|  |  |
| --- | --- |
| [MOR-2089] [IDS issues to feedback to Agnisys](http://dwjira/browse/MOR-2089) Created: 13/Mar/20  Updated: 10/Jun/20 | |
| **Status:** | Backlog |

|  |
| --- |
|  |

|  |  |
| --- | --- |
| **Description** |  |

|  |
| --- |
| Task to track IDS issues to raise with Agnisys . This task will be updated as the review of regmap generation and post-processing progresses over the coming days.  List all reasons we post-process output of IDS.   1) Status bits :  excessive sync registers used for edge detection   For Morpheus we took the IDS defaults for interrupt sync regs , resulting in a circuit  as follows      We then post process the RTL to remove FF0 and FF1 from above circuit and connect the interrupt source directly to FF2 input in above circuit.    Using {intr.detect\_nometa=true}  property on status bits reduces the default 3 registers (meta + 2 sync for edge detection)  on h/w inputs to 2 ( both for edge detection).  However a single register would do when edge detecting a synchronous level signal.    setting intr.detect\_nometa results in the following circuit produced by IDS ( from "Interrupts section" of IDS docs)      We would like IDS to effectively remove FF1 in above diagram, and connect the interrupt source to FF2 input ,  when nometa is set.  This reduces number of regs and improves interrupt latency.  Currently we have to post process the outputs of IDS to resolve this.   2) hdl\_paths for {registered=false; volatile=true} register fields missing from XML hdl\_paths are not present in ipxact XML output by IDS for registers with property {registered=false; volatile=true}.    We see this problem in our EVENT\_COUNT[0-7]  registers, which  allow the user to read registers that are implemented outside of the regmap RTL via the regmap .  The default values would be correct here for our application . They provide an HDL path to the wires (external register outputs )  that are read via the regmap and provide backdoor read and wait for change functionality in UVM for these signals.  As they are not present in the XML we need to restore them through post processing.     3) Decoding of memories is external to IDS generated RTL regmap block We've found no way of defining memory locations within IDS XLSX.  Using  external attributes for memory in IDS results in a regmap that decodes/mediates transfers to an external memory block, via the APB bus.  In our design , access of memories is not handled by IDS generated RTL,   however it would be useful to specify block locations/sizes/descriptions for documentation and header files using the single XLSX source.  Currently we add the memory addresses to the header files generated by IDS and then add memory info to the RALF during postprocessing to all memory access via UVM RAL.       4) Spaces in embedded directives result in broken descriptions containing  "{ " in XML   We are setting IDS variables in descriptions  e.g.  { aliased = true }    .  Since updating to new IDS, these must contain no spaces inside the brackets,  otherwise parts of these elements are written to the XML description fields.  The brackets ( normally  opening but not closing) , then make the dut.ral produced by ralgen ( a TCL format , where { } have special meaning ) unparseable.  This started happening when we updated IDS in March 2020.     5) HDL paths output by IDS to XML are abs paths in reg block   IDS puts the full HDL path of fields in <snps:hdl\_path>  elements.  Synopsys ralgen forms the hdl\_path of an field by concatenating the hdl\_path of the block, reg and field.  From ralgen user guide ( sec "Generated Back-Doors)  The generated backdoor simply concatenates the path elements specified in the individual hdl\_path attributes to form the complete path to the target register or memory. For example, the RALF file shown in Example 3-1 yields the path S1\_TOP\_PATH.b1\_i.dec.r1\_reg to the register r1.    If no <snps:hdl\_path> tag is found  in the XML for block,reg, or field, then ralgen uses the name of the block/reg/field in the construct described above.  For blocks and regs , using the name( i.e. the default when no snps:hdl\_path is specified ) produces the correct result.   For fields, we need to specify the naming convention  used in the RTL generation of the register bits ( i.e. field\_name + "\_q" ).    The error in IDS is to prepend the wrapper name +  block  + reg components to the field hdl\_path.  Only the field component of the path should be specified.    e.g.  <snps:hdl\_path>dut\_ids.main\_idsinst.DEV\_ID\_RIDTAG\_q</snps:hdl\_path>  Should be  <snps:hdl\_path>DEV\_ID\_RIDTAG\_q</snps:hdl\_path>     By changing all snps:hdl\_path  tags as above,  synopsys ralgen will generate correct HDL paths for our system descriptions.     6)  Bug in regmap.v RTL manifests when setting aliastab reg width to 8 bits if aliased reg is larger than 8 bits   We have some alias tab regs that only need 1 byte of storage to accommodate the fields in the aliased register  that require aliasing .  where the aliased register is larger than 8 bits.   i.e.  only a subset of the aliased register fields need to be in aliastab,  and the total space required to store them in the aliastab is one byte.  In our example,  we find that the RTL generated by IDS works when the aliastab register is 16 bits wide,  with only the lower 8 bits implemented .  In this working case,  that aliased register block instantiation port connections concerned are as follows:  .PLL\_CAL\_A\_wr\_valid\_in0(aliastab\_idsPLL\_CAL\_A\_wr\_valid0),  .PLL\_CAL\_A\_wr\_valid\_out0(aliastab\_idsPLL\_CAL\_A\_wr\_valid0),     However, when we change the aliastab reg ( PLL\_CAL\_A ) width to 8 bits,  IDS generates the port connections as follows  .PLL\_CAL\_A\_wr\_valid\_in0(aliastab\_idsPLL\_CAL\_A\_wr\_valid0),  .PLL\_CAL\_A\_wr\_valid\_out(aliastab\_idsPLL\_CAL\_A\_wr\_valid),      In this case, the wire aliastab\_idsPLL\_CAL\_A\_wr\_valid0  is undriven.   This makes it impossible to update the aliastab register over the APB bus.  If we correct this error , i.e. change the  PLL\_CAL\_A\_wr\_valid\_out  connections to the following  .PLL\_CAL\_A\_wr\_valid\_out0(aliastab\_idsPLL\_CAL\_A\_wr\_valid0),  then our aliastab functionality is restored. |

Generated at Thu Feb 04 10:48:37 GMT 2021 by Steve Murphy using Jira 8.6.0#806000-sha1:737bcc05e3687605702e76b312ace69277496331.