jjtag

Pablo Bleyer Kocik [pablo at bleyer dot org]

jjtag is a framework for interfacing and managing Joint Test Action Group (JTAG) IEEE-1149.1 controllers in Java.

jjtag defines a common API for JTAG TAP controllers and provides utility classes that encapsulate frequent JTAG actions.

The TAP Controller

The IEEE-1149.1 standard defines a common interface for JTAG control, called the *Test Access Port* (TAP). The TAP interface is a set of output and input pins that allows to serially insert and extract internal register data, and at the same time to control the embedded JTAG controller's state. Since the data is processed serially, the registers that will be read or written through the TAP are arranged in a daisy-chain fashion. This structure is called a boundary scan chain. Chains are not bound to a single physical device. The TAP interface was designed such that multiple ICs that support JTAG may be linked together to create compound chains.

Originally, as the JTAG name implies, the interface was conceived to perform electrical connectivity tests between components in a board; however in time developers found other uses for it. It is common to use the TAP to initially program the Flash memories of a board in order to bootstrap it (bit-banging the cells that control the I/O ports of a chip), and some silicon vendors use the interface to implement embedded In-Circuit Emulators (ICE) for their processors (using the TAP to select cell groups that include debug data and debug control registers).

Table 1 shows the signals present in a TAP interface. The TDI, TDO, TMS and TCK signals are mandatory. The TRST signal is optional since it is always possible to soft-reset the TAP machine clocking five ones into the TMS pin ('11111').

TAP signal name	I/O (wrt Target)	Description
TDI	I	TAP data input
TDO	O	TAP data output
TMS	I	TAP state control
TCK	I	TAP clocking
TRST	I	TAP reset

Table 1: TAP signals

TAP data is

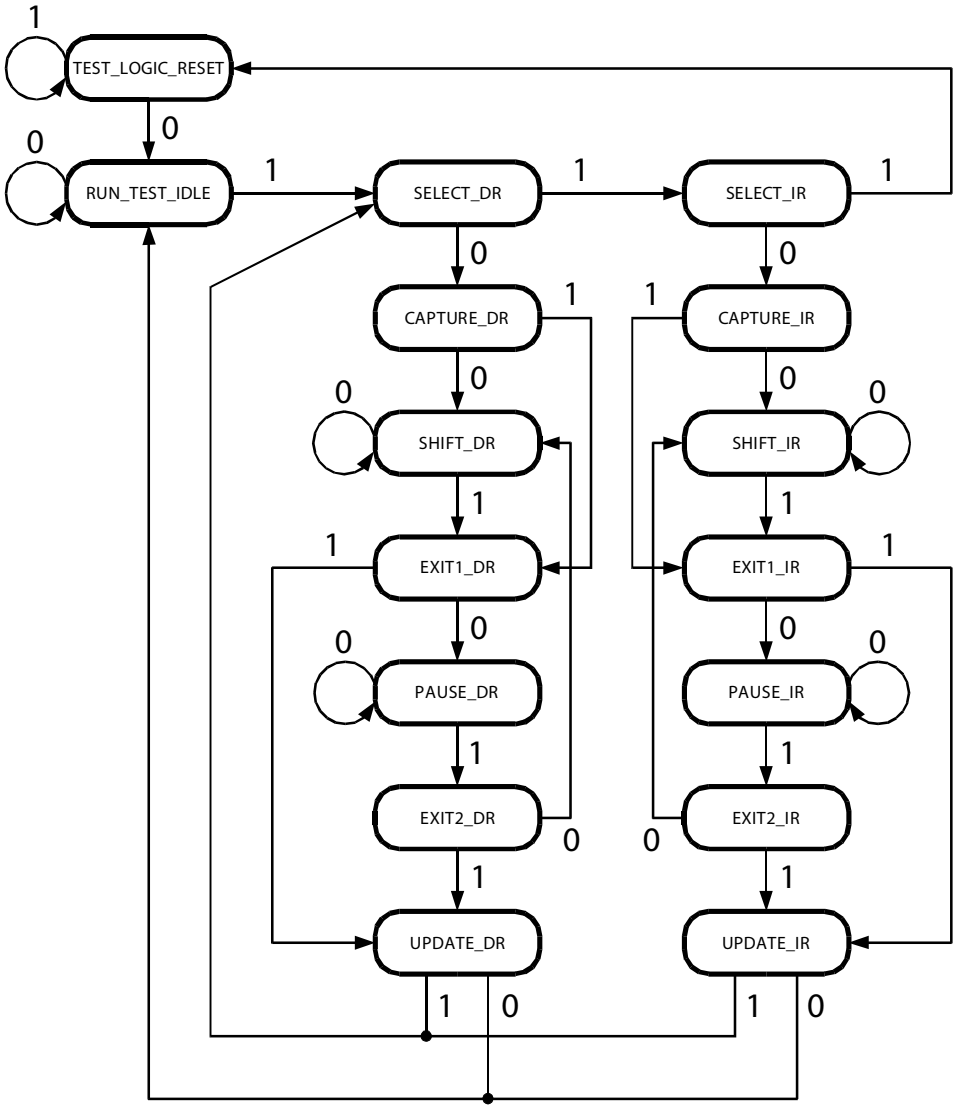

Figure 1: TAP State Machine

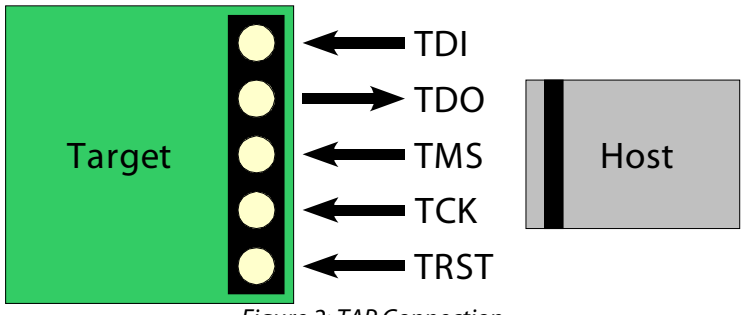

Figure 2: TAP Connection

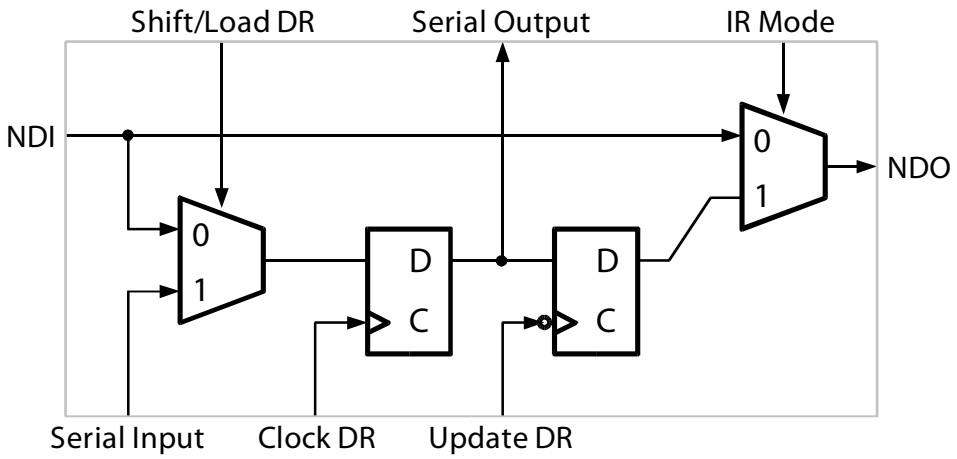

Figure 3: Boundary Scan Cell

API

byte state;	Holds the current state of the JTAG controller.
int delay;	Controls the inter-delay time between bit clocking. Meaning is controller-dependent, delay is usually expressed in milliseconds.
void initialize() throws Exception;	Initializes and resets the controller.
void trst(int);	Controls the TRST signal. If the target TAP interface does not have one, then a soft-reset should be performed.
void tms(byte[], int);	Controls the TMS signal. The controller should keep track of the target's TAP state and update the state variable accordingly.
void tdi(byte[], int, byte[], boolean);	Controls the TDI signal.
void rtdi(byte[], int, byte[], boolean);	
String path(byte);	
void jump(byte);	
void reset();	