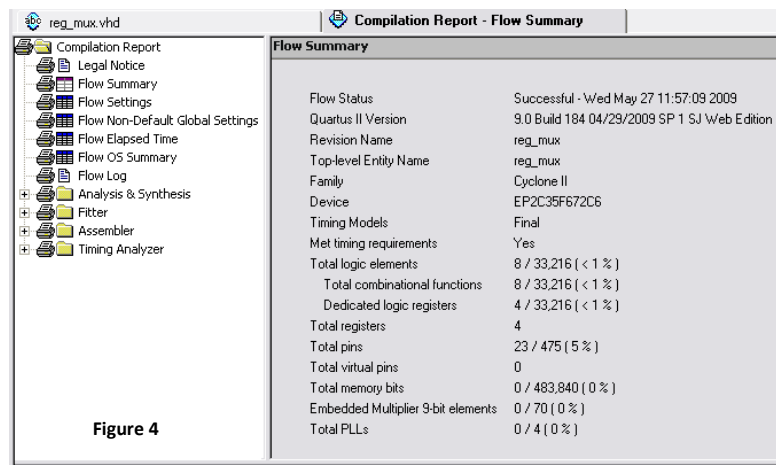
3. Synthesizing the Design

a) Select the device in which the circuit should be implemented by selecting **Assignments > Device**. For example, select Cyclone II EP2C35F672C6 when using the DE2 development board.

b) Compile the design by clicking  or selecting **Processing > Start Compilation**.

Note: For a faster compilation, particularly useful in large designs while the VHDL code is still being debugged, click  or select **Processing > Start > Start Analysis and Synthesis**. In this case, no timing information is recorded. After the code is working properly, full compilation () should then be performed.

c) When the compilation ends, the Compilation Report of figure 4 is exhibited, which contains several pieces of valuable information, some of which are described in the next section.



4. Inspecting Synthesis Results

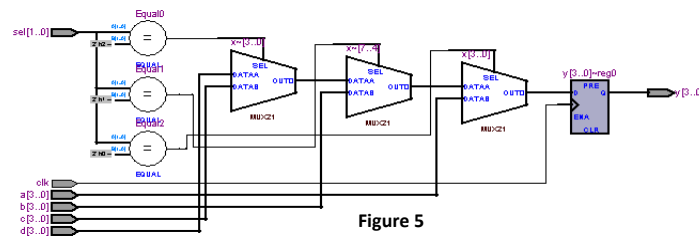
This section describes some of the results produced by the compiler.

a) *Device type and number of pins*: Check in figure 4 if the device type is the intended one (Cyclone II EP2C35F672C6, in this example). Check also if the total number of pins is as expected ($4 \times 4 + 2 + 1 = 19$ inputs + 4 outputs = 23 pins).

b) *Number of logic elements*: figure 4 shows also the amount of logic needed to implement the circuit. In this case, 8 logic elements were needed, out of ~33k LEs available in the chosen device.

c) *Number of registers*: Since y is a 4-bit signal (figure 1a), 4 flip-flops are expected to be inferred by the synthesizer. This too can be checked in the Flow Summary of figure 4, which shows a total of 4 registers.

d) *RTL View*: This tool shows how the code was interpreted by the compiler (before optimization and fitting). Select **Tools > Netlist Viewers > RTL Viewer**, which exhibits the circuit of figure 5 (or similar). Observe that because 2-input multiplexers are available in the device, a total of 3 units were used to implement the 4-input mux (the other three blocks at the input are for processing sel). Note also the 4-bit register at the output.



e) *Equations*: They represent the actual circuit implemented by the compiler. In the Compilation Report, select **Fitter > Equations** (if this option is not available, go to **Tools > Options > General > Processing** and mark **Automatically generate equation file during compilation**, then recompile the code). To interpret the equations, see first Section 8 ahead. For example, “A1L41Q = DFFEAS(A1L60, GLOBAL(A1L2),...);” means that the internal signal A1L41Q is produced by a D-type flip-flop (DFF) whose data input comes from A1L60 and whose clock input is fed by the global signal A1L2. Confirm in the equations that the total number of DFFs is indeed 4.

f) *Timing analysis*: In the Compilation Report, select **Timing Analyzer > Summary** and check the worst-case values for the three time delays below (these parameters are used, for example, to determine the circuit’s maximum clock frequency).

t_{co} (clock to output delay): Time necessary to obtain a valid output after a clock transition.

t_{su} (clock setup time): Time during which the data and/or enable inputs must be stable before a clock transition occurs.


t_h (clock hold time): Time during which the data and/or enable inputs must remain stable after a clock transition has occurred.

g) *Pin assignments*: In Section 6, it will be shown how to make or change pin assignments. For now, we want to simply check the assignments made automatically by the compiler. In the Compilation Report, select **Fitter > Resource Section > Input Pins**. This leads to the table on the left of figure 6. Next, select **Fitter > Resource Section > Output Pins** to see the output pins, shown on the right of figure 6.

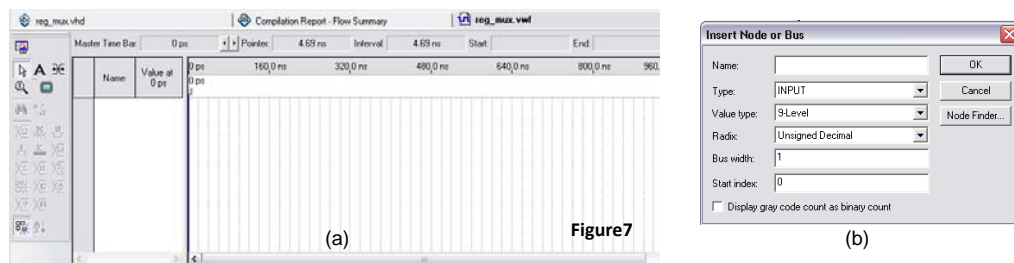
Input Pins							Output Pins						
	Name	Pin #	I/O Bank	X coordinate	Y coordinate	Cell number		Name	Pin #	I/O Bank	X coordinate	Y coordinate	Cell number
1	a[0]	AE9	8	20	0	0	1	y[0]	D10	3	20	36	0
2	a[1]	H11	3	18	36	3	2	y[1]	E10	3	18	36	0
3	a[2]	G21	5	65	33	0	3	y[2]	E8	3	7	36	3
4	a[3]	A9	3	20	36	2	4	y[3]	F11	3	18	36	1
5	b[0]	C13	3	31	36	2							
6	b[1]	G22	5	65	33	1							
7	b[2]	AC10	8	20	0	1							
8	b[3]	AC9	8	20	0	2							
9	c[0]	H12	3	18	36	2							
10	c[1]	B10	3	22	36	3							
11	c[2]	C10	3	20	36	1							
12	c[3]	AD10	8	22	0	1							
13	clk	P2	1	0	18	2							
14	d[0]	F6	2	0	33	0							
15	d[1]	D11	3	22	36	0							
16	d[2]	B9	3	20	36	3							
17	d[3]	A10	3	22	36	2							
18	sel[0]	D13	3	31	36	3							
19	sel[1]	G11	3	22	36	1							

Figure 6

5. Simulating the Circuit

a) To perform manual graphical simulation, we need first to create (draw) the input waveforms, based on which the simulator will calculate and plot the output waveform. Click  or select **File > New**, which will lead again to the dialog of figure 3b.

b) Select **Vector Waveform File** and click **OK**. The wave pane of figure 7a will then be exhibited.





c) Select **View > Fit in Window** (or press Ctrl+W) to have the complete plot exhibited in the waveforms window.

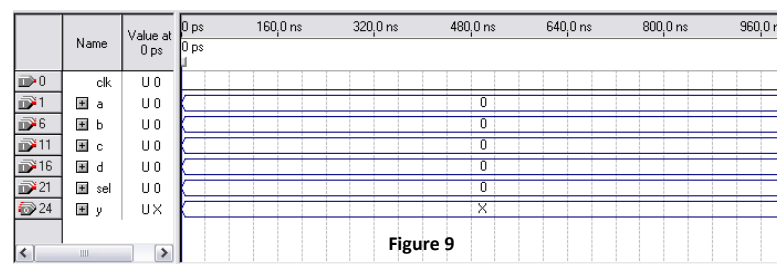
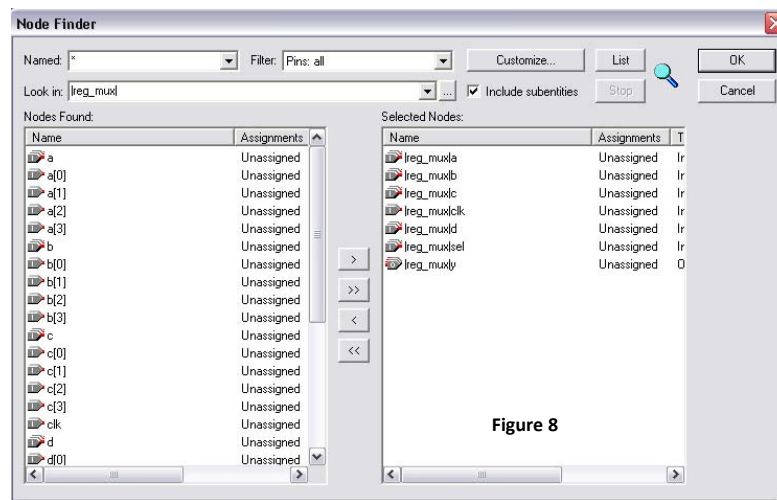
d) The time axis in figure 7 goes from 0 to 1 us. If a different end time is needed, select **Edit > End Time** and enter the desired value. Press Ctrl+W again to see the whole time axis.

e) The grid can also be adjusted as needed. Select **Edit > Grid Size** and enter 40 ns.

f) Now add the signals to the waveform editor. Press the right mouse button in the white area under **Name** and select **Insert > Insert Node or Bus**, which leads to the dialog of figure 7b.

g) In the Radix field select **Unsigned Decimal**, then click **Node Finder**. The dialog box of figure 8 (but empty) will be presented.

h) In the Filter field, select **Pins: all & Registers: post-fitting**, then click **List**. The window will be filled with the signals (partially) shown in the left column of figure 8. Select the desired signals, which can be copied to the column on the right with  (individually), with  (all), or by simply double-clicking the left mouse button on the name of the desired signal. Click **OK** twice, which will fill the waveforms window with the signals of figure 9.



i) To change the position (order) of a signal in figure 9, just select its name, release the mouse button, then press it again and hold, and drag the signal to the desired position. Note in figure 9 that all input signals come first, with the clock at the top.

j) Now we need to draw the input waveforms (*clk*, *a*, *b*, *c*, *d*, *sel*), after which the simulator will compute and draw the output waveform (*y*). The stimuli of figure 10 will be adopted.

j1) Draw the waveform for *clk*: Select the entire line of *clk* and click on the clock icon . Enter 80ns for the period and '0' for the initial value. Click **OK**.

j2) Draw the waveform for *a*: Highlight line *a* from 0 to 80ns, click the arbitrary value icon , enter 2, and click **OK**. Repeat the operation entering 3 for the interval from 80ns up to the end of the simulation (1us).

j3) Draw the waveforms for *b*, *c*, *d*: Repeat the process above, entering the values shown in figure 10.

j4) Draw the waveform for *sel*: Select line *sel* and click the counter icon . Enter Start value = 0, Increment = 1, and Count every = 160ns. Click **OK**.

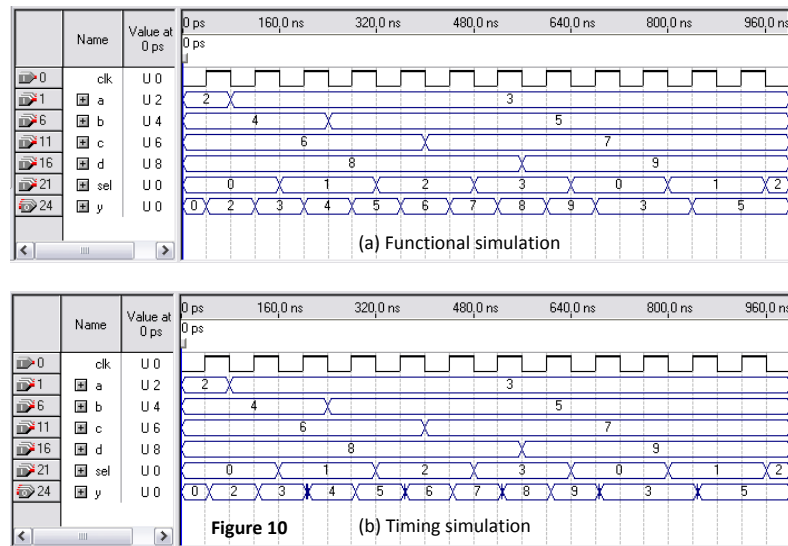
j5) Save the file with the same name as the entity's and with extension *.vwf* (Vector Waveform File), that is, *reg_mux.vwf*. Recall that the last waveform (*y*) will be filled by the simulator.

k) Now we must choose between *functional* or *timing* (default) simulation. The former checks only the design functionalities, while the latter includes also the device's internal propagation delays, thus representing the actual circuit.

- For functional simulation, continue in step 5(l) below.


- For timing simulation, go to step 5(m).

l) *Functional simulation*: Select **Processing > Generate Functional Simulation Netlist**. After the program finishes producing the netlist, select **Assignments > Settings > Simulator Settings**. In the Simulation mode list, choose **Functional**. Finally, in the Simulation input field, enter the name of the waveform file (*reg_mux.vwf*) and click **OK**. When done, go to step 5(n).



m) *Timing simulation* (default settings): Select **Assignments > Settings > Simulator Settings**. In the Simulation mode list, choose **Timing**. In the Simulation input field, enter the name of the waveform file (*reg_mux.vwf*) and click **OK**.

n) Recompile the code in case the setup in 5(l) or 5(m) was modified, then proceed to step 5(o). Otherwise, go directly to 5(o).

o) Run the simulation by clicking  or selecting **Processing > Start Simulation**. The simulator will draw the waveform for *y* of figure 10a if it is a functional simulation (note that there are no delays between the transitions of *clk* and *y*) or of figure 10b if it is a timing simulation (note the propagation delay between positive clock transitions and the settling of *y*).

p) Examine the results: When the simulation ends, Quartus II displays the results in a separate window. Instead of examining that window, click the tab related to the original waveforms window (*vwf* plots), to which the new results will then be automatically copied.

Note: If the results are not copied to the original waveforms window, select **Assignments > Settings > Simulator Settings > Simulation Output Files** and mark **Overwrite simulation input file with simulation results**, then re-run the simulation.

6. Making Pin Assignments

Pin assignments can be made manually or using a CSV (Comma Separated Value) file. Assignments can be deleted with **Assignments > Remove Assignments**.

6.1 Manual pin assignments

a) Select **Assignments > Pins**, which opens the window of figure 11.

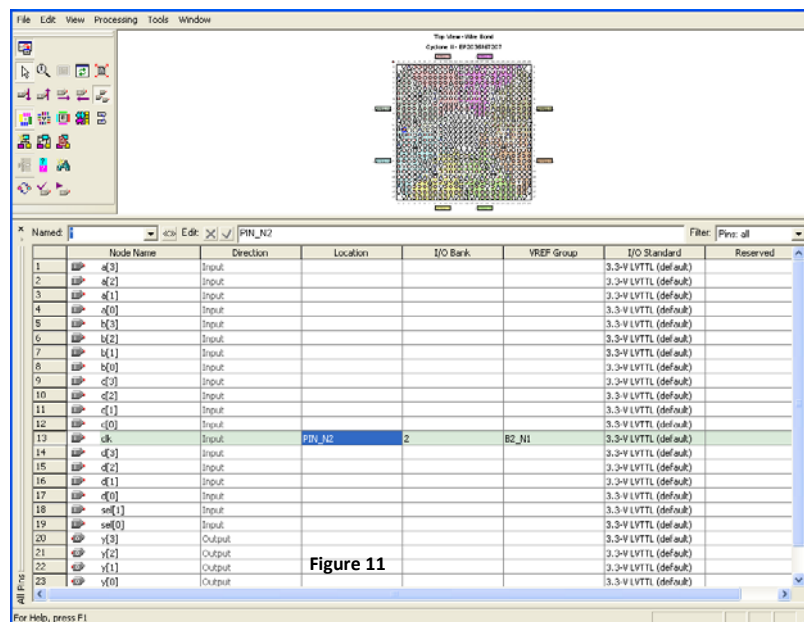


Figure 11

b) As an example, the pin for *clk* was already assigned in figure 11. Double-click the left mouse button in the white area under Location. A pin number can be selected from the pull-down menu, or its name can be typed in (for example, for PIN_N2, only N2 or n2 needs to be typed). Since the clock is a special signal, a specific type of pin should be assigned to it, identified as Dedicated Clock in the pin list. Select, for example, N2.

c) Repeat this process for each of the circuit pins.

d) Finally, recompile the design.

6.2 Importing pin assignments with a CSV file

Pin assignments can be exported to and imported from a file, saved with the extension *.csv*. The initial pin assignments, made automatically by the compiler, cannot be exported.

a) Select **Assignments > Assignment Editor**.

b) Select **Assignments > Import Assignments**. As an example, try to import the pin assignments for the DE2 board, available in a file called *DE2_pin_assignments.csv*.

c) Recompile the code.

Note: To export a pin assignment, select **Assignments > Pin Planner**, then **File > Export**, and save the file with the extension *.csv*.

7. Downloading the Design to the FPGA

a) Connect the board containing the CPLD or FPGA device to an USB port of your computer and turn the power on the board on.

Note: If this is the first time that you are using a board that interfaces using an USB port, execute Section 2 of Appendix E: *Installing the USB-Blaster Driver* (this is needed only once).

b) In Quartus II, click the Programmer icon  or select **Tools > Programmer**. figure 12 will be displayed. Note the following in the figure:

- The programmer file is *reg_mux.sof*.
- The Program/Configure box is checked.

- The driver is USB-Blaster.
- The mode is JTAG.

c) Click **Start**, and the device will be programmed.

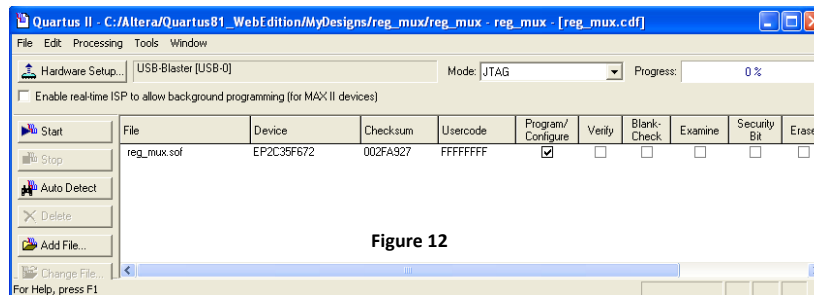


Figure 12

8. Interpreting the Fitter Equations

Below are the main symbols used in the Fitter equations.

a) Logic operators: ! (NOT), & (AND), # (OR), \$ (XOR)

b) Flip-flops:

DFF (D, CLK, CLRN, PRN) (DFF with reset and preset, both active low)

DFFE (D, CLK, CLRN, PRN, ENA) (DFF above plus enable)

DFFEAS (D, CLK, CLRN, PRN, ENA, ADATA, ALOAD) (DFF above plus asynchronous data load)

DFFEAS (D, CLK, CLRN, PRN, ENA, ADATA, ALOAD) (DFF above with synchronous clear)

TFFE (T, CLK, CLRN, PRN, ENA) (TFF with reset, preset, and enable)

Note: Recall that in the context of this book an output-zeroing command is called *reset* when it is *asynchronous* or *clear* if it is *synchronous*.