Proposal for IEEE 1500 Standard

Grammar and architecture support

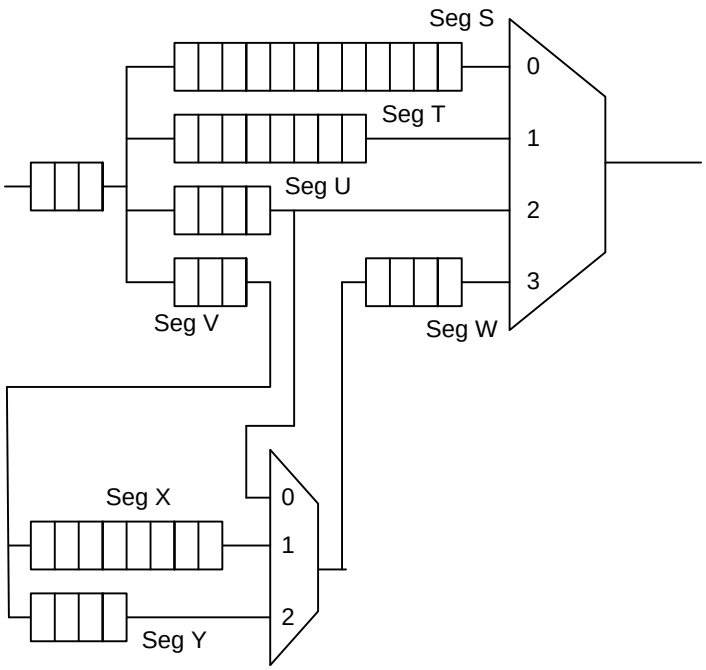
in IEEE 1149.1 BSDL

CJ Clark
Intellitech

<multi-bit select> :== SELFIELD LPAREN <extended field name> RPAREN
<segment association> :== SELECTVAL LPAREN <segment description>
                          { COMMA <segment description } RPAREN
<segment description> :== <target segment> LPAREN <oper val> RPAREN
<target segment> :== <extended field name> | <reg or seg name>

Note: <oper val> is not a mistake we have already defined it in R_C. The mnemonic allowed here shall be associated with the SELECT <extended field name>

<oper val> ::= <mnemonic pattern> | <binary pattern> | <hex pattern> | <decimal pattern>



I call the unlabeled 3 bit register as "A". 4:1 mux is M1, and 3:1 mux is M2
Note that in R_A we have:

<register assembly list>::= <reg or seg name> <left paren> <register assembly segments> <right paren>

We have a <reg or seg name> already, so rather than using SEGMENT, SEG_S is a register segment (it's not a whole TDR). Then those are used by the mux or instantiated in the R_A as something in the chain.

Attribute REGISTER_ASSEMBLY of mychip : entity IS

"SEG_S (" &
    "(S[12] NOUPD), "&
    "), "&

```
"SEG_T ( " &
    "(T[8] NOUPD  ),  "&
    "),  "&
"SEG_U ( " &
    "(U[4] NOUPD SHARED ),
    "),  "&
"SEG_V ( " &
    "(V[3] NOUPD),
    "),  "&
"SEG_X ( " &
    "(X[8] NOUPD),
    "),  "&
"SEG_Y ( " &
    "(Y[3] NOUPD),
    "),  "&
"SEG_LONG ( " &
    "( V1 is SEG_V ),
    "(M2 IS SEGMUX  SELECT(notdrawn)    "&   -- 3:1 mux
    " SegmentVal( SEG_U(0),SEG_X(1), SEG_Y(2) ),  "&
    "(W[4]  ) &
    "),  "&
```

So SEGMENTVAL is not pointing to named segments but actual register segments. Like "IS" it is the instantiation of a segment (little s) in a R_A description. I'm using it directly trying to avoid 'instantiating it' with … IS … as to me, if it is instantiated then it EXISTS and is in the scan path. I'm keeping R_F and instance names as we have already defined as the item in the TDI-TDO path.

The whole picture then is:

```
Attribute REGISTER_ASSEMBLY of mychip : entity IS

"WHOLE ( " &
    "(A[3] NOUPD),                                  "&  -- 3 bit register
    "(M1 IS SEGMUX  SELECT(notdrawn)        "&   -- big Mux
    " SegmentVal( SEG_S(0), SEG_T(1), SEG_U(2) SEG_LONG(3) ),  "&
     ")";
```

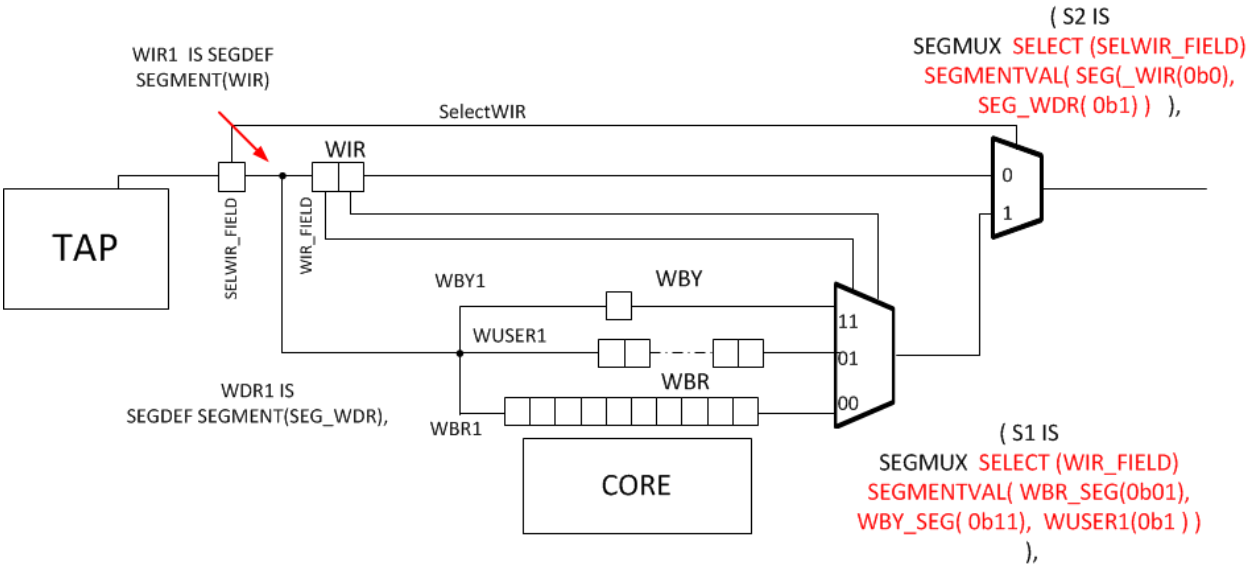
This avoids the duplication of the SEGMENT name and instance names. And lots of SEGDEF/SEGEND. Some may like this better as it is the shortest to date. And it does not create confusion on what is series and what is being selected. No change to anything we have. We allow SEGMENTVAL and SELECT to be on the SEGMUX which quickly tells the tool that this is a multi-bit select (and SEGSTART or SEGSEL is not required 'first').

"notdrawn" for the selector field means that I didn't see it in Ken's drawing. But that selector field can be anywhere outside of a what the MUX is switching.

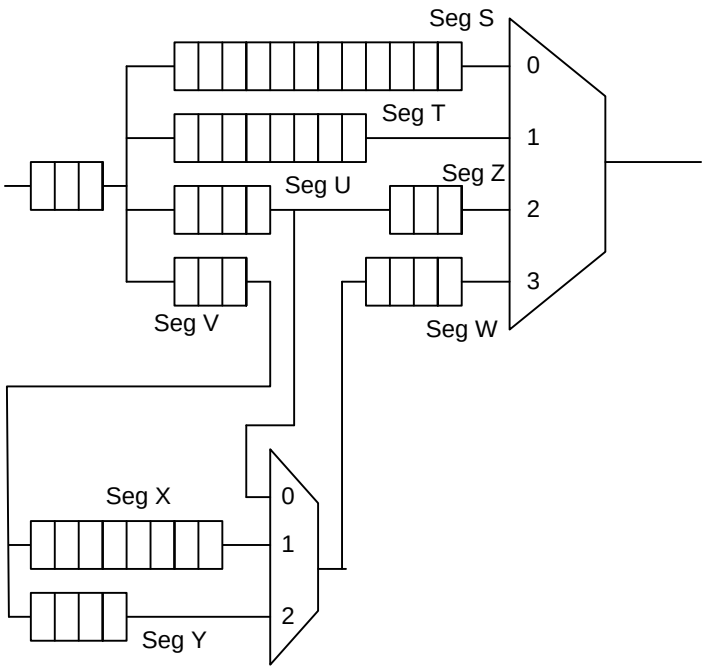
The simple 1500 drawing then is as follows (note this is not all of 1500 but the simple drawing of the basic 1500 concept. They are shy on examples in the standard

Syntactically and grammar checked

```
Attribute REGISTER_ASSEMBLY of mychip : entity IS

"SEG_WIR ( " &
"         ( WIR_FIELD[2] ResetVal(0b00) TAPReset  ) "&
"), " &
"SEG_WBY ( " &
"         ( WBY[1] ) "&
"), " &
"SEG_WBR ( " &
"         ( WBR[1] ) "&
"), " &
"SEG_WUSER ( " &
"         ( WUSER[20] ) "&
"), " &
"SEG_WDR ( " &
"(D1 IS SEGMUX SELECT(WIR_FIELD)    "&   -- 3:1 mux
    " SegmentVal( SEG_WBR(0),SEG_WUSER(1), SEG_WBY(3) ) ) "&

")";


Attribute REGISTER_ASSEMBLY of mychip : entity IS

"REG_1500 ( " &
    "(SELWIR_FIELD [1] ResetVal(0b0) TAPReset ),  "& -- first bit in chain

    "(D2 IS SEGMUX SELECT(SELWIR_FIELD)           "&
    "  SegmentVal(SEG_WIR(0b0), SEG_WDR(0b1))   ) "& -- last in chain
                                                      -- SEG_WIR or SEG_WDR
     ")";
```



Note that two muxes are in the description and two are in the schematic.




```
Attribute REGISTER_ASSEMBLY of mychip : entity IS

"SEG_S ( " &
    "(S[12] NOUPD ) "&
```

```
    "),  "&
"SEG_T ( " &
    "(T[8] NOUPD  )  "&
    "),  "&
"SEG_U ( " &
    "(U[4] NOUPD SHARED )
    "),  "&
"SEG_V ( " &
    "(V[3] NOUPD)
    "),  "&
"SEG_X ( " &
    "(X[8] NOUPD)
    "),  "&
"SEG_Y ( " &
    "(Y[3] NOUPD)
    "),  "&
"SEG_UANDZ ( " &
    "(i1 IS SEG_U),
    "(Z[4] NOUPD)
    "),  "&

"SEG_LONG ( " &
    "( V1 is SEG_V ),
    "(M2 IS SEGMUX  SELECT(notdrawn)    "&   -- 3:1 mux
    " SegmentVal( SEG_U(0),SEG_X(1), SEG_Y(2) )  ),"&
    " (W[4] )" &
    ");  "&


The whole picture then is:

Attribute REGISTER_ASSEMBLY of mychip : entity IS

"WHOLE ( " &
    "(A[3] NOUPD),                                  "&  -- 3 bit register
    "(M1 IS SEGMUX  SELECT(notdrawn)        "&   -- big Mux
    " SegmentVal( SEG_S(0), SEG_T(1), SEG_UANDZ(2) SEG_LONG(3) ) ) "&
     ")";
```

Note that there are two muxes in the schematic and two SEGMUX statements.
Note how the SHARED keyword on U would allow a tool to prevent access during mission mode operation. Unfortunately, there is not a mechanism to know what Design Specific Instructions are mission mode and what ones are not.

Rule changes

8.21.1

e) When a SEGMUX segment is encountered, all <register assembly segments> ordered before the SEGMUX segment and after the closest preceding SEGSEL or SEGSTART segment shall be excludable as a unit.

NOTE—The SEGSEL, SEGSTART, and SEGMUX are not part of the excludable segment; they are not excludable. These named segments are defined in the Standard BSDL Package (see **Error! Reference source not found.**)

This rule needs a tweak even if we do nothing. The <register assembly segments> is probably a misnomer as we are using segment to mean a specific thing. This may be better as <register assembly entries> or <register assembly instances> to avoid confusion with segments. (SEGSEL, SEGSTART and SEGMUX aren't named segments either, another confusing term). And it is more precise to refer to the 'instantiation' of SEGSTART… etc. as we don't just encounter a SEGSTART but an instance of the SEGSTART field is found in the R_A.

e) All <register assembly ~~segments~~ entries> between the instantiation of a SEGSTART or SEGSEL and the instantiation of a SEGMUX shall be wholly excludable as a segment.

NOTE—The SEGSEL, SEGSTART, and SEGMUX are not part of the excludable segment; they are not excludable. These register fields are defined in the Standard BSDL Package (see **Error! Reference source not found.**).

xxx) An instance of a SEGMUX in a REGISTER_ASSEMBLY not preceded by an associated instance of a SEGSTART or SEGSEL shall have both a SELECT and SEGMENTVAL keyword.

xxx) All <register assembly entries> defined in a <register assembly list> and instantiated by the inclusion of the associated <reg or seg name> in a SEGMENTVAL keyword shall be wholly excludable as a segment.

xxx) A <register field or instance> specified by a SELECT keyword shall not be included in a segment specified by the SEGMENTVAL keyword.

xxx) An instantiation of a SEGMUX with a SELECT and SEGMENTVAL keyword shall exist only in the REGISTER_ASSEMBLY of a design specific register.

- Note: Standard registers shall not have multi-bit selection via SELECT and SEGMENTVAL.

Note that at the same time, the rules structured like this, prevent, desirably in my view, DOMAIN and DOMAIN_EXTERNAL to be used with a SEGMUX that has a field selector. We're reserving SEGSEL for its ability to view the 'ready to scan'. (another approach is to force the LSB of a selector field to capture the 'ready to scan'.

R_A grammar to re-do:

<register assembly string>::= <quote> <register assembly list>
{ <comma> <register assembly list> } <quote>
<register assembly list>::= **Error! Reference source not found.** <left paren> <register assembly entries ~~segments~~> <right paren>
<register assembly entries ~~segments~~>::= <left paren> <register entry ~~segment~~ > <right paren>
{ <comma> <left paren> <register entry ~~segment~~ > <right paren> }

<register entry ~~segment~~ > ::= <instance and options> | <field and options> |
<boundary instance> | <using statement>

h) When appearing in the **BOUNDARY** register, all <register assembly segments> shall be a <boundary instance> or an instance of the DOMCTRL, SEGSEL, SEGSTART, or SEGMUX register fields defined in the Standard BSDL Package (see **Error! Reference source not found.**).

No changes needed for this proposal; however <register assembly segments> should be <register assembly entries>. "Appearing" should be dropped as well as prepositional phrase opening. Suggest:

h) All <register assembly entries> of a <register assembly list> with a <reg or seg name> specified as **BOUNDARY** shall be a <boundary instance> or an instance of the DOMCTRL, SEGSEL, SEGSTART, or SEGMUX register fields.

Note: These register fields are defined in the Standard BSDL Package (see **Error! Reference source not found.**).

i) When controlling excludable segments in the boundary-scan register, all necessary DOMCTRL and SEGSEL fields shall appear in the **BOUNDARY** register.

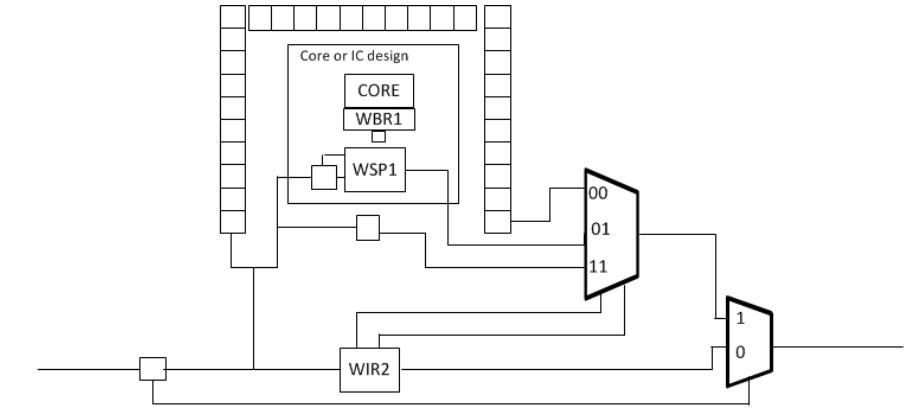
No change needed for this proposal. However, similar to above, it needs re-write for consistency with 1149.1 rules. Controlling is imprecise.

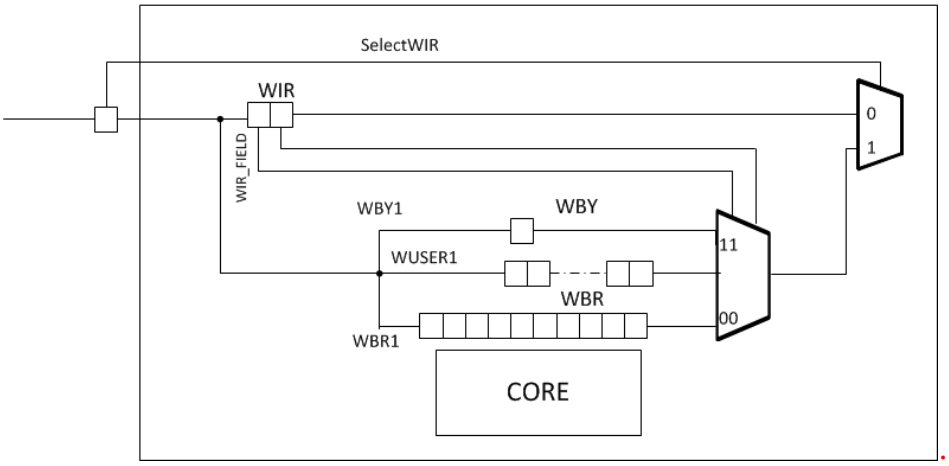
i) All domains and excludable segments in a <register assembly list> with a <reg or seg name> specified as **BOUNDARY** shall have instances of DOMCTRL and SEGSEL fields respectively in the same <register assembly list>.

j) When appearing in the **BOUNDARY** register, DOMCTRL and SEGSEL fields shall only control excludable segments in the boundary-scan register.

No change needed for this proposal. However, the rule needs clarification and an update.

j) All instances of DOMCTRL and SEGSEL fields in a <register assembly list> with a <reg or seg name> specified as **BOUNDARY** shall control excludable segments in the same <register assembly list>.





This is syntactically correct through parser.

```
Attribute REGISTER_ASSEMBLY of chip_2012 : entity IS
"SEG_WIR1    ( " &
"        ( WIR_FIELD1[2] ResetVal(0b00) TAPReset  ) "&
"), " &
"SEG_WBY1 ( " &
"        ( WBY[1] ) "&
"), " &
"SEG_WBR1 ( " &
"        ( WBR[1] ) "&
"), " &
"SEG_WUSER1 ( " &
"        ( WUSER[20] ) "&
"), " &
"SEG_WDR1 ( " &
    -- would like SEG_WIR1 here so fields don't have to be renamed
"(D1 IS SEGMUX SELECT(WIR_FIELD1)    "&   -- 3:1 mux
    " SegmentVal( SEG_WBR1(0),SEG_WUSER1(1), SEG_WBY1(3) ) ) " &

")"  ;


attribute REGISTER_ASSEMBLY of chip_2012 : entity IS
```

```
"REG_1500 ( " &
    "(SELWIR_FIELD1 [1] ResetVal(0b0) TAPReset ),  "& -- first bit in chain

    "(D2 IS SEGMUX SELECT(SELWIR_FIELD)           "&
    "   SegmentVal(SEG_WIR1(0b0), SEG_WDR1(0b1))  ) "&  -- last in chain
                                          -- SEG_WIR1 or SEG_WDR1
     ")";


attribute REGISTER_ASSEMBLY of chip_2012 : entity IS

"SEG_WIR2    ( " &
"        ( WIR_FIELD2[2] ResetVal(0b00) TAPReset  ) "&
"), " &
"SEG_WBY2 ( " &
"        ( WBY[1] ) "&
"), " &
"SEG_WBR2 ( " &
"        ( WBR[1] ) "&
"), " &
"SEG_WUSER2 ( " &
"        ( REG1 IS REG_1500 ) "&
"), " &
"SEG_WDR2 ( " &
            -- better to use SEG_WIR2 here in my view
"(D3 IS SEGMUX SELECT(WIR_FIELD2)    "&   -- 3:1 mux
    " SegmentVal( SEG_WBR2(0),SEG_WUSER2(1), SEG_WBY2(3) ) )  "&

")" ;


attribute REGISTER_ASSEMBLY of chip_2012 : entity IS

"REG2_1500 ( " &
    "(SELWIR_FIELD [1] ResetVal(0b0) TAPReset ),  "& -- first bit in chain
--  ideally this should be a segment or instance too
    "(D4 IS SEGMUX SELECT(SELWIR_FIELD2)           "&
    "   SegmentVal(SEG_WIR2(0b0), SEG_WDR2(0b1))   ) "&  -- last in chain
                                                    -- SEG_WIR1 or SEG_WDR1
     ")";
```

Added keywords. Note this has been a good exercise to flesh out many missing and incorrect sections of the draft regarding BSDL. We currently do not have rules for the many combinations below which are illegal even without this proposal. For instance, an instance of a SEGSEL cannot have a NOPI, NOPO, NOUPD, PULSE1, PULSE0 or SHARED. Similarly an instance of a SEGSTART cannot have these keywords plus any of the reset keywords. An instance of a SEGMUX shall not have any of those keywords either.

NOPI The TDR contents do not change during the *Capture-DR* TAP controller state (does not capture.) See Figures **Error! Reference source not found.** and **Error! Reference source not found.** for examples of a TDR cell that would be described by this keyword.

NOPO The TDR contents do not affect any test or functional logic. See Figures **Error! Reference source not found.** and **Error! Reference source not found.**, which would be described by this keyword if the optional PO was not implemented.

NOUPD There is no update stage; parallel outputs, if any, of the register change during the *Shift-DR*, not the *Update-DR* TAP controller state. If the field also has a reset assignment, then the reset applies to the shift stage. **NOPO** implies **NOUPD**. See Figures **Error! Reference source not found.** and **Error! Reference source not found.**.

MON The shift-capture stage captures the value in the Update stage, allowing verification of the value currently being driven on PO. **MON** implies **NOPI**. **MON** is incompatible with **NOUPD** and **NOPO**. See Figures **Error! Reference source not found.** and **Error! Reference source not found.**.

PULSE1 When a '1' is shifted into the cell, the output will go high (1) for a single TCK cycle after the *Update-DR* TAP controller state and then return to low (0). **PULSE1** is incompatible with **NOUPD** and **NOPO**. See Figure **Error! Reference source not found.**.

PULSE0 When a '1' is shifted into the cell, the output will go low (0) for a single TCK cycle after the *Update-DR* TAP controller state and then return to high (1). **PULSE0** is incompatible with **NOUPD** and **NOPO**. See Figure **Error! Reference source not found.**.

SHARED The register field cells are shared with mission mode logic and if scanned during mission mode could interfere with mission mode operation.

PORRESET The register contents and parallel outputs change in response to the on-chip POR signal, also used to reset the TAP controller (see **Error! Reference source not found.**).

TRSTRESET The register contents and parallel outputs change in response to the TRST* TAP port, also used to reset the TAP controller (see **Error! Reference source not found.**).

TAPRESET The register contents and parallel outputs change when the TAP controller enters the *Test-Logic-Reset* state, regardless of the state of the TMP controller, if it is provided. Either an on-chip POR or TRST* TAP port assertion will also force this reset.

CHRESET The register contents and parallel outputs change when the TAP controller enters the *Test-Logic-Reset* state and the Persistence controller is in the *Persistence-Off* state. Either an on-chip POR or TRST* TAP port assertion will also force this reset. Note that the use of this keyword is an error if the optional TMP controller is not provided.

DOMAIN The associated fields control or are controlled by the named domain, which has an on-chip controller.

DOMAIN_EXTERNAL The associated fields are controlled by the named domain, which does not have an on-chip controller.

SEGMENT Associates by the specified name the SEGSEL with a SEGSTART when the SEGSEL is not in the same TDR with the excluded segment.

SELFIELD Specifies the <register field>, <reg or seg name> or <extended field name>that controls a SEGMUX. SELFIELD is always used with SEGMENTVAL and shall not be used with SEGMENT, DOMAIN or DOMAIN_EXTERNAL.

SEGMENTVAL Associates a register segment specified by a <reg or segname> with a <mnemonic identifier> , BINARY_PATTERN, DECIMAL_PATTERN or HEX_PATTERN. The value specified represents the value the SELECT register must have to select the specified register segment.

Figure from IEEE 1500 (to do).

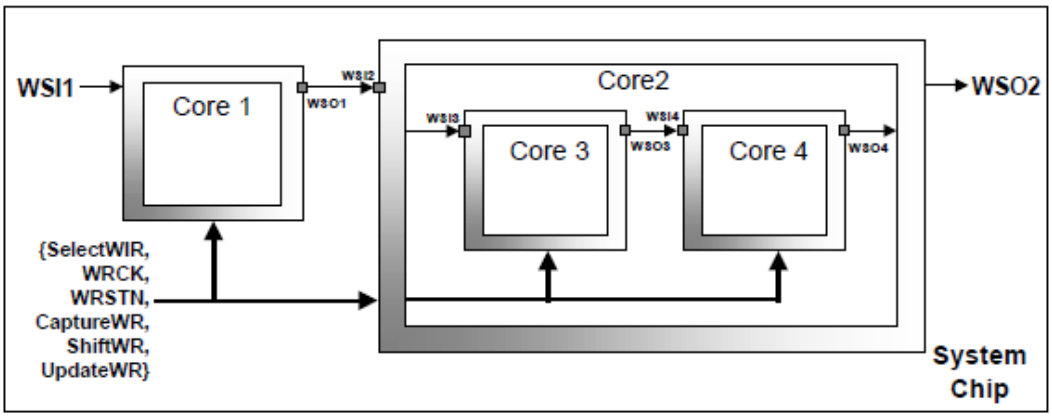


Figure 37—Serial WSPs with subcores