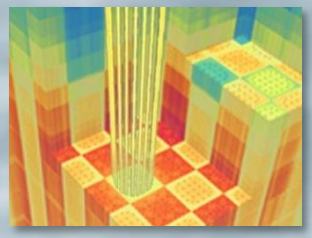
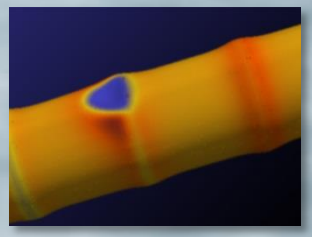
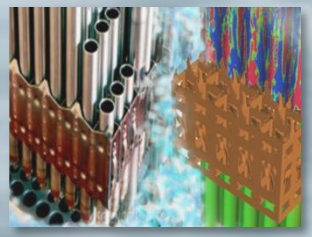
VERA Training: Input Description

Training Module 3

May 16-17, 2016







Training Objectives

- Give short overview of PWR core components
- Show how the reactor components are connected to the input
- We will not cover every input card, just the important concepts

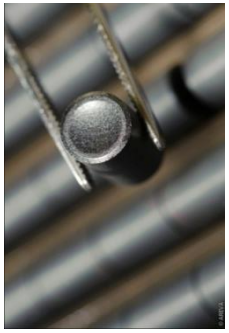
Hands on training will be given in later sections

Agenda

- Introductions
- Short PWR Overview
  - Go over quickly for non-utility users
- Detailed Review of Input
  - Geometry Concepts
  - Assemblies
  - Inserts
  - Control Rods
  - Statepoints
  - Code Options

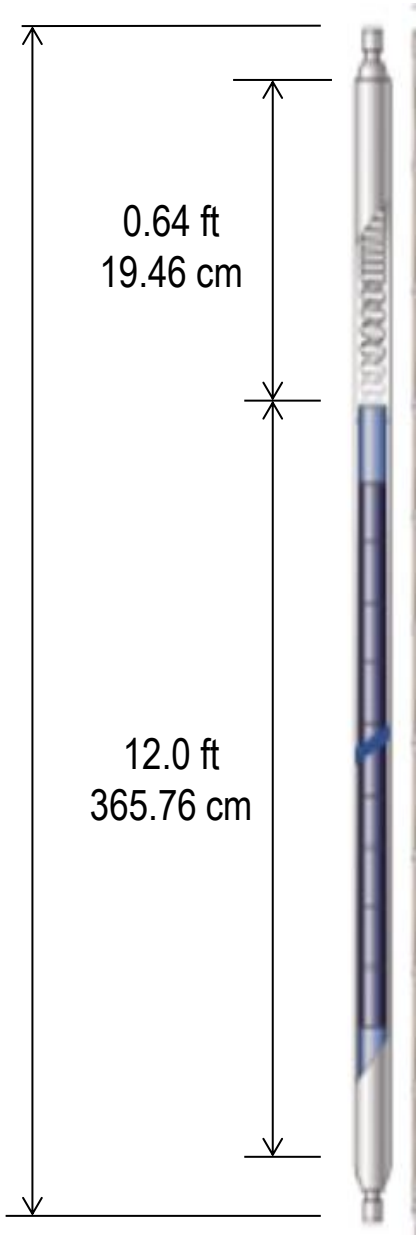
PWR Core Components

PWR Fuel Rods



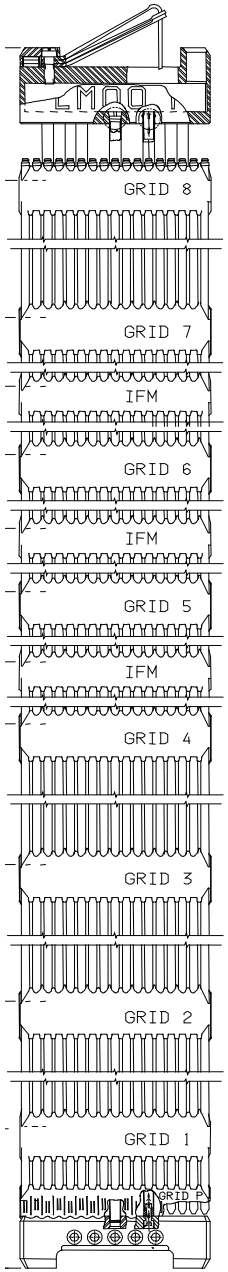
PWR Fuel Pellet
(0.82 cm OD)

- UO_2 Pellets
- multiple initial U-235 enrichments
  (2.1, 2.6, 3.1 w/o heavy metal)
- Density ~ 10.2 g/cc



All dimensions in presentation are typical of Watts Bar (non-proprietary)

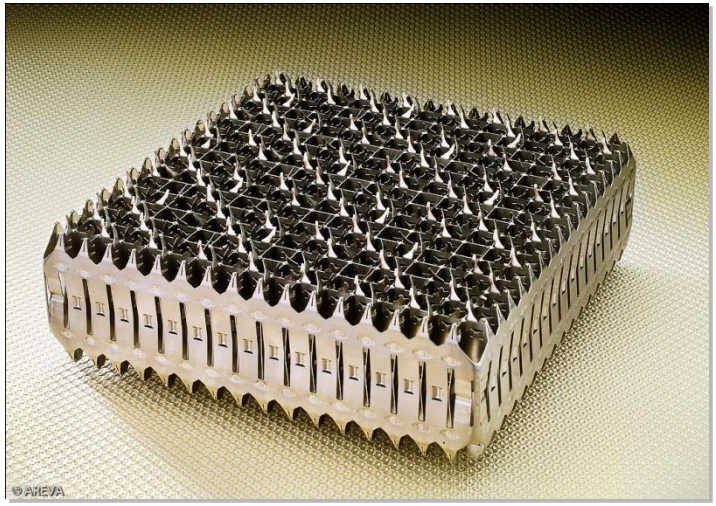
Fuel Assembly

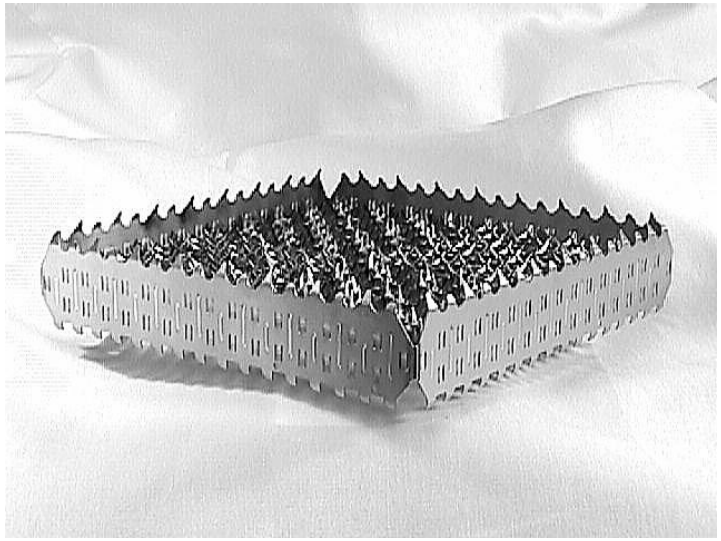


Fuel Assembly

Some idea of relative size!

Spacer Grids





8 (or more) grids per assembly

Grid Materials:

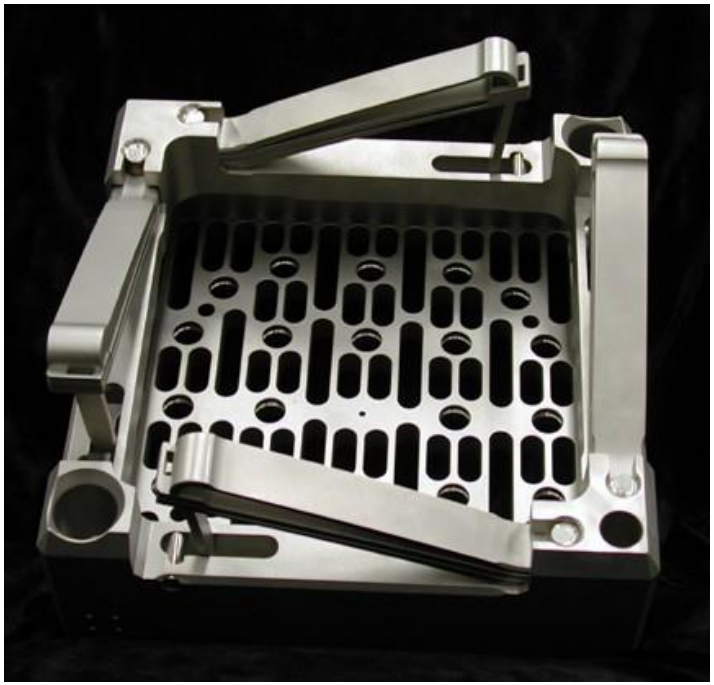
- Inconel (top and bottom)
- Zirconium (middle)
- Combination for structure + springs

Axial Height 3.8 cm
Mass 875-1014 g

Notes on modeling grids:

- Neutron transport usually models as a smeared volume
- Subchannel T/H usually models with loss coefficients and mixing factors
- CFD can model explicitly using CAD drawings (or porous material?)

Top and Bottom Nozzle



Materials: Stainless steel

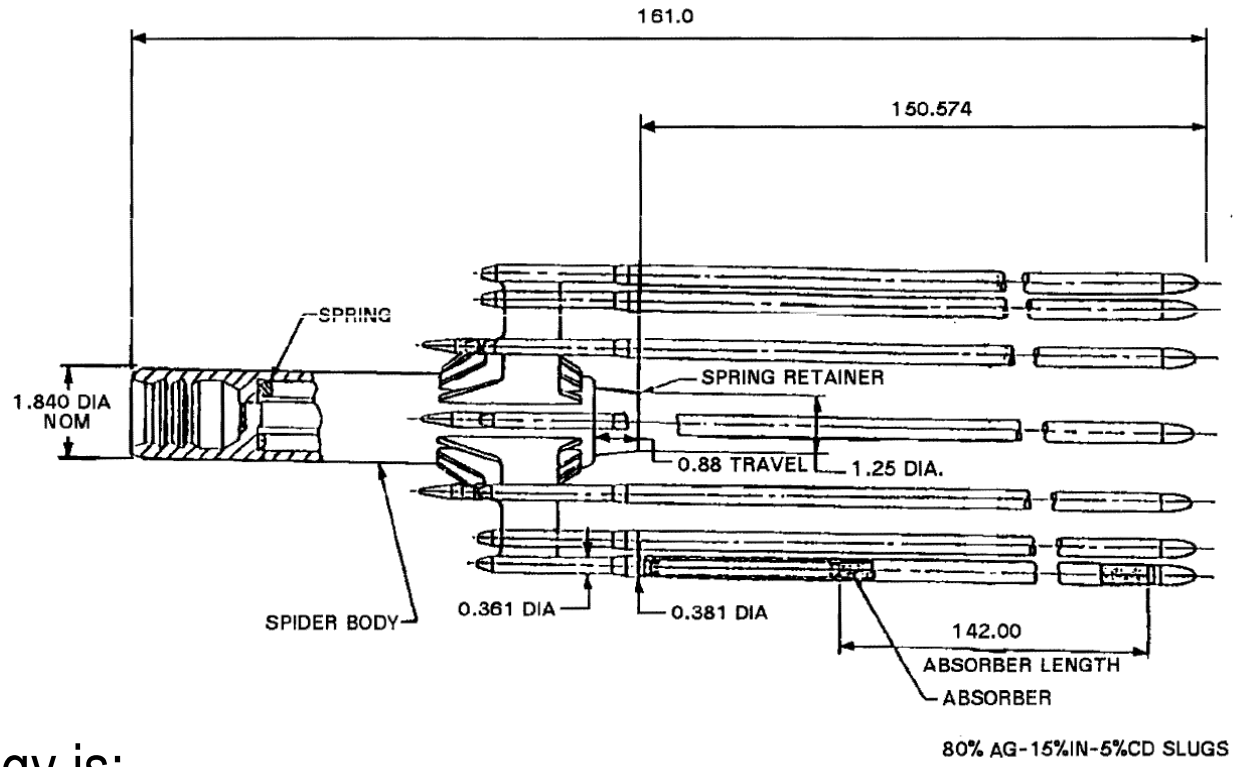
Axial Height 6-9 cm
Mass 6250 g

Modeled similar to spacer grids

Note on Reactivity Control

- As the core is operating, criticality must be maintained (fission source = absorptions + leakage)
- As the core depletes, the fission source term will decrease. Therefore, you must start out with “extra” source in the core at the beginning of life (BOL). This is “excess reactivity”.
- To counteract the excess reactivity at BOL, you must have extra absorptions at the BOL to match the excess reactivity and maintain criticality.
- As the source depletes, the extra absorptions must also deplete or be removed. This is done with one of the following:
  - Soluble boron in coolant
  - Control Rods (RCCA)
  - Integral Burnable Absorbers (IFBA, Gadolinia, Erbia)
  - Discrete Burnable Absorbers (Pyrex, WABA)

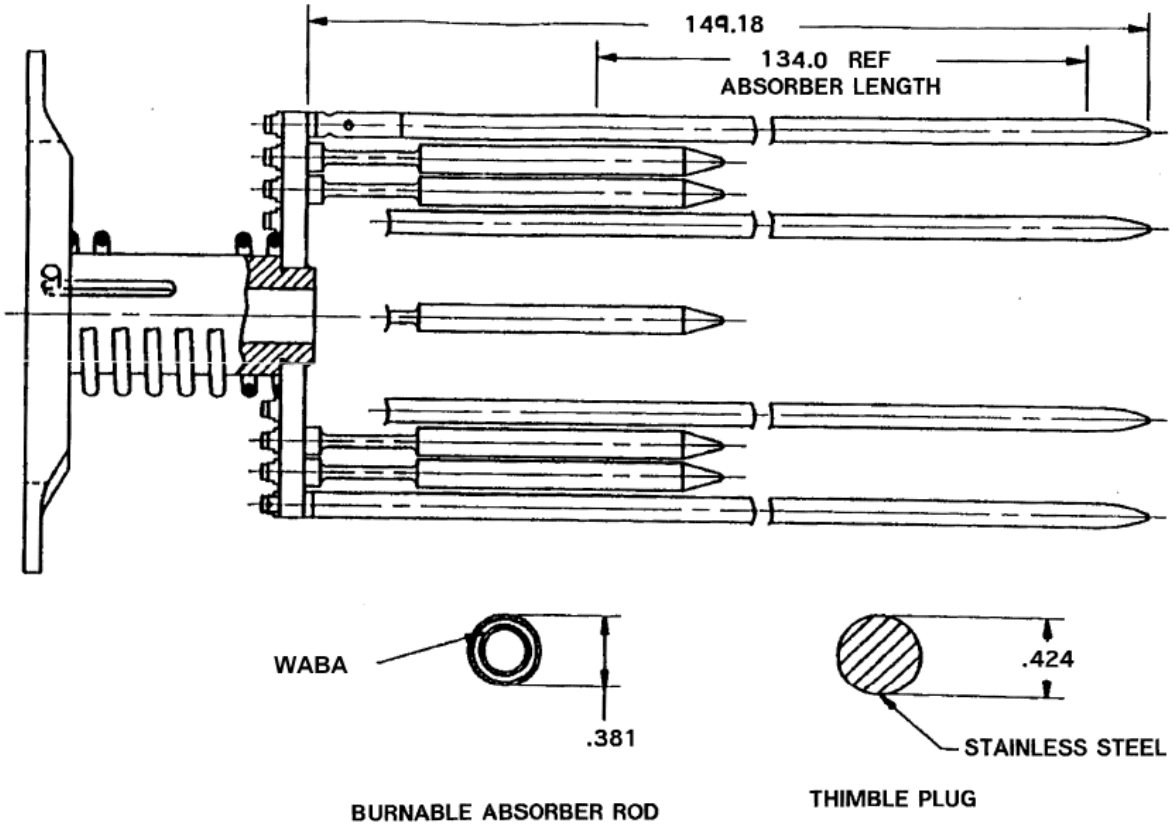
Control Rods



The correct terminology is:
Rod Cluster Control Assemblies (RCCA)

Control Rods move during operations

Discrete Burnable Absorber Assemblies

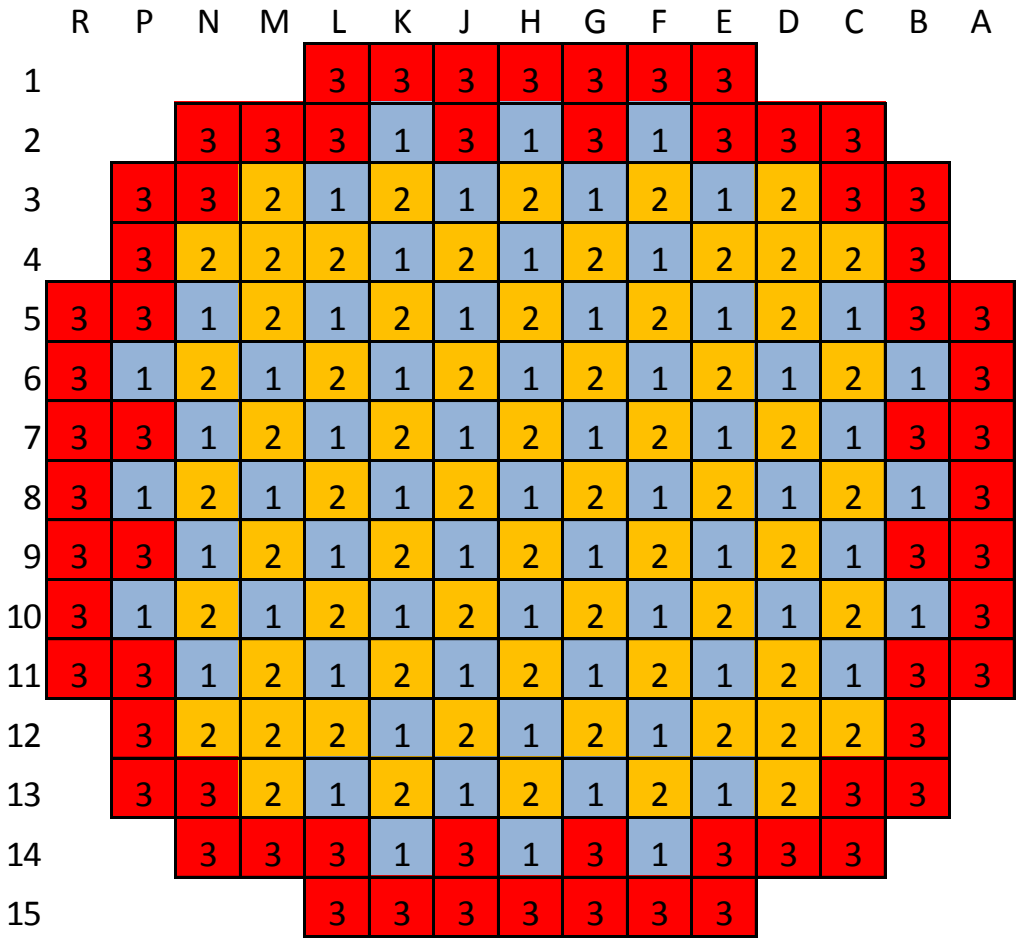


BA's can have different configurations of absorber rods (8, 12, 16, 20, 24)

BA's are usually Pyrex or WABA

BA's do not move during operation

Fuel Loading Pattern

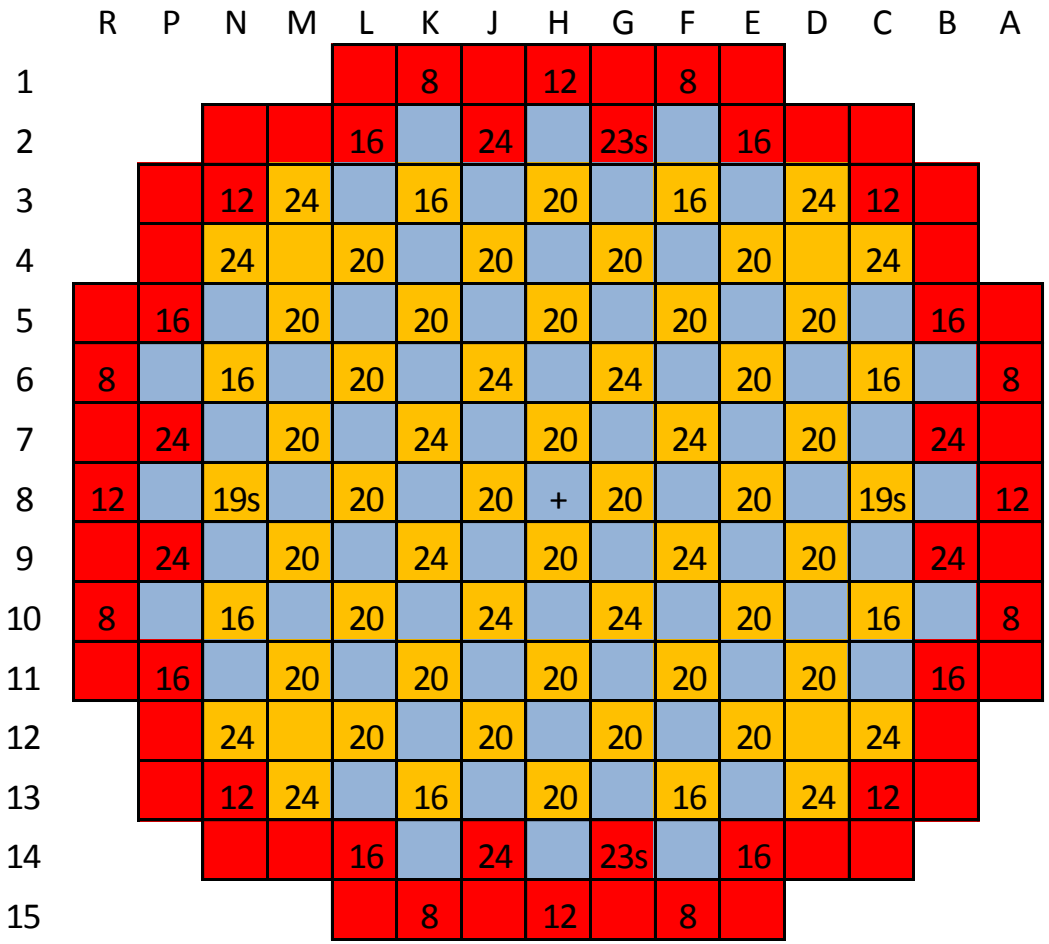


Reg 1	2.10%
Reg 2	2.60%
Reg 3	3.10%

Watts Bar Unit 1 Cycle 1

- 3 enrichment zones
- No IFBA
- WABA

Burnable Absorber Loading Pattern

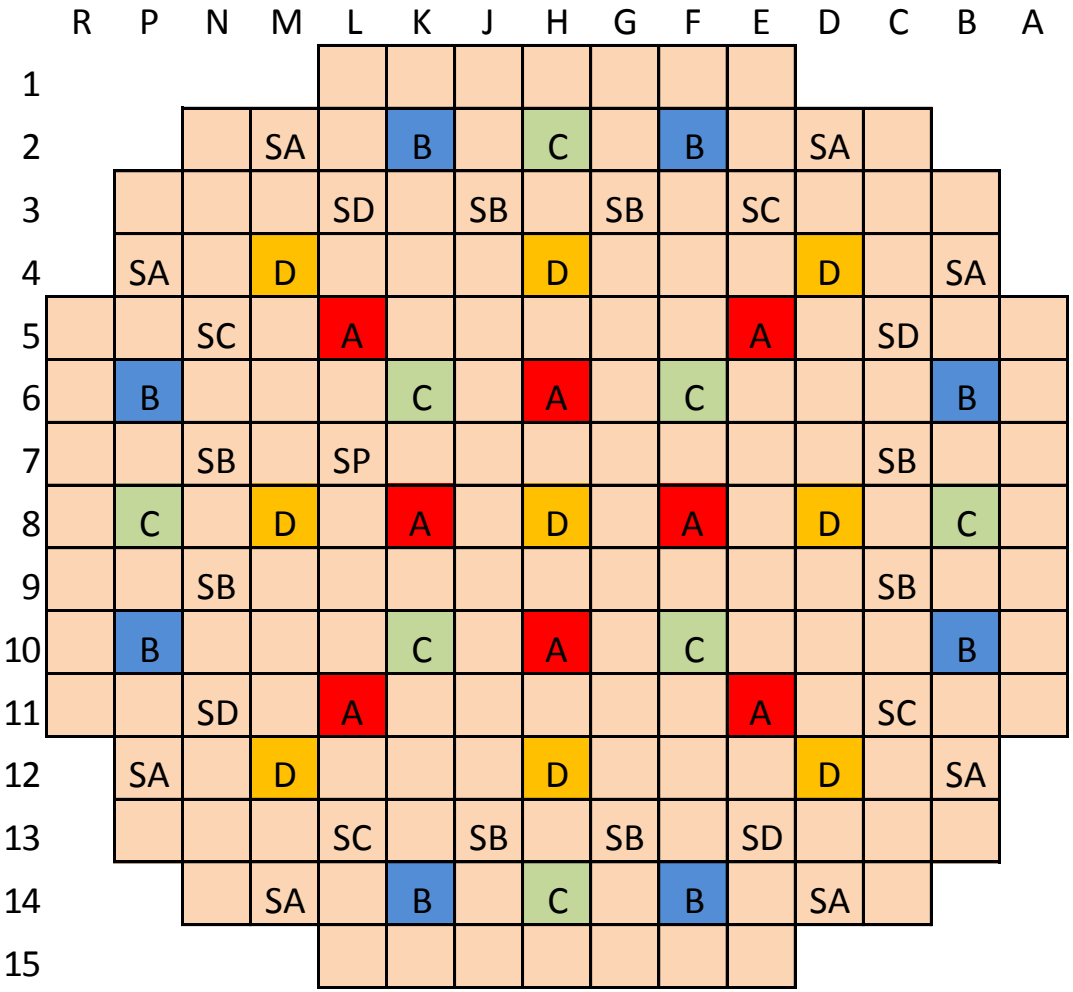


Watts Bar Unit 1 Cycle 1

- 5 Pyrex Assembly Types
- +4 neutron sources

Note: large number of BA's because all the fuel is fresh

RCCA Bank Positions

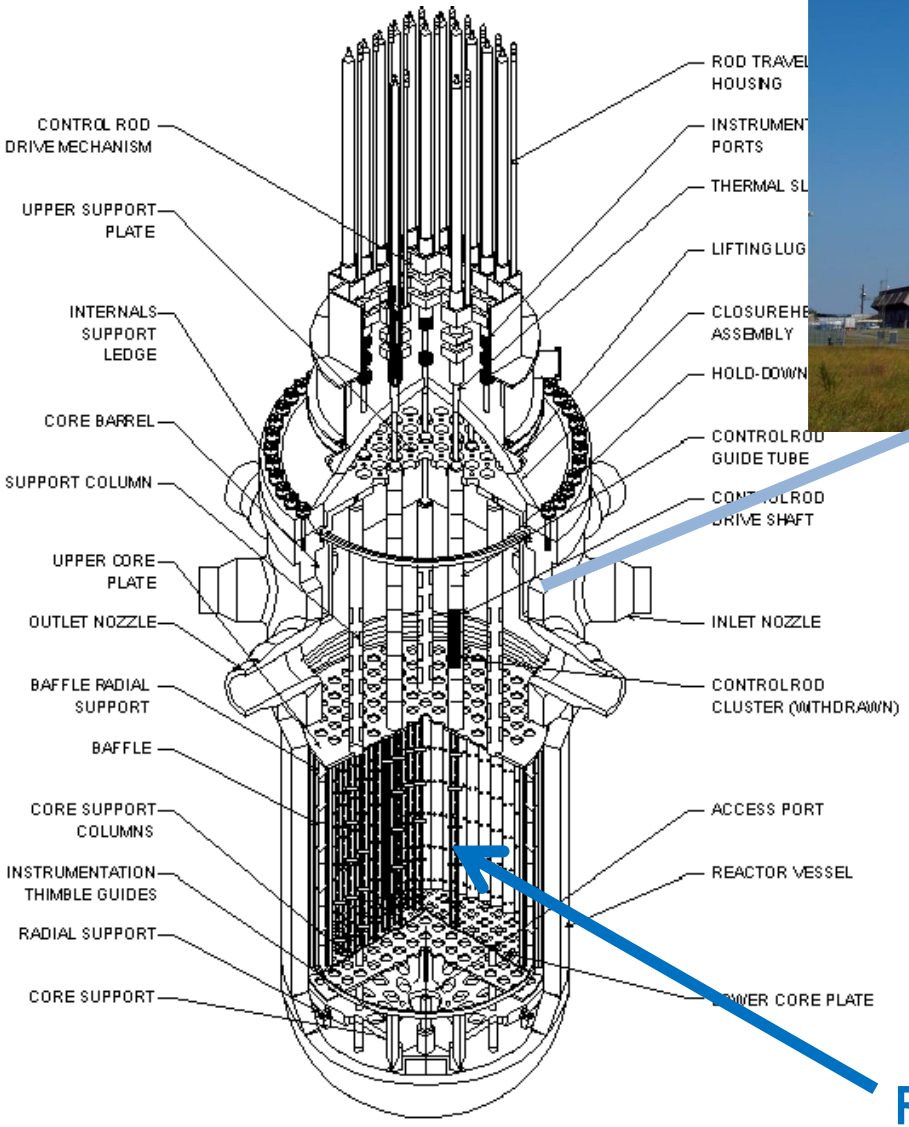


Watts Bar Unit 1 Cycle 1

- Operational Banks A-D
- Shutdown Banks
- Banks are symmetric

During normal operation, all rods are withdrawn and criticality maintained with soluble boron

PWR Vessel



TVA Watts Bar

Current CASL scope is the inside of the pressure vessel.
This will change as we move to transients

Reactor Core

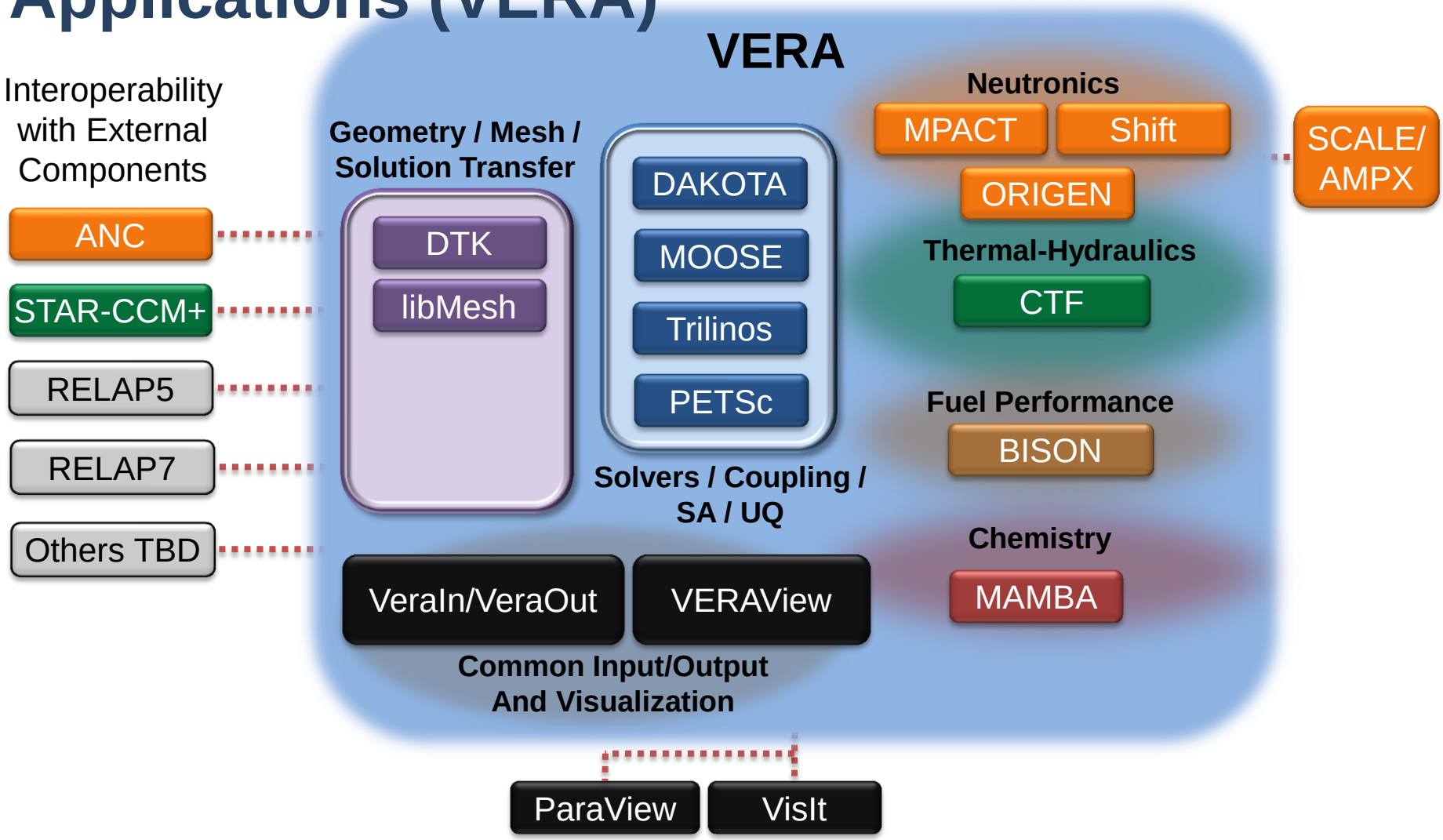
VERA-Input

VERA-Input

Why do we have a common input?

- VERA is a “virtual environment” that is composed of many different computer codes, each with its own input
- It was recognized that users should not have to become familiar with the input of every code
- Another benefit of a common input is to reduce errors due to inconsistencies between code inputs

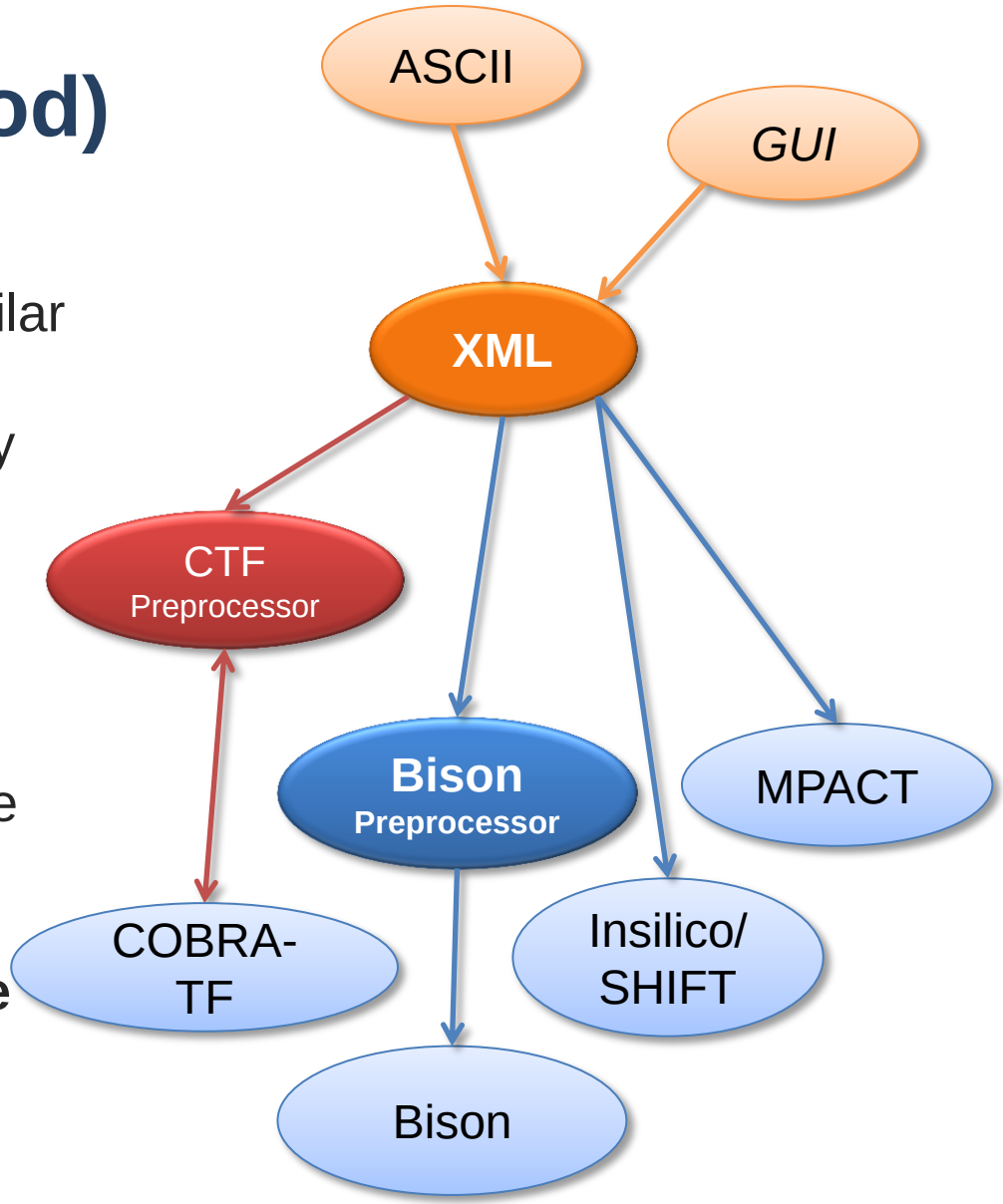
Virtual Environment for Reactor Applications (VERA)



Many different codes and inputs!

VERA-Input Internals (a peak under the hood)

- Input provides ability to create, archive, compare, and modify similar to current industry workflows
- Provide common reactor geometry for each physics components
  - assemblies, poisons, control rods, non-fuel structures, baffle, power, flow, depletion, etc.
- Reduce inconsistencies between coupled physics codes through the use of a common geometry description
- **Users only interact with a single ASCII input!**
  (or GUI in the future)



Why ASCII?

- Provides simple, intuitive interface to build complex models
- Input is free format, uses minimum characters, and allows symmetry options
- Input is easy to edit on remote computers and move files back and forth between computer systems
- Provides archivable format that can be used with version control and operating system saccess rules.

Students may want to open a copy of an input deck on their laptop to follow along (p9.inp)

Input Manual

- The current input manual (Rev 2) is include in the code distribution and can be downloaded from the CASL website.

  Palmtag, S., and A. Godfrey, “User Manual for the VERA Input Processor”, CASL Technical Report: CASL-U-2014-0014-002, February 28, 2015.

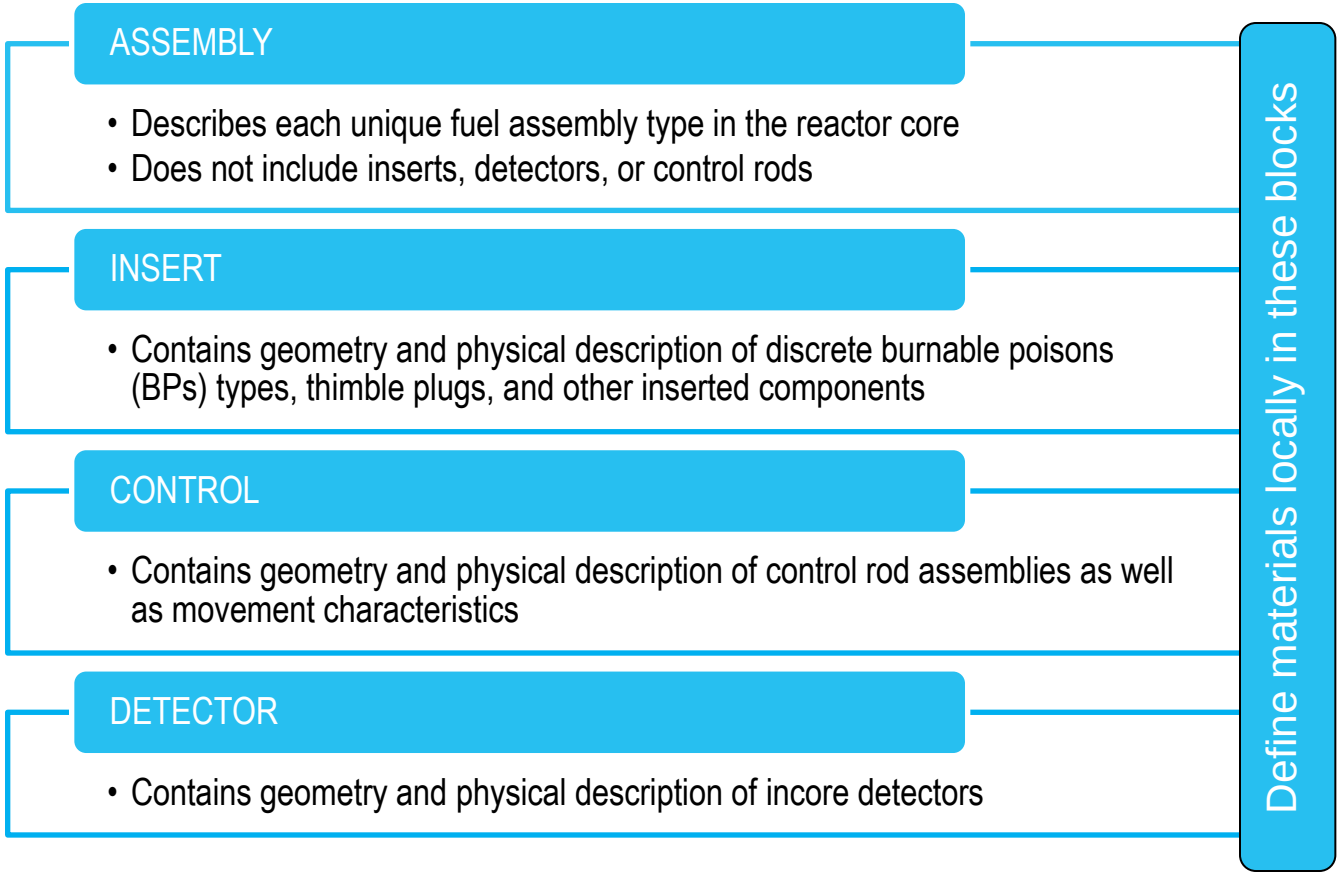
  http://www.casl.gov/docs/CASL-U-2014-0014-002.pdf

- Rev 3 will be released in the summer of 2016

  http://www.casl.gov/docs/CASL-U-2014-0014-003.pdf

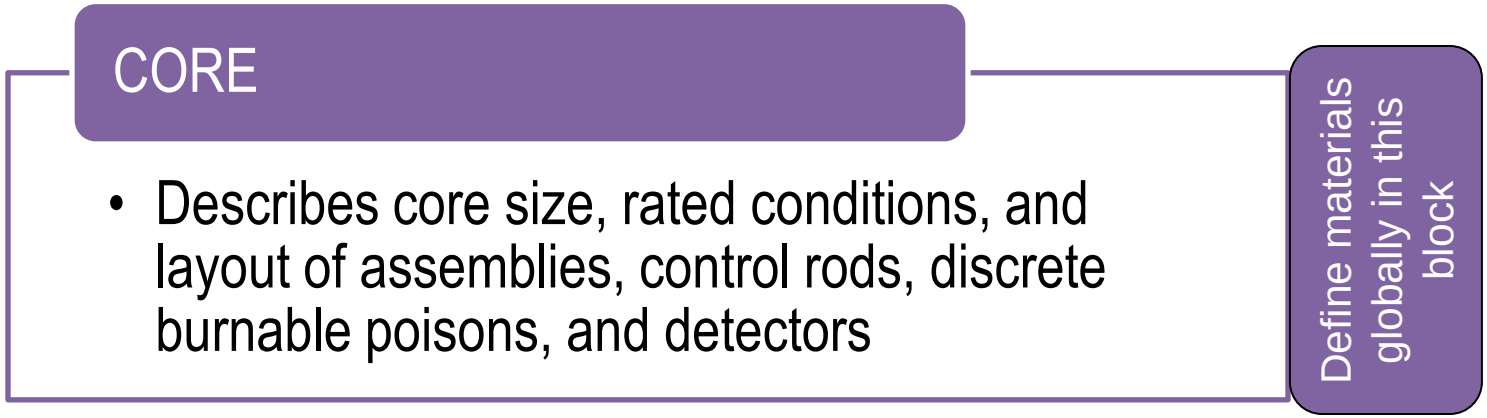
Input Blocks (Geometry Objects)

The VERAIn Standard Input Deck is divided into several [BLOCKS] which align with reactor geometry objects



Input Blocks (Core)

- The [CORE] block lays out the geometry objects into the core (assemblies, detectors, inserts, etc.)
- The [CORE] block does not change during a cycle depletion



Input Blocks (Statepoints)

- [STATE] blocks define the current core conditions at a single point in time (power, rod position, inlet temperature, etc.)
- Multiple [STATE] blocks exist, one for each statepoint

STATE

- Describes core operating parameters for each statepoint in a simulation

Input Blocks (Code Options)

Each physics code may have a block to set code-specific options

EDITS

- Specifies axial levels at which power and/or neutronics-T/H coupling will be solved

MPACT

- Neutron transport solver options and other code specific options

COBRATF

- Subchannel thermal-hydraulics code options

What does this look like?

```
[ASSEMBLY]
  title "Westinghouse 17x17"
  npin 17
  ppitch 1.260

  fuel U21 10.257 94.5 / 2.110
  fuel U26 10.257 94.5 / 2.619
  fuel U31 10.257 94.5 / 3.100

  cell 1      0.4096 0.418 0.475 / U21 he zirc
  cell 2      0.4096 0.418 0.475 / U26 he zirc
  cell 3      0.4096 0.418 0.475 / U31 he zirc
  cell 4             0.561 0.602 / mod   zirc
  cell 5             0.418 0.475 /     he zirc

  lattice LAT21
          4
          1 1
          1 1 1
          4 1 1 4
          1 1 1 1 1
          1 1 1 1 1 4
          4 1 1 4 1 1 1
          1 1 1 1 1 1 1 1
          1 1 1 1 1 1 1 1 1
```

```
[CORE]
  size 15
  rated 3411 131.68
  apitch 21.5
  height 406.337

  assm_map
      1
      2 1
      1 2 1
      2 1 2 1
      1 2 1 2 2
      2 1 2 1 2 3
      1 3 1 3 3 3
      3 3 3 3

  crd_bank
      D  -  A  -  D  -  C  -
      -  -  -  -  - SB  -  -
      A  -  C  -  -  -  B  -
      -  -  -  A  - SC  -  -
      D  -  -  -  D  - SA
      - SB  - SD  -  -  -
      C  -  B  - SA  -
      -  -  -  -
```

```
[CASEID]
  title 'WBN1 Cycle 1'

[STATE]
  power  0.0     ! %
  tinlet 557.33  ! F - 565K
  tfuel  565.0   ! K
  boron  1285    ! ppmB
  modden 0.743   ! g/cc
  feedback off
  sym qtr

  rodbank SA 230
          SB 230
          SC 230
          SD 230
           A 230
           B 230
           C 230
           D 167
```

Geometry Concepts

We build up the assembly geometry from smallest → largest

1. Define material
2. Define cell
3. Define 2D lattice/segment
4. Define 3D assembly
5. Add grids and nozzle
6. Place assemblies in core

1. Define Materials

Structural Material

mat [user-name] [density (g/cc)] {[libname$_i$] [fraction$_i$], i=1, N}

```
mat he     0.000176  he-4
mat inc    8.19
mat gmat    8.0  zirc4 0.5  ss 0.5
mat zirc   6.56  zirc4 1.0
mat aic   10.20
mat pyrex  2.23
mat b4c    6.56
```

Fuel Material

fuel [user-name] [density] [th-den] / [U-235 enrichment]

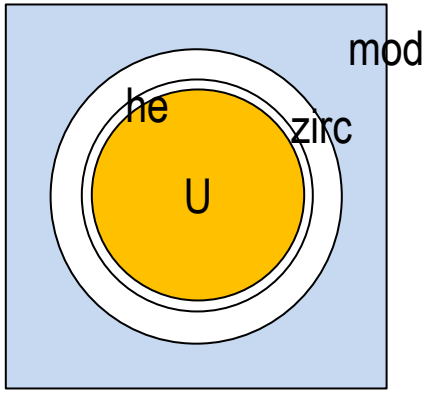
```
fuel  U21 10.257 95.0 / 2.11
fuel  U26 10.257 95.0 / 2.60
fuel  U31 10.257 95.0 / 3.10
```

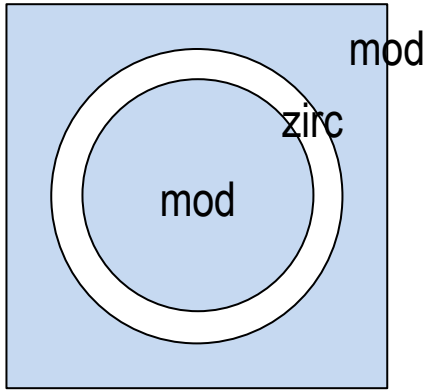
2. Define Cells

```
cell 1   0.4096 0.418 0.475 / U21 he  zirc    ! low enriched pin cell
cell 2   0.4096 0.418 0.475 / U26 he  zirc    ! med enriched pin cell
cell 3   0.4096 0.418 0.475 / U31 he  zirc    ! hi  enriched pin cell

cell X          0.561 0.602 / mod  zirc       ! guide tube
cell O          0.559 0.605 / mod  zirc       ! instrument tube

cell 8          0.418 0.475 / he   zirc       ! plenum pin
cell 9          0.418 0.475 / zirc zirc       ! end plug
```



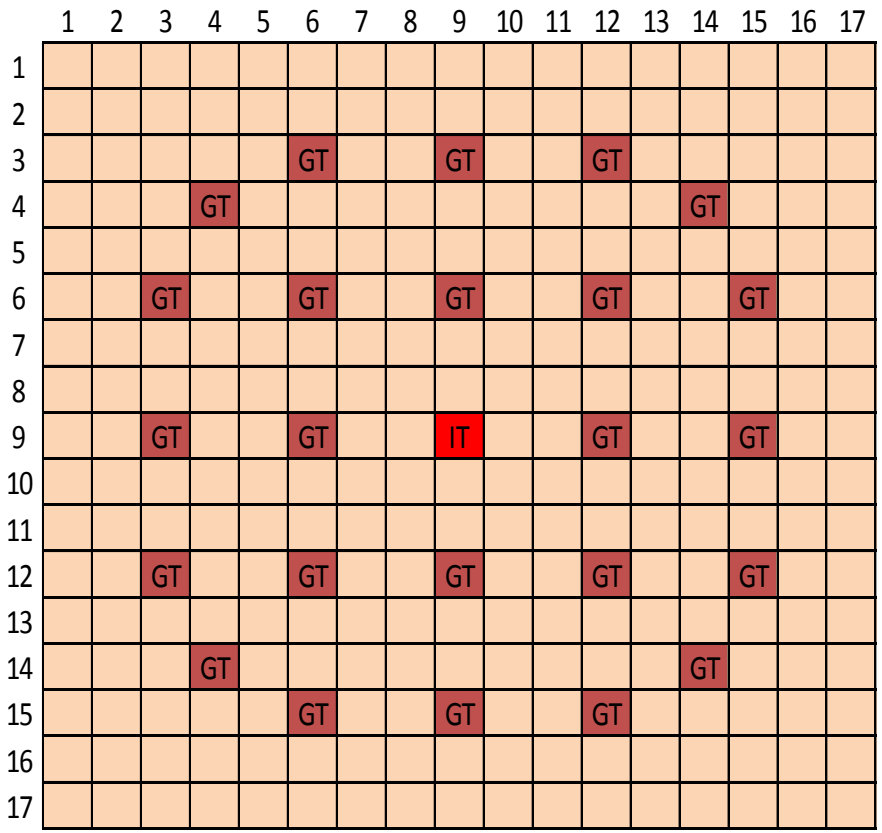


Outer material defaults to “mod”, which is determined by T/H

3. Define 2D Lattice/Segment

```
lattice FUEL1
     O
     1 1
     1 1 1
     X 1 1 X
     1 1 1 1 1
     1 1 1 1 1 X
     X 1 1 X 1 1 1
     1 1 1 1 1 1 1 1
     1 1 1 1 1 1 1 1 1

! Numbers/labels in map
! refer to cell labels
```



Only need to define octant (1/8) maps due to symmetry

Define multiple lattices for each unique 2D slice

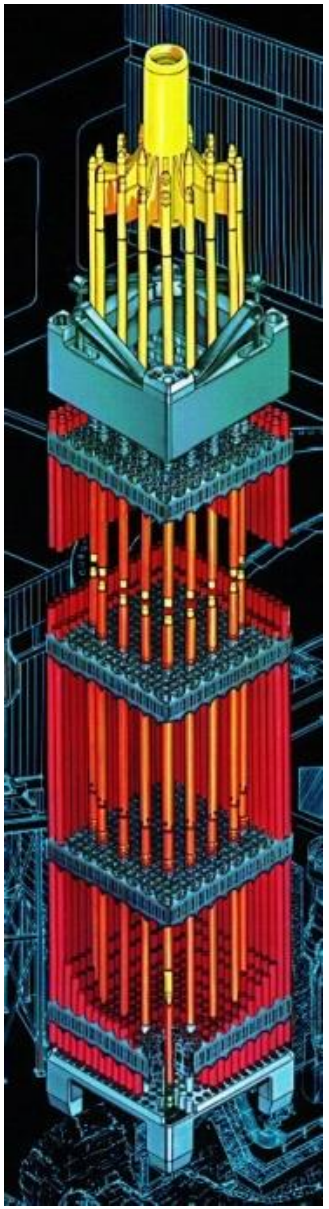
4. Define 3D Assembly

Stack up 2D lattices to form a 3D assembly

```
axial A1              ! Assembly label
               6.050  ! Bottom elevation
      LGAP1  10.281   ! Lattice label, elevation
      PCAP1  11.951   ! etc.
      FUEL1 377.711
      PLEN1 393.711
      PCAP1 395.381
      LGAP1 397.501

! Lattice labels are defined on lattice maps
! Assembly label used in core maps
```

Assembly name “A1” will be used when building the core

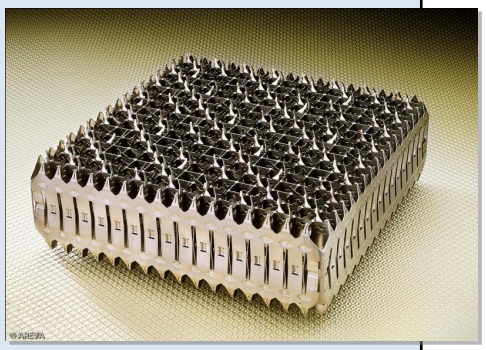


5. Add Grids and Nozzles

```
grid END inc  1017 3.866   ! Name, material, mass (g), height (cm)
grid MID zirc 875  3.810

grid_axial                 ! Axial grid locations
    END  13.884
    MID  75.2
    MID 127.4
    MID 179.6
    MID 231.8
    MID 284.0
    MID 336.2
    END 388.2

lower_nozzle  ss 6.05  6250.0  ! material, height, mass (g)
upper_nozzle  ss 8.827 6250.0  ! material, height, mass (g)
```



Note: The position of the grid mass and grid height will be switched in a future code release

6. Place Assemblies in Core

```
[CORE]
  assm_map
                    A3 A3 A3 A3 A3 A3 A3
              A3 A3 A3 A1 A3 A1 A3 A1 A3 A3 A3
           A3 A3 A2 A1 A2 A1 A2 A1 A2 A1 A2 A3 A3
           A3 A2 A2 A2 A1 A2 A1 A2 A1 A2 A2 A2 A3
        A3 A3 A1 A2 A1 A2 A1 A2 A1 A2 A1 A2 A1 A3 A3
        A3 A1 A2 A1 A2 A1 A2 A1 A2 A1 A2 A1 A2 A1 A3
        A3 A3 A1 A2 A1 A2 A1 A2 A1 A2 A1 A2 A1 A3 A3
        A3 A1 A2 A1 A2 A1 A2 A1 A2 A1 A2 A1 A2 A1 A3
        A3 A3 A1 A2 A1 A2 A1 A2 A1 A2 A1 A2 A1 A3 A3
        A3 A1 A2 A1 A2 A1 A2 A1 A2 A1 A2 A1 A2 A1 A3
        A3 A3 A1 A2 A1 A2 A1 A2 A1 A2 A1 A2 A1 A3 A3
           A3 A2 A2 A2 A1 A2 A1 A2 A1 A2 A2 A2 A3
           A3 A3 A2 A1 A2 A1 A2 A1 A2 A1 A2 A3 A3
              A3 A3 A3 A1 A3 A1 A3 A1 A3 A3 A3
                    A3 A3 A3 A3 A3 A3 A3
```

Additional Core maps for RCCA's, RCCA Banks, Inserts, and Detectors

[INSERTS] and [DETECTOR] Concepts

- INSERT and DETECTOR geometry is defined the same as assembly geometry
  - cells → 2D segments → 3D axial description → core map

```
[INSERT]
 title "Pyrex"
 npin 17
 cell X  0.214 0.231 0.241 0.427 0.437 0.484 / he ss he pyrex-vera he ss

rodmap  PY24
   -
   - -
   - - -
   X - - X
   - - - - -
   - - - - - X
   X - - X - - -
   - - - - - - - -
   - - - - - - - - -       ! Dashes represent "empty" locations
```

[INSERT] locations

- Once Inserts are defined, they are loaded into the core like the assemblies were loaded

```
[CORE]
insert_map
     -
    20 TP
     - 24  -
    20 TP 20  -
     - 20 TP 20  -
    20  - 16  - 24 12
     - 24  - 16  - TP
    12 TP  8 TP
```

- User can use octant maps instead of full-core maps
- Insert names are user-defined strings that have been defined in [INSERT] blocks
- Dashes represent “empty” locations
- User defined insert names:
  - TP (Thimble plug)
  - 16 (16 pyrex rod insert)
  - 20 (20 pyrex rod insert)
  - 24 (24 pyrex rod insert)
  - Etc.

[CONTROL] Concepts

- RCCS geometry is defined same as assembly geometry
  - cells → 2D segments → 3D axial description → core map
- RCCS geometry is defined as “fully inserted”
- Total “stroke” is defined as distance from “fully inserted” to “fully withdrawn”
- Total number of notches is number of steps from fully inserted to fullywithdrawn

Example: Stroke 360 cm, 228 total notches

- 228 notches is fully withdrawn
- 114 notches is inserted half-way
- 0 notches is fully inserted

[CONTROL] locations

- Once control rods are defined, we can load them into the core like assemblies and inserts
- Control rods need both a location and a bank name

```
[CORE]
 crd_map
   1
   - -
   1 - 1
   - - - 1
   1 - - - 1
   - 1 - 1 - -
   1 - 1 - 1 -
   - - - -
 crd_bank
   D  -  A  -  D  -  C  -
   -  -  -  -  - SB  -  -
   A  -  C  -  -  -  B  -
   -  -  -  A  - SC  -  -
   D  -  -  -  D  - SA
   - SB  - SD  -  -  -
   C  -  B  - SA  -
   -  -  -  -
```

- User can use octant maps and/or qtr-core maps
- Control names are user-defined strings that have been defined in [CONTROL] blocks (“1” in this case)
- Dashes represent “empty” locations
- User defined bank names

[STATE]

- STATE blocks define individual reactor “states” at a point in time
- Define variables that change during a depletion: power, inlet temperature, bank position, depletion step, etc.

```
[STATE]
  power 65.7
  tinlet 557.6 F
  rodbank D 192
  deplete EFPD 9.0

[STATE]
  power 99.7
  tinlet 558.1 F
  rodbank D 219
  deplete EFPD 32.0
```

- Can have as many [STATE] cards as necessary

[MPACT/COBRATF] Code Option Blocks

- Each individual code has a “code block” to define code specific options.

```
[MPACT]
  scattering         TCP0
  ray_spacing        0.1
  azimuthals_octant  12
  k_tolerance        1e-5
  flux_tolerance     1e-4

  num_space        4234
  num_angle        1
  num_threads      1
  par_method       EXPLICITFILE
  par_file         part_p5_73r_58z.txt
```

```
[COBRATF]
  beta_sp    0.005
  parallel         1
```

Initialization Option

- There is a new option in the input parser to define a set of default values for most of the code blocks
  - Default set of materials
  - Default options for the code blocks
- This option simplifies the input
- This option is activated by running the parser with the "--init" command line option (double dashes)
- It is highly recommended that you set up new models with the automatic initialization

Materials

- There are special rules when defining materials
- Materials can be defined in the CORE block
  - Materials have global “scope” throughout the input deck
- Materials can be defined in ASSEMBLY/INSERT/ DETECTOR/CONTROL blocks
  - Materials have local “scope”, only for that particular block
  - Useful for cases where you might have two assemblies from different vendors, and each one has a slightly different “zirc” composition
- The material definitions are usually not necessary if you use the automatic initialization option from the last slide

Materials

- VERA-CS includes a large number of default materials for LWRs, mostly based on SCALE 6.2 compositions
  - The defaults are available with the - -init option on the parser
  - air, aic, al2o3, b4c, boron, cs, gad, he, inc, pyrex, ss, water, zirc4. etc.
  - Defined in /VERAInExt/verain/scripts/Init/CORE.ini
- Customers can define their own proprietary materials and bring them in as include files
  - One workflow is to define and verify material definitions in a separate file with read-only access. Individual users can than "include" the verified material definitions in their input

Include Files

- VERAIn allows include files to be used.

  Examples:

```
include ctf_options.inc
include /home/projects/Reactor/cycle12/assembly2.inc
```

- One workflow option is to:
  - Create input decks for individual assembly inputs (or control, etc.)
  - Verify inputs
  - Make the files read-only files on the system
  - Allow the users to “include” the files in their inputs

Look at Example Input

Link to Full Core Input file

Link to Full Core XML file (optional)

Don’t worry about all of the details now. We will go through examples in the next training sections!

Running Parser

- VERA Input parser is in the VERA GIT directory under:

  …/verain/scripts/verain/scripts/react2xml.pl

  Or in the automatic path

- Run parser with command line:

```
react2xml.pl --init [file.inp] [file.xml]
```

- You can view the resulting xml file, but it is meant to be read by individual physics codes, not humans

Next

- The next training session will go through several examples in details
- Start with single-assembly input and move up to a full-core input

Questions?

www.casl.gov