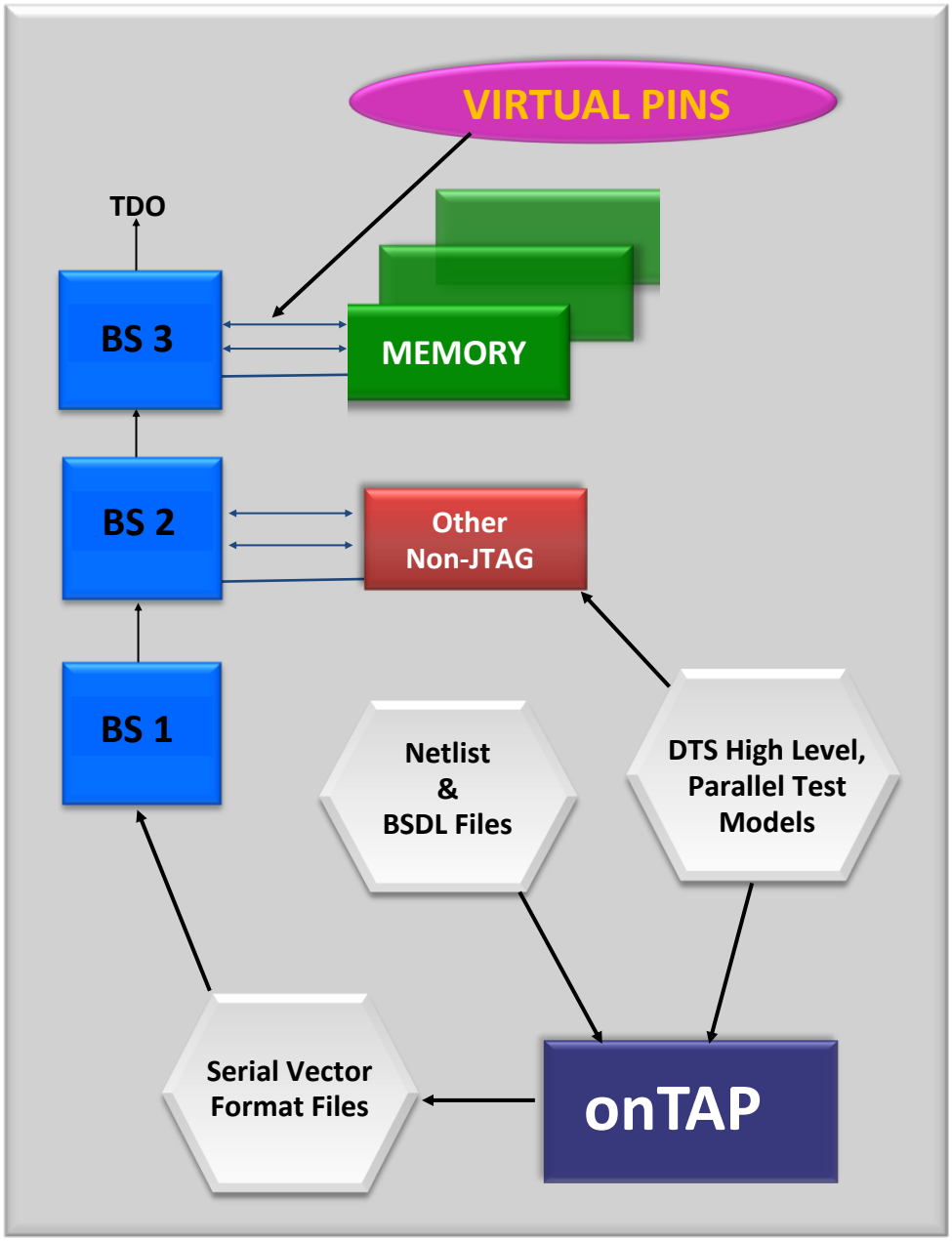
Memory Cluster Testing

Memory and cluster testing tests the connectivity between boundary scan pins and non-JTAG logic. Create reusable test models for memory and other types of logic in DTS, a high-level modeling language.

Develop non-JTAG test models with the C-like flow of DTS language. Models are prepared regardless of details of boundary scan implementation. onTAP's Virtual Nails serializes a model's test vectors into scans and automatically associates boundary scan pins, to act as driver/sensors with non-JTAG device pins.



onTAP compiles high-level DTS models into SVF files. "Virtual Pins" map Boundary Scan pins to the pins on the Memory/Cluster test devices.

Highlights:
Memory & Cluster Test

- Tests connectivity between boundary scan and non-boundary scan pins
- Employs high-level DTS test language to create reusable test models
- Allows for direct control of JTAG pins to control direction and enable pins on intervening buffers
- Runs with onTAP JTAG Controller (standard or low voltage)
- Up to 10MHz clock rates through programming cables
- External control of critical pins, such as Write Enable pins, when accessible
- Virtual Pins serializes data and adapts non-JTAG pins to boundary scan pins

All Rights Reserved

TURNKEY JTAG TEST SOLUTION
BIST REUSABLE JTAG TEST JTAG Test GPIO Memory Test
Configure Logic ATE FPGA Parallel JTAG Test
IEEE 1532 BURN IN CPLD IEEE 1149.6 IN SYSTEM PROGRAMMING ICT
FLASH PROGRAMMING BOUNDARY SCAN TEST SOLUTIONS

Using DTS Models for Memory / Cluster Testing

Memory and cluster testing is a powerful technique used to check the connectivity between boundary scan pins and non-JTAG logic. Reusable test models for memory and other types of logic may be prepared in DTS, a high-level modeling language.

DTS brings C-like flow control and expressions to the development of test models for non-JTAG devices. Models may be prepared without regard to the details of boundary scan implementation, since onTAP's Virtual Nails serializes a model's test vectors into scans and automatically associates boundary scan pins, to act as driver/sensors, with non-JTAG device pins.

An example that shows how DTS models can be used to set and test pin values on non-JTAG logic follows. This example develops some test code for a hypothetical memory device and employs some test instructions that show DTS's capabilities. Note that after the pins and pin groups are declared, the test instructions focus only on the device's pins, independent of boundary scan implementation. However, also note that an EXTERNAL pin has been declared in the header. EXTERNAL pins may be inserted where the model needs to directly control boundary scan pins, e.g., to drive enable pins on buffer devices.

The example will write some data into memory addresses and call a clock. Data will then be read back from those addresses.

```
                // Declare a header and pin associations
                .HEADER;
                .INPUT(8=CLK,9=CS,10=OE);
                .BIDIR_TRI(11=DQ0,13=DQ1,14=DQ2,15=DQ3,16=DQ4,17=DQ5,18=DQ6,19=DQ7);
                .OUTPUT(34=ADDR0,35=ADDR1,36=ADDR2,.....,54=ADDR21);
                .EXTERNAL(U34.A25); // insert external pin for direct control of B/S pins
                .END HEAD;
                .DECLARE GROUP DQ(DQ7,DQ6,DQ5,DQ4,DQ3,DQ2,DQ1,DQ0);
                .DECLARE GROUP ADDR(ADDR21,ADDR20,ADDR19,.....,ADDR0);

                int x,fails;
                sub clock(); // forward declare sub routines
                fails = 0;


                .MAIN;
                IL(CLK); // Input Low sets CLK low
                IH(CLK,OE); // Input High sets CLK and OE high
                IL(CLK,CS); // set CLK and CS low
                ADDR = 1; // pin groups may be used in expressions as variables
                IH(U34.A25); // assert control on external pin…say B/S pin that controls buffer
                // write hex 55 to addresses based on a walking one
                for ( x = 1; x <= 22; ++x ) // example of FOR loop run-time flow control
                {
                DQ = 0x55;
                IG(ADDR; // Input Group asserts ADDR and DQ pin values
                clock(); // clock data
                OG(DQ);
                if ( DQ != 0x55 )
                {
                ++fails;
                if ( fails > 100 ) // example of how if can be used at run-time to exit a program
                {
                    MESSAGE_PAUSE("Exiting with over 100 failures");
                    exit;
                }
                ADDR <<= 1; // left shift ADDR value
                }
}
Clocks() // clocks sub-routine
{
                IL(CLK);
                IH(CLK);
}
.END MAIN;
```

This model shows the basis of DTS testing. Once pins and pin groups are declared, C-like expressions may then be employed to set and test pin values as well as to control program flow.