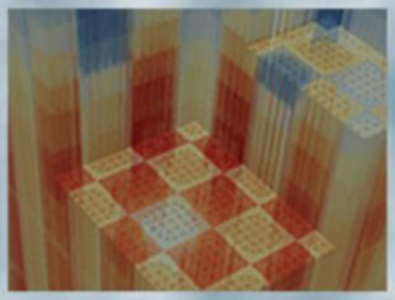
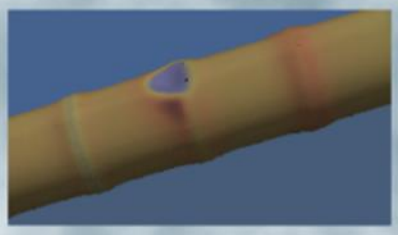
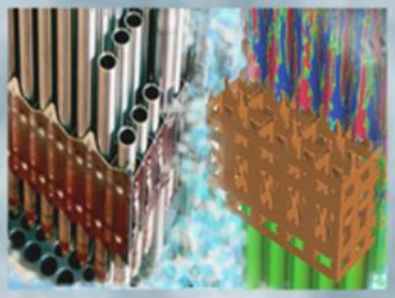
CASL-U-2016-1099-000







Standalone BISON Through VERA: User's Manual

Shane Stimpson
Oak Ridge National Laboratory

June 3, 2016

REVISION LOG

Revision	Date	Affected Pages	Revision Description
0	06/03/2016	All	Original Report

Document pages that are:

Export Controlled NO

IP/Proprietary/NDA Controlled NO

Sensitive Controlled NO

Approved for Public Release YES

CONTENTS

ACRONYMS

ASCII	American Standard Code for Information Interchange
CASL	Consortium for Advanced Simulation of Light Water Reactors
CS	core simulator
CSV	comma-separated value
CTF	COBRA-TF
CZP	cold zero power
EFPD	effective full power day
HDF	hierarchical data format
HFP	hot full power
HZP	hot zero power
IFBA	integral fuel burnable absorber
INL	Idaho National Laboratory
JFNK	Jacobian-free Newton-Krylov
MC	Monte Carlo
MOOSE	Multiphysics Object Oriented Simulation Environment
MPI	message passing interface
MW	megawatt
ORNL	Oak Ridge National Laboratory
SNA	Supercomputing in Nuclear Applications
VERA	Virtual Environment for Reactor Applications
WBN1	Watts Bar Nuclear Unit 1
XML	extensible markup language

1. INTRODUCTION

Activities to incorporate fuel performance capabilities into the Virtual Environment for Reactor Applications (VERA) are receiving increasing attention [1,2,3]. The multiphysics emphasis is expanding as the neutronics (MPACT) and thermal-hydraulics (CTF) packages are becoming more mature. Capturing the finer details of fuel phenomena (swelling, densification, relocation, gap closure, etc.) is the natural next step in the VERA Core Simulator (VERA-CS) development process since these phenomena are currently not directly taken into account. While several codes could be used to accomplish this, the BISON fuel performance code [4] being developed by the Idaho National Laboratory (INL) is the focus of ongoing work in the Consortium for Advanced Simulation of Light Water Reactors (CASL). Built on INL's MOOSE framework [5], BISON uses the finite element method for geometric representation and a Jacobian-free Newton-Krylov (JFNK) scheme to solve systems of partial differential equations for various fuel characteristic relationships. There are several modes of operation in BISON, but, for this work, it uses a 2D azimuthally symmetric (R-Z) smeared-pellet model.

This manual is specific to the procedure pertaining to the standalone BISON one-way coupling from VERA-CS. Section 2 covers the process to generate BISON inputs based on input/output information from VERA-CS, starting from the American Standard Code for Information Interchange (ASCII) VERA input [6] through generation of a separate BISON input file for each rod in the problem. Section 3 provides some brief background and guidelines on running the inputs. Section 4 covers the current post-processing capabilities which heavily leverage the VERAView visualization tool [7].

2. GENERATING STANDALONE BISON INPUTS

There are four steps to generate the standalone BISON inputs:

1) Create a VERA ASCII input
2) Convert the ASCII input to extensible markup language (XML) (and XML to CTF)
3) Execute VERA-CS to generate hierarchical data format (HDF5) output files
4) Run XML2MOOSE to produce BISON inputs

2.1 Creating a VERA ASCII Input

2.1.1 General Overview

The VERA ASCII input is intended to be set up quickly and easily, even for complex reactor designs. A detailed overview of the input specification will not be covered here, but it can be found in the VERAIn Common Input Manual [6]. However, enough background will be presented here to provide a general idea of the key components in the VERA ASCII input.

The top block(s) in each file are the STATE blocks, which specify a number of state-specific parameters such as power (%), soluble boron concentration (ppm), coolant inlet temperature, system pressure, and depletion time steps. Very simple histories can be specified using only one STATE block, but several blocks are typically used to capture various changes throughout the cycle. For example, in simulating Watts Bar Nuclear Unit 1 (WBN1), 30–40 statepoints is the typical number per cycle [8]. A new input file is required per cycle, which will become more apparent in the next section when the shuffling specification is covered.

Next, the CORE block outlines the general core shape, the layout of assemblies (using the ASSEMBLY indexes specified in subsequent ASSEMBLY blocks), the layout of burnable poison configurations (using indexes from the INSERT block), and the control rod bank layout (using

indexes from the CONTROL block). Also included here are the rated core power and flow, along with the specifications for upper and lower core plates and reflector regions. Lastly, various materials definitions can be included here to be used in the ASSEMBLY blocks.

The ASSEMBLY blocks define the pin cell and lattice geometries used to construct axial stacks of lattices comprising each assembly. The pin cell specifications include all radii (fuel pellet, inner clad, outer clad, etc.) and material layouts. Lattice specifications include the 2D radial layouts of the pin cells in each lattice. These are stacked together with axial bounds for each region to construct the 3D assembly. Also included in this block are the spacer grid specifications in which the type (such as Inconel or Zircaloy) and midpoint of each grid are stated.

Blocks are also used to outline code-specific information (MPACT, COBRATF, and BISON). Later in this manual, some guidance will be provided on what can be included in the BISON block. Details on what the MPACT and COBRATF blocks can contain are defined in the VERAIn Manual [6].

2.1.2 Shuffling Specification

Since the shuffling specification through VERA is a significant part of multicycle simulation, it is outlined here. As mentioned above, a separate input file is necessary for each cycle depletion being executed. Furthermore, separate files are necessary for each shuffling procedure between cycles. Restart files are used to write the isotopics at the end of each cycle; these are then read in and shuffled based on the shuffle_label map specified in the STATE block of the shuffle input file. Once the shuffle has been completed, a new restart file is written and used as input for the next cycle. Figure 1 shows a representative shuffle_label specification in which fuel from the previous cycle is specified using the X/Y label <A-R>-<1-15> (corresponding to its location in the previous cycle) and "+" indicates the location of fresh fuel. Additionally, any label with an integer prefix indicates that the assembly was last present in a cycle other than the previous one. For example, 3B-10 indicates the assembly was from the B-10 location in cycle 3. Note that this is not intended to be a real shuffle layout, merely an example.

```
shuffle_label            3B-10 4G-2  4H-1  4A-11 4H-13 4M-2  3K-14
             L-1   P-6   +     +     +     +     +     +     B-12  F-3
      K-2    +     +     +     F-14  +     N-4   +     J-4   +     +     +     P-10
      R-7    +     R-9   L-10  +     A-5   H-14  J-1   +     F-2   N-14  +     G-15
3N-6  +      +     B-7   +     R-11  +     H-3   +     D-3   +     K-1   +     +     3J-6
4A-6  +      C-8   +     C-14  C-3   R-5   +     E-1   P-7   F-13  +     P-13  +     4A-8
4C-2  +      +     D-13  +     P-12  +     F-7   +     C-12  +     P-4   +     +     4G-12
4P-11 +      K-9   M-13  B-4   +     M-7   E-15  N-13  +     B-5   G-10  G-6   +     N-8
4A-7  +      +     L-14  +     K-7   +     R-10  +     M-3   +     G-1   +     +     4D-7
4N-10 +      K-3   +     B-13  E-14  H-15  +     M-14  B-11  N-3   +     L-15  +     4J-2
3C-10 +      +     D-2   +     J-15  +     E-2   +     F-11  +     C-6   +     +     3J-12
      P-9    +     M-9   G-4   +     F-15  C-13  A-9   +     R-8   B-9   +     D-9
      F-9    +     +     +     L-2   +     E-6   +     P-3   +     +     +     N-2
             N-12  P-5   +     +     +     +     +     +     +     C-4   J-14
                         3K-13 4B-3  4J-10 4G-14 4B-6  4D-14 3K-5
```

Figure 1. Representative Shuffle Map

Additionally, an "op_date" indicating the date operation began/ended for the cycle is specified. These dates are used to determine the shutdown time during the shuffling procedure, and isotopes are decayed accordingly. The cycle start date is specified in the CORE block, and the end date is specified in the last STATE block.

For a basic example of how the shuffling procedure works, please consult the "bison_from_vera_multi_qtr_shuffle_mir" and "bison_from_vera_multi_qtr_shuffle_rot" regression tests available in MOOSEExt/test, with mirror and rotational symmetry, respectively. These tests demonstrate the shuffling capability for three cycles using a small test core. Further details of these tests are provided in Sect. 2.5.

2.1.3 The BISON Block

The BISON block in the VERA input is where many of the values from the template can be updated, but it also includes several other important flags. Below is a sample BISON block:

```
[BISON]
  solve_type standalone
  fuel_pin_input_file_template = ../bison.template
  non_fuel_pin_input_file_template = ../bison.template
  power_file = mpact.h5
  mesh_type = smeared_pellet
  mesh_clad_bot_gap_height = 1.52e-3
  executioner_start_time = 0.0
  executioner_end_time = 2e5
  executioner_dtmin = 0.1
  bcs_plenumpressure_plenumpressure_startup_time = 10800.0
  functions_coolant_pressure_ramp_x = 0 10800.0
  functions_coolant_pressure_ramp_y = 0 1.0
  bcs_plenumpressure_plenumpressure_initial_pressure = 1.8e+06
  bcs_plenumpressure_plenumpressure_initial_pressure_ifba = 0.8e+06
  outputs_file_base = output
```

To cover some of these important flags, the discussion begins with `solve_type`. Valid entries for this include STANDALONE, TIAMAT, and TIAMAT_INLINE. This effectively directs XML2MOOSE to the mode for which inputs should be parsed. Not surprisingly, the mode for this manual is STANDALONE. Next the flags for the template files are generated. The standalone mode does not simulate guide tubes or burnable poison inserts, so the non-fuel template is not used and can point to the same template as for fuel. Tiamat, however, uses a different template when simulating guide tubes.

The next three flags are extremely important. `power_file` lists the HDF5 output files produced from running the cycle depletions (as will be generated in the process outlined in Sect. 2.3). Both relative and absolute paths can be used. `cycle_xml` lists the XML files for each cycle, and shuffle_xml contains the XML files used to execute the shuffling (from which the shuffle maps are pulled). Generation of XML files is covered in Sect. 2.2. The files for all cycles should always be entered consecutively, without skipping any. If results are desired for Cycle 7, enter all files for Cycles 1–7. XML2MOOSE will use the shuffling maps to determine the appropriate data to pull from each file.

The final described here is the `only_cycle` flag, in which only the BISON inputs for the specified cycle are generated. However, the history of the rods from previous cycles will still be constructed as appropriate. If this flag is not indicated, XML2MOOSE will generate files for all cycles. At present, the `only_cycle` flag is the recommended mode because it makes it easier to verify the number of inputs generated. It also makes the number of files easier to manage. In simulating WBN1, for example, there are roughly 15,000 input files generated per cycle. If several cycles are generated at once, the number of files created can be overwhelming to manage.

All other flags can be traced to locations in the BISON template file (see Sect. 2.4.1).

2.2 Converting the ASCII File to XML

Once the VERA ASCII input file has been created, it must be converted to XML using the react2xml script:

```
react2xml.pl vera.inp vera.xml
```

Codes such as MPACT can read the XML file directly, while codes like CTF and BISON require an additional preprocessor step to convert the XML file to a format they can read. The preprocessor for BISON (XML2MOOSE) will be covered in Sect. 2.4. While the preprocessor for CTF (XML2CTF) will not be covered in detail, it should be kept in mind when preparing to run VERA-CS, and it can be run similarly:

```
xml2ctf --xmlfile=vera.xml
```

Because separate files are necessary for each cycle depletion and shuffle, this process must be repeated for each file.

2.3 Running VERA-CS

Once all of the appropriate input files are preprocessed, they can be executed with MPACT/CTF to simulate the reactor through the cycle depletion. At the end of each cycle, separate executions are necessary to complete the shuffle procedure and move to the next cycle. Output from these simulations will be HDF5 files that contain the pin power and moderator temperature distributions for each rod, usually with 40–60 axial planes of resolution. These data are used by XML2MOOSE to construct each BISON input file.

It is worth noting that XML2MOOSE requires each HDF5 file to be of a complete cycle. Many cycle depletion cases require the use of at least one restart file to complete the simulation. The HDF5 files produced from the restarted cases must be appended together. This can be easily accomplished using HDF5 commands [9]. One caveat to this is that VERA-CS reports power for each state on the HDF5 file, but the first statepoint reports the raw power in megawatts (MW), whereas subsequent statepoints report percent power. When appending HDF5 files together, it is important to ensure that the final appended file has only raw power for the first statepoint, as if it ran to completion without a restart. This requires changing the value from MW to % as appropriate.

2.4 XML2MOOSE

2.4.1 The BISON Template

The BISON template file is a crucial part of this process, and it is very important to understand what contained in the template and how it can be changed from the VERA input. The template can be found in the MOOSEExt/test directory as `bison.template`, and it contains several different categories of data:

1) VERA_DEFINED
2) VERA_MODIFIABLE
3) VERA_PREPARED
4) TIAMAT
5) BISON

VERA_DEFINED data are specified exclusively by XML2MOOSE and are not adjustable through the BISON block. These would include things such as geometry specification, fuel density/porosity, and other system characteristics.

VERA_MODIFIABLE inputs are data that can be manipulated in the BISON block. If not specified there, they either have a default value or they are omitted entirely. Each modifiable input has an additional flag indicating the variable in the BISON block that can be used to change the value. For example, `mesh_nx_p` (also show below as in the template) can be used to change the number of radial elements in the fuel.

```
#VERA_MODIFIABLE as mesh_nx_p  (omitted if file is specified)
```

One can set the value of `mesh_nx_p` in the BISON block, and it will be populated into the template. The an additional comment—"(omitted if file is specified)"— indicates that this flag is deleted from the input when an external mesh file is specified, as it completely controls the mesh specification.

VERA_PREPARED items are a combination of the VERA_DEFINED and VERA_MODIFIABLE. If no flag is entered from the BISON block, VERA will automatically populate it with a value determined from the input, but VERA will not use a default value. For example, the simulation end time is determined by the number of states and depletion specification in the input, but this can be overridden to truncate the simulation. This is not necessary for most modes of operation, but it useful for testing.

Data flagged as TIAMAT are only used when XML2MOOSE is generating inputs for TIAMAT; otherwise they are deleted from the template.

BISON inputs are only used for standalone BISON; otherwise they are omitted. Some flags will behave differently if they are being used for standalone BISON or TIAMAT. This behavior is documented in the template.

2.4.2 Execution

Most of the work is performed in the XML2MOOSE preprocessor step, particularly creation of the BISON input files. Using the same XML file that was used to perform the VERA-CS neutronics/thermal-hydraulics simulations with an additional BISON block, XML2MOOSE can be run as in the following command, where the XML file created is vera.xml:

```
xml2moose -c vera
```

This will create separate BISON inputs for each rod by populating the BISON template file.

2.4.3 File Naming Convention

Each file created by XML2MOOSE will conform to the naming convention given below to convey the starting cycle index and location in the core:

```
vera_C03_ASSYB097_P009_6.5.1.4.5.6.bison.i
```

where `vera` indicates the base filename executed with XML2MOOSE, `C03` indicates that the rod originated from cycle 3, `ASSYB097` indicates the assembly name in the VERA input is ASSYB

and it starts at index 97, `P009` means it is at pin index 9, and `6.5.1.4.5.6` are the corresponding cell indexes from the VERA input that comprise the axial stack of the rod.

The assembly indexes for a core such as WBN1 with 193 assemblies is shown below. Future modifications may replace the assembly index with the X-Y label, which might make identification easier.

			1	2	3	4	5	6	7					
		8	9	10	11	12	13	14	15	16	17	18		
	19	20	21	22	23	24	25	26	27	28	29	30	31	
	32	33	34	35	36	37	38	39	40	41	42	43	44	
45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69	70	71	72	73	74
75	76	77	78	79	80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99	100	101	102	103	104
105	106	107	108	109	110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134
135	136	137	138	139	140	141	142	143	144	145	146	147	148	149
	150	151	152	153	154	155	156	157	158	159	160	161	162	
	163	164	165	166	167	168	169	170	171	172	173	174	175	
		176	177	178	179	180	181	182	183	184	185	186		
			187	188	189	190	191	192	193					

Fig. 2. Assembly indexing with a jagged core.

Similarly, the corresponding figure of pin indexes in a 17 × 17 assembly is shown below. This indexing is more straightforward if there are no jagged boundaries, which are present in a core layout for the assembly index.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85
86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102
103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136
137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153
154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170
171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187
188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204
205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221
222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238
239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272
273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289

Fig. 3. Pin indexing in an assembly.

2.4.4 Imposed Transition Times

Because BISON can encounter convergence issues if simulation parameters change too rapidly, several restrictions are imposed to ensure that there is ample time when transitioning, particularly between zero- and full-power conditions. Initially, all rods start at cold zero power (CZP) and transition to (HZP) conditions over an imposed 3-hour time frame. From HZP, the transition to hot full power (HFP) occurs over a 24-hour period. From here, the reactor operates as specified until the end of the cycle is reached, when it transitions back to HZP over 24 hours and then to CZP over 3 hours, where it remains for the duration of the cycle outage. This process is repeated for the number of cycles required.

Sometimes an instantaneous power change is imposed in VERA-CS. For example, in WBN1 Cycle 1, there is a transition near the end of the cycle from 100% power to 86.9% power. It is held there for a short time, and then it returns to 100%. A similar 24-hour period is imposed in BISON to robustly handle this transition.

It is worth noting that if explicit startup/shutdown periods are more explicitly modelled in VERA-CS, these will also be appropriately reflected in the BISON power history.

2.5 Examples and Regression Tests

Six regressions tests are currently used to ensure performance of this capability, and they are highly recommended as a starting point for examples of how to use it. For this reason, each test will be briefly described.

2.5.1 bison_from_vera

The bison_from_vera test is the simplest regression test, as it is a basic single pin case that was run out to 300 effective full power days (EFPDs). It has a 3.1% ^{235}U enrichment and a simple power history of 95% until it reaches 10 EFPDs. Then it goes up to 110% until it reaches 100 EFPDs. At that point it operates at 120% until it reaches 300 EFPDs. While this power history is unlikely, it is the way to test the capability and ensure that power changes are accounted for correctly.

2.5.2 bison_from_vera_ifba

This test builds on the previous test by adding an integral fuel burnable absorber (IFBA) coating to the solid fuel pellets. It also includes annular fuel pellets in the blanket regions. Currently, BISON does not account for annular pellets with the internal mesh generator, so the preprocessor ensures that the volume of these annular pellets is accounted for in the plenum.

By comparing the BISON inputs generated for this and the previous case, the minor differences imposed by being identified as an IFBA rod can be seen, such as the inclusion of the `he_prod` block in the input.

2.5.3 bison_from_vera_multi

bison_from_vera_multi is a 3 × 3 cross core with 3 × 3 pin assemblies, and it simulates the core without shuffling over three cycles. The depletion inputs for each cycle are included (c1.inp, c2.inp, and c3.inp), as well as the corresponding HDF5 files. XML2MOOSE is run using the bison.vera input, where the BISON block can be seen to reflect these cycles accordingly.

2.5.4 bison_from_vera_multi_qtr_shuffle_mir

This test also uses 3 × 3 pin assemblies, but in a full 3 × 3 core layout, and it simulates three cycles, including shuffling. From c3_shuffle.vera, it can also been observed that one assembly (1A-3) skips

cycle 2 and comes from cycle 1. This test uses mirror symmetry, and a later test uses rotational symmetry.

2.5.5 bison_from_vera_multi_qtr_onlyC3

This test is identical to bison_from_vera_multi_qtr_shuffle_mir except with the `only_cycle` flag included to restrict the input generates only the files pertinent to cycle 3.

2.5.6 bison_from_vera_multi_qtr_shuffle_rot

This is the last test, and it is identical to bison_from_vera_multi_qtr_shuffle_mir, but with rotational boundary conditions along the symmetry line.

3. EXECUTING BISON

With several thousand files generated by XML2MOOSE, it is a difficult task to execute them all. One possibility is to run them all in tandem using the MOOSE MultiApps capability by creating each pin as a separate MultiApp. However, there is an outstanding bug in BISON that prevents all outputs from being written when being run in this manner. The most success achieved at this point was by dividing the rods into batches and running them individually. For example, for each cycle in WBN1, 14,784 BISON inputs are created. These were divided into 616 jobs of 24 rods each, and each was executed using 12 message passing interface (MPI) processes in a total runtime of approximately 35,000 core-hours on Falcon. However, other systems have restrictions that will prevent this approach from being successful. On Titan and Eos, users are limited to two running jobs at once, so queueing 616 separate jobs is impractical. Also, sometimes there are restrictions on the number of tasks that can be combined into one job. Titan and Eos have a limit of 100 aprun (equivalent of mpirun) calls per job, so no more than 100 rods can be run in a single job.

Running all of the rods created by XML2MOOSE may be the most difficult task involved in this process. Future development to address some of these issues is necessary before considering widespread deployment.

4. POST-PROCESSING RESULTS

Once all of the BISON cases have been executed, results can be gathered and visualized using VERAView [7]. A post-process program called `bison_post` can be called and will conglomerate the output data onto the HDF5 file(s) specified by `power_file`. `bison_post`, which is currently limited to reading data from the output comma-separated value (CSV) files that contain limited data related to maximum, minimum, average, and integral quantities of interest. An EXODUS output file is also generated by BISON that contains more detailed distributions of data, but it cannot be processed by `bison_post` yet.

To run `bison_post`, all of the output CSV files must be in the same directory as the input file. This means that if the inputs were moved or divided into batches before running, the output files must be gathered into one location. As when running XML2MOOSE, bison_post can be executed using the following:

```
bison_post -c vera
```

Using data from the CSV files, the following 2D datasets will be added to the HDF5 file and will be registered in VERAView:

1) cumulative_damage_index
2) average_clad_temperature
3) bison_burnup
4) maximum_clad_hoop_stress
5) minimum_clad_hoop_stress
6) maximum_fuel_centerline_temperature
7) minimum_gap_distance
8) he_prod
9) maximum_clad_temperature
10) minimum_clad_temperature
11) plenum_pressure
12) average_fuel_temperature
13) rod_input_power
14) rod_output_power

An example of the maximum centerline temperature (K) distribution from WBN1 Cycle 6 is presented below:

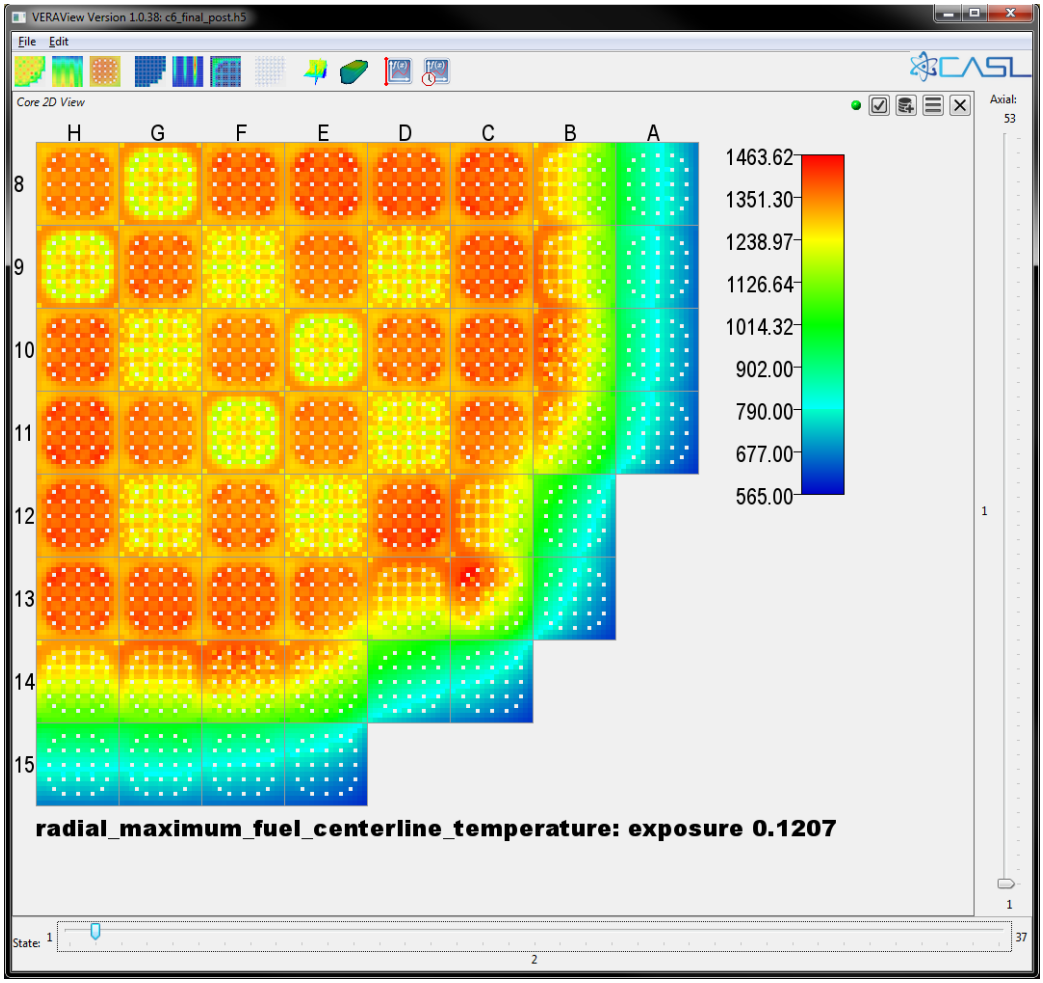


Fig. 4. VERAView visualization of maximum centerline fuel temperature in Cycle 6.

REFERENCES

1. K. T. Clarno et al., "High fidelity modeling of pellet-clad interaction using the CASL virtual environment for reactor applications," in: *Proceedings of the ANS Joint International Conference on Mathematics and Computation (M&C 2015), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method*. Nashville, TN, USA (2015).

2. J. Powers, S. Stimpson, K. Clarno, R. Pawlowski, *Integrating BISON into VERA-CS and Fuel Temperature Calculations*, Technical Report CASL-U-2016-1059-000, Oak Ridge National Laboratory (2016).

3. S. Stimpson et al., *Standalone BISON Fuel Performance Results for Watts Bar Unit 1, Cycles 1–3*, Technical Report CASL-U-2015-1010-001, Oak Ridge National Laboratory (2016).

4. J. D. Hales et al., *BISON Theory Manual: The Equations behind Nuclear Fuel Analysis*, Technical Report, Idaho National Laboratory (2015).

5. D. Gaston et al., "Moose: A parallel computational framework for coupled systems of nonlinear equations." *Nuclear Engineering Design*, 239 (2009): 1768–1778.

6. S. Palmtag and A. Godfrey, *VERA Common Input User Manual*, CASL-U-2014-0014-002, Revision 2, Oak Ridge National Laboratory. Available online at http://www.casl.gov/docs/CASL-U-2014-0014-002.pdf (February 23, 2015).

7. A. Godfrey and R. Lee, *VERAView User's Guide*, Technical Report CASL-U-2016-1058-000, Oak Ridge National Laboratory (2016).

8. A. Godfrey et al., *VERA Benchmarking Results for Watts Bar Nuclear Plant Unit 1 Cycles 1–12*, Technical Report CASL-U-2015-0206-000, Oak Ridge National Laboratory, Available online at http://www.casl.gov/docs/CASL-U-2015-0206-000.pdf (2015).

9. The HDF Group. *Hierarchical Data Format*, version 5, 1997–2016. http://www.hdfgroup.org/HDF5/.