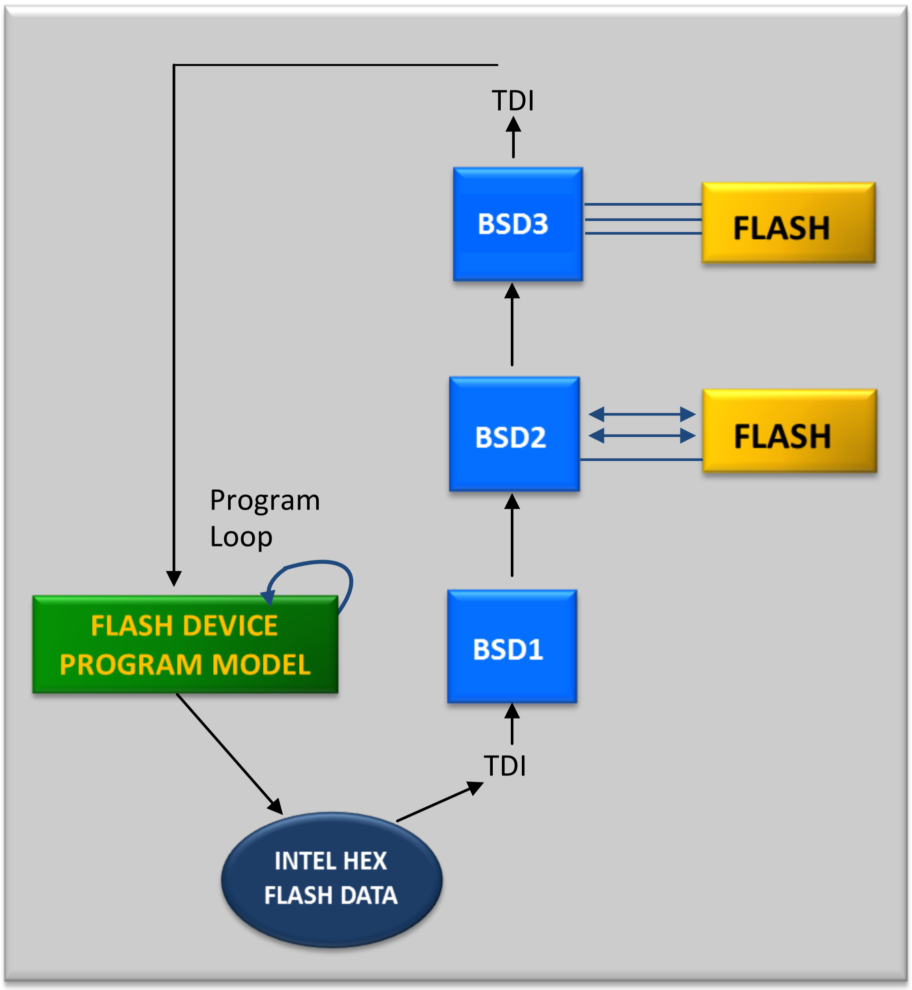
onTAP Series 4000 with ProScan

Flash Programming

onTAP uses flexible, reusable models of flash devices to read FLASH data from Intel HEX formatted files and write the data to FLASH devices from scannable boundary scan pins. Serializing scan data at run-time, while programming, provides compact easily maintained models.

Time to Program

Programming time (PROG_TIME) will be determined by the relationship between the number of addresses (ADDRESSES) in the FLASH, the number of instructions (i.e., scans) in the loop, the number of scan cells in the path (SHIFTS), and the effective clock throughput RATE (CLOCK).



onTAP's Flash program models read data from HEX files, serializes it and applies it to Flash devices through scannable pins on boundary scan devices.

Highlights:
Flash Programming

- **Program and verify FLASH devices**
- **Serializes and applies data from boundary scan pins**
- **Reads INTEL HEX files to transfer data**
- **Employs high level, C-like, reusable program models**
- **Runs with onTAP's USB TAP CONNECT Controller**
- **Up to 10 MHz clock rates through TAP CONNECT Controller**
- **External control of *Write Enable* pins when accessible**
- **Test multi-die modules**
- **Virtual Pins automatically adapts FLASH pins in netlist circuit models to boundary scan pins**
- **Windows 7—Windows 10 compatible on 32 and 64 bit machines with 2.2 GHz Processor and 3 GB RAM**

PROG_TIME = ADDRESSES * SHIFTS * SCANS / CLOCK.

For example, the programming time to load 100,000 boot code addresses, requiring four scans per address, at a 3.5 MHz effective throughput rate, through a scan chain 500 cells in length, would be:

PROG_TIME = 100,000 * 500 * 4 / * 3.5 MHz = 40 seconds

onTAP's Flash Boot Code loading formula.

Each address scan loads one or more bytes of program data.

$\text{CLOCK}_{\text{USB}}$ = 3.5 MHz C

$\text{LOCK}_{\text{PPORT}}$ = 2.25 MHz

Program Models

onTAP's FLASH programming models are written in the DTS test language, featuring C-like logic expressions and flow control. DTS provides the flexibility to adapt existing models for new applications and circuit conditions. Some of the DTS features that make onTAP's FLASH programming so powerful include:

- Isolation from boundary scan circuits. Models may be prepared independently of boundary scan circuit considerations, addressing only the requirements of the FLASH devices. onTAP's Virtual Pins handle all of the serializing required to interact between JTAG scannable pins and FLASH devices.
- Capability to define pins in groups, such as ADDRESS, and then to directly manipulate the pin groups, just like variables in complex expressions.
- Declaration of variables, sub-routines, and C-like program flow control provide the flexibility to efficiently develop algorithms. • Many built in functions that may be used, for example, to open and close files, read data from files, and provide trace results for debug purposes.

```
DECLARE GROUP ADDR(A20,A19,A18,…. ,A8,A7,A6,A5,A4,A3,A2,A1,A0); .
DECLARE GROUP DQ(DQ7,DQ6,DQ5,DQ4,DQ3,DQ2,DQ1,DQ0);

OpenProgrammingFile(FILE=C:\temp\flash_files\MCC-SMALL.hex,FORMAT=INTEL_HEX,
    DEVICE=AM29LV256,ADDRESS=ADDR,DATA=DQ,BYTES=2,START_ADDRESS=0,END_
ADDRESS=0);
START_TIMER;
ADDR=X'0000000';

DO {
        if( GetNextFileData("INTEL_HEX","ADDR,DQ") == 0 )
        break;   // address and data saved in internal variables, FILE_ADDRESS and FILE_DATA
// Use FILE_ADDRESS and FILE_DATA to assign address and hex values from file to respective pin groups.
    // This allows other data values to be asserted on pin groups as intermediate steps.
    IH(CE,WE) IG(ADDR=X'0000007'); // Unlock Bypass Program...DEBUG value
    IL(CE,WE) IG(DQ=X'00A0');
    ADDR=FILE_ADDRESS >> 1;
    IH(CE,WE) IG(ADDR,DQ=FILE_DATA); // ADDR FILE_ADDRESS right shifted one position
    IL(CE,WE); //
   }
WHILE( TIMER < 40s );       // provides safety timeout
```

Refer to onTAP's Memory /Cluster programming option for further information.

Sample Program Code

This excerpt of code from a program model shows how much can be accomplished with relatively little code. Code to erase the device is not shown. After declaring pin groups and opening a hex data file, a DO-WHILE() loop is created in which file data is successively read by the built-in function GetNextFileData(). GetNextFileData() assigns current address and data values to built in variables FILE_DATA and FILE_ADDRESS from where they may be assigned to pin groups and then written to FLASH devices. Built-in functions, Input High, IH(), and Input Low, IL(), directly assign values to pins, while Input Group, IG(), assigns values to the pins in a pin group.

onTAP takes these assignments at run time, and each time through the program loop, adjusts cell values in the serialized scan data to accomplish the intent of each instruction.