POWERMUX a proposal for managing INIT_DATA as it crosses multiple power domains

CJ Clark, Intellitech

REGISTER_ASSEMBLY has been proposed and accepted
- Scan chain length deferred and calculated from R_A

```
attribute REGISTER_ASSEMBLY of chip_2011 : entity is
      "INIT_DATA ( "&
        -- TDI
        "(USING SerdesO),( i1 IS init_data DEFAULT.ALLBITS (CHPMFG(Test)) ), "&
        "(USING SerdesO),( i2 IS init_data DEFAULT.ALLBITS (CHPMFG(Test)) ), "&
        "(USING SerdesH),( i3 IS init_data DEFAULT.ALLBITS (CHPMFG(Test)) ), "&
        "(USING SerdesO),( i4 IS init_data DEFAULT.BSTERM  (BSTERM(CPflt))
DEFAULT.BSCM (BSCM(DC_CPL)) DEFAULT.BSSWING (BSSWING(1115mV)) ) "&
        -- TDO
        ")";
```

REGISTER_ASSEMBLY with i1, i2, i3 and i4 provide 4 x 6 bits or 24 bits of scan data.
(John Siebold example, recall SERDESH and SERDESO have a 6 bit tdr segment).

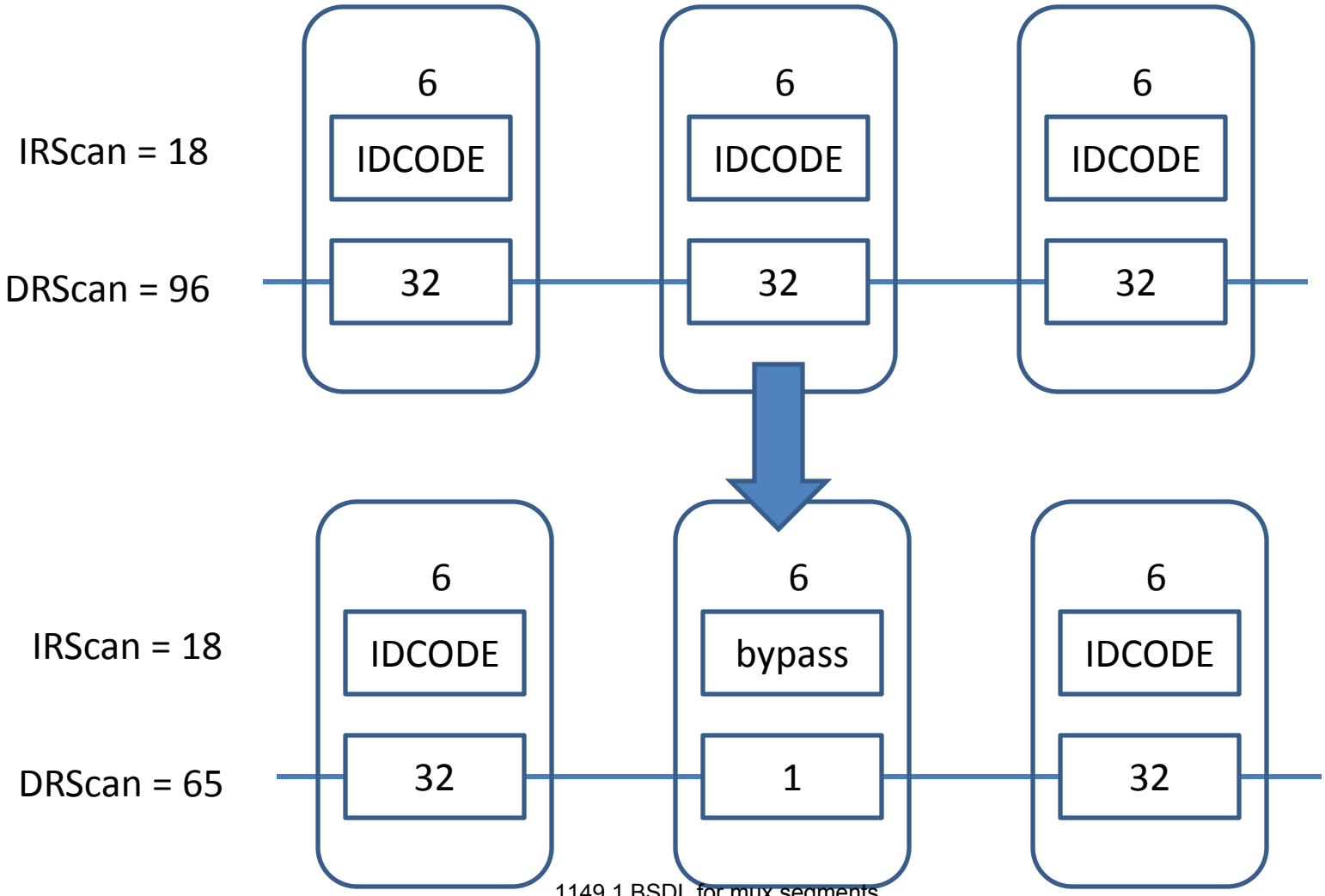
Recall that without i3 in the description the same process Takes place of reading the instances and totaling up the Lengths.

```
attribute REGISTER_ASSEMBLY of chip_2011 : entity is
      "INIT_DATA ( "&
        -- TDI
        "(USING SerdesO),( i1 IS init_data DEFAULT.ALLBITS (CHPMFG(Test)) ), "&
        "(USING SerdesO),( i2 IS init_data DEFAULT.ALLBITS (CHPMFG(Test)) ), "&
             -- i3 is missing or commented out
        "(USING SerdesO),( i4 IS init_data DEFAULT.BSTERM  (BSTERM(CPflt))
DEFAULT.BSCM (BSCM(DC_CPL)) DEFAULT.BSSWING (BSSWING(1115mV)) ) "&
        -- TDO
        ")";
```

A tool which knows how to read REGISTER_ASSEMBLY and found only i1, i2 and i4 would calculate that there are 3 x 6 bits or 18 bits of scan data.
(John Siebold example, recall SERDESH and SERDESO have a 6 bit tdr segment).

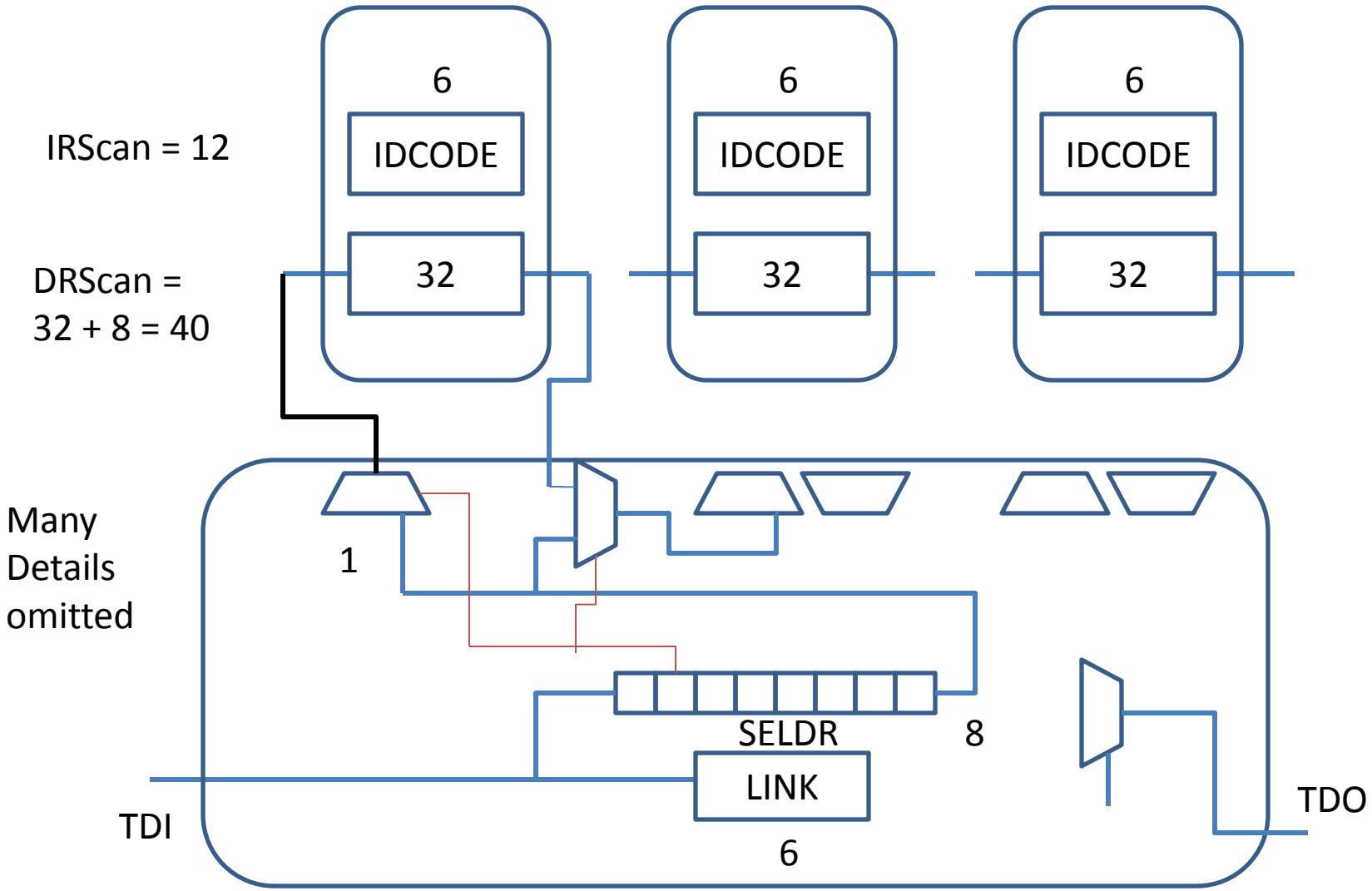
A tool could programmatically allow one to add to R_A or subtract from it.

Recall that Board level scan-chains – loading bypass as an IR scan changes length by 31 bits for the subsequent DR scans - most tools know how to manage this



1149.1 BSDL for mux segments

Scan ‘linking’ devices have been known since 1990
- most tools know how to support these
- 4 to 16 paths or ‘rings’ in the chain or in idle/TLR

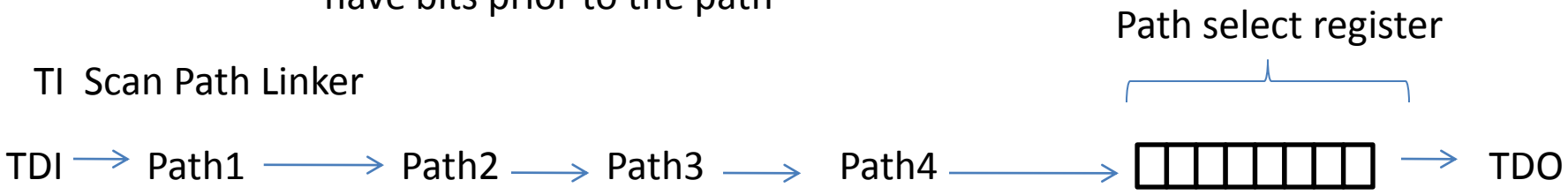


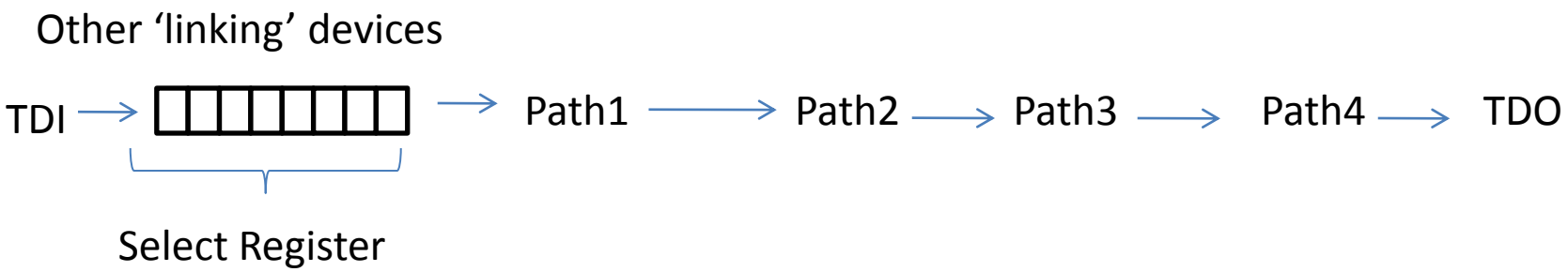
1149.1 BSDL for mux segments

Bits in a board level scan-chain - well known

device collapses or expands chain.

TI device has bits at end after paths, other devices have bits prior to the path

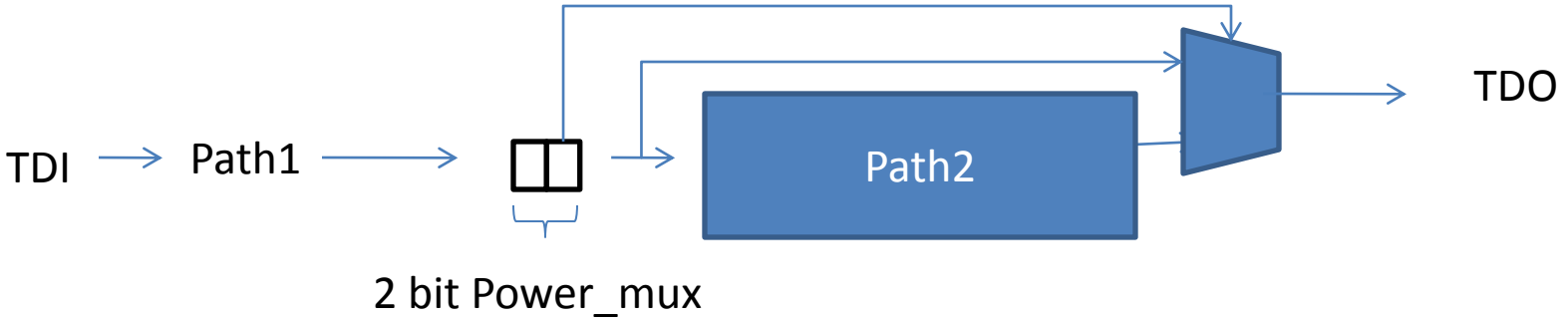




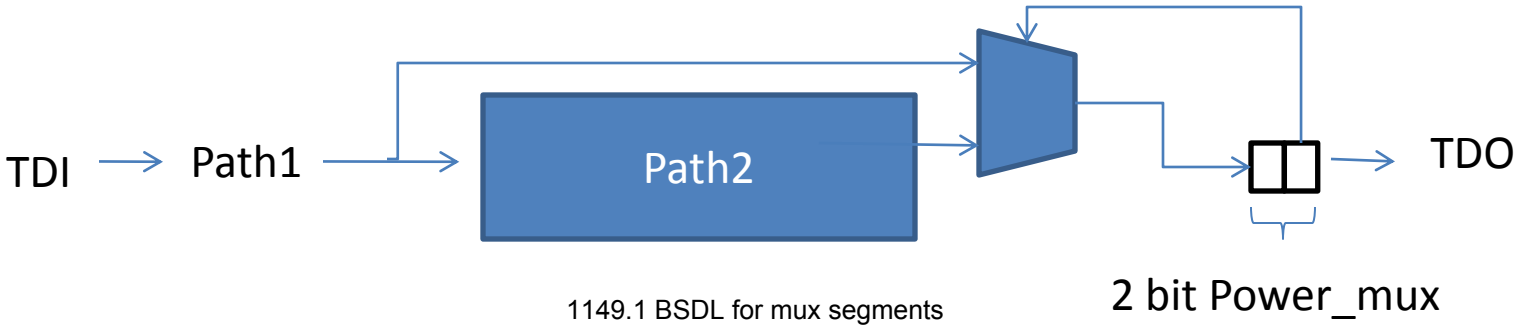
INIT_DATA – what happens when we cross power domains?

A 2 bit power_mux could configure chain as needed.

1 bit for mux control, one bit for power domain override
- observe if domain is 'functionally' on (may be necessary for some TDRs)
-Override domain power on/off for test purposes
- Bits after the mux allows observation of mux control (did it achieve 'on')

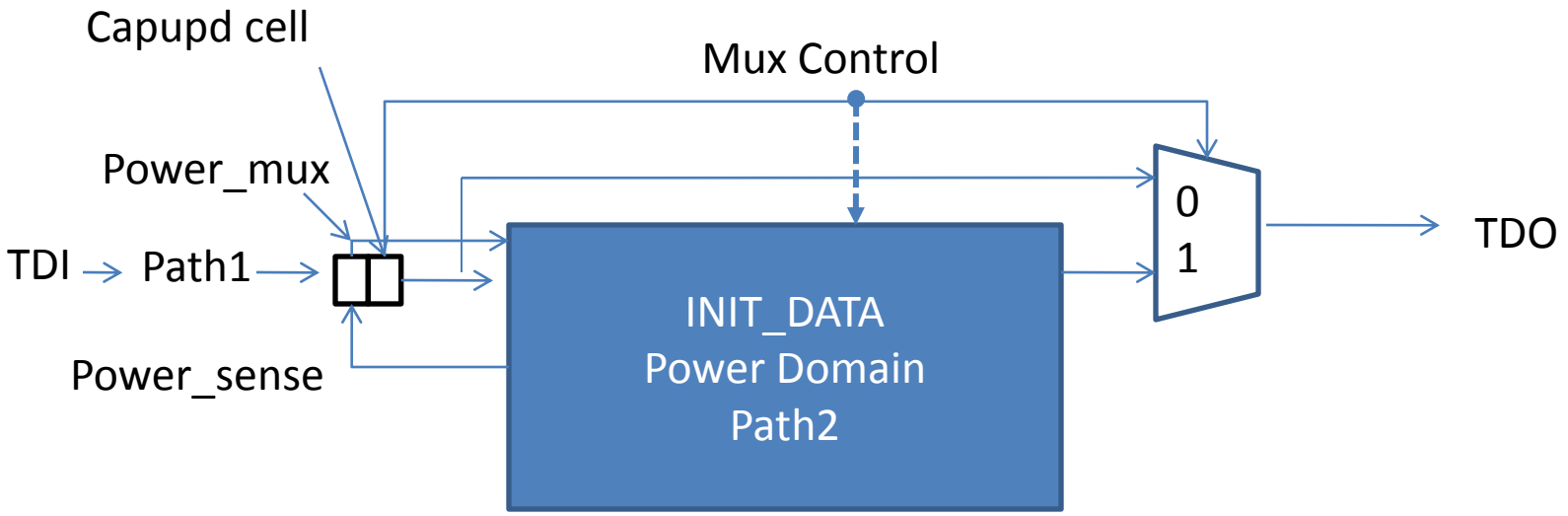


Or if WG prefers it could be done this way. Initial state (reset) is known
To be 'off/collapsed'.



1149.1 BSDL for mux segments

INIT_DATA bits within an IC



Rules: Mux and cells are always outside of Power Domain
Power mux cells control mux on 'other side' of domain prior to next Power Mux or TDO (No power_mux around another power mux)
2 bits, one cell controls mux and other is optional to override system which may have power domain off.

A capupd cell is one of the cells we have not allowed yet as a pre-defined Cell like NOPI or NOPO. CAPUPD captures from the update register. The mux Control is a critical signal that needs testability . One could argue that Power_sense Could be used instead of UPD as the capture value. One could also argue that The cells after the Path2 must be used for obtaining value from CAPUPD on mux control

It’s possible for 1149.1-2012 package file to specify the POWER_MUX

```
attribute REGISTER_MNEMONICS of  std_1149_1_2012 : package is
 “MUX    (On      (1) < chain segment is included >, " &
 "        Off     (0) < chain segment not included >), " &
 “POWER  (On      (1) < Domain is functionally on> ), " &
 “        Off     (0) < Domain is functionally off>), “ &
 “CONTROL (Override (1) < Test controls domain power > ), " &
 “         Normal   (0) < Power domain in normal mode> )“;
```

```
 attribute REGISTER_FIELDS of  std_1149_1_2012 : package is
 “powermux [2] ( "&
-- TDI
 "(DOMAIN  [1] IS  (0 )  DEFAULT(CONTROL(Normal)) “ &
 “                       RESETVAL(Control(Normal)), "&
 “                       CAPTURES(POWER(-)) ), "&
 "(MUX     [1] IS  (1 )  DEFAULT(MUX(Off) ), "&
 “                       RESETVAL(MUX(Off) ),”&
 “                       CAPUPD)”&
"      )";
```

1149.1 BSDL for mux segments

Use of POWERMUX in R_A

Consider if SERDESH is in a separate POWER Domain

```
attribute REGISTER_ASSEMBLY of chip_2011 : entity is
      "INIT_DATA ( "&
        -- TDI
        "(USING SerdesO),( i1 IS init_data DEFAULT.ALLBITS (CHPMFG(Test)) ), "&
         "(USING SerdesO),( i2 IS init_data DEFAULT.ALLBITS (CHPMFG(Test)) ), "&
         "(USING STD_1149_1_2012), “&
          “( mux1  IS  POWERMUX  ), "&
        "(USING SerdesH),( i3 IS init_data DEFAULT.ALLBITS (CHPMFG(Test)) ), "&
"(USING SerdesO),( i4 IS init_data DEFAULT.BSTERM  (BSTERM(CPflt))
DEFAULT.BSCM (BSCM(DC_CPL)) DEFAULT.BSSWING (BSSWING(1115mV)) ) "&
        --  TDO
        ")";
```

Mux1 will collapse from Mux1 to TDO.

Use of POWERMUX in R_A

Two powermux would allow a description of a segment which muxes prior to TDO
- powermux muxes at TDO or the next powermux

```
attribute REGISTER_ASSEMBLY of chip_2011 : entity is
      "INIT_DATA ( "&
        -- TDI
        "(USING SerdesO),( i1 IS init_data DEFAULT.ALLBITS (CHPMFG(Test)) ), "&
         "(USING SerdesO),( i2 IS init_data DEFAULT.ALLBITS (CHPMFG(Test)) ), "&
         "(USING STD_1149_1_2012), “&
          “( mux1  IS  POWERMUX  ), "&"(USING SerdesH),( i3 IS init_data
DEFAULT.ALLBITS (CHPMFG(Test)) ), "&
         "(USING STD_1149_1_2012), “&
          “( mux2  IS  POWERMUX  ), "&
        "(USING SerdesO),( i4 IS init_data DEFAULT.BSTERM  (BSTERM(CPflt))
DEFAULT.BSCM (BSCM(DC_CPL)) DEFAULT.BSSWING (BSSWING(1115mV)) ) "&
        -- TDO
        ")";
```

Mux1 will collapse from Mux1 to Mux2. Mux2 collapses from Mux2 to TDO.
The mux design must be such that the power domain INIT_DATA path is switched in/out at the point of 1) the next POWERMUX or 2) the end of the chain (TDO).