Wavgen.py: part of COO's waveform definition language (WDL)

Peter Mao

Wavgen.py converts waveforms defined by (time, board, channel, value) tuples into states and calls.

```
modulefile minimal.mod
signalfile minimal.signals

sequence Start:
CALL Clock

waveform Clock:
0 4 0 1.25
0 4 1 1.25
0 4 0 1.25
0 4 1 1.25
2 4 0 -2.75
2 4 1 -2.75
2 4 0 -2.75
2 4 1 -2.75
3 RETURN Clock

waveform Clock1:
0 4 0 1.25
0 4 1 1.25
2 4 0 -2.75
2 4 1 -2.75
3 RETURN Clock1

parameter exptime=0
parameter Expose=0
parameter Readout=0
```

```
In [3]: w.loadWDL('minimal_TMP.wdl')
Using MOD file: /home/ztf/Software/wdl/minimal.mod
Loading signal mnemonics from /home/ztf/Software/wdl/minimal.signals
Specify base file name to generate output, or 'stdout' to print to screen.
Catalog of timing objects:
index  label                          type      exit   time [us]
---------------------------------------------------------------------
   0:  Start                          sequence   002    0.04
   1:  Clock                          waveform   002    0.03
   2:  Clock1                         waveform   002    0.03
```

```
In [4]: w.script()
[PARAMETER#]
exptime=0
Expose=0
Readout=0
[LINE#]
Start: # sequence
STATE000; CALL Clock      #     0      4
Clock: # waveform
STATE001; STATE000        #     1      2
STATE002; RETURN Clock    #     2      3
Clock1: # waveform
STATE001; STATE000        #     1      2
STATE002; RETURN Clock1 # 2      3
Out[4]: True
```

- Given the timing information, wavgen determines the minimum set of machine states and the ACF script with which to call the states.
- Sequences, which call waveforms or other sequences are also parsed.
- For reasons lost to time, ACF parameters are also carried along.
- The output of wavgen is ALMOST ACF-ready, with only the enumeration of tags missing.

```
In [6]: w.state()
[CONFIG]
STATE0\NAME=STATE000
STATE0\MOD4=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE0\MOD10=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE0\MOD11=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE0\MOD3="1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"
STATE0\MOD5="0,1"
STATE0\MOD6="0,1"
STATE0\MOD7="0,1"
STATE0\MOD8="0,1"
STATE0\CONTROL="0,3F"
STATE0\MOD9="0,1,0"
STATE1\NAME=STATE001
STATE1\MOD4="1.25,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE1\MOD10=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE1\MOD11=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE1\MOD3="1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"
STATE1\MOD5="0,1"
STATE1\MOD6="0,1"
STATE1\MOD7="0,1"
STATE1\MOD8="0,1"
STATE1\CONTROL="0,3F"
STATE1\MOD9="0,1,0"
STATE2\NAME=STATE002
STATE2\MOD4="-2.75,1,0,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE2\MOD10=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE2\MOD11=",1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1,,1,1"
STATE2\MOD3="1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"
STATE2\MOD5="0,1"
STATE2\MOD6="0,1"
STATE2\MOD7="0,1"
STATE2\MOD8="0,1"
STATE2\CONTROL="0,3F"
STATE2\MOD9="0,1,0"
STATES=3
```

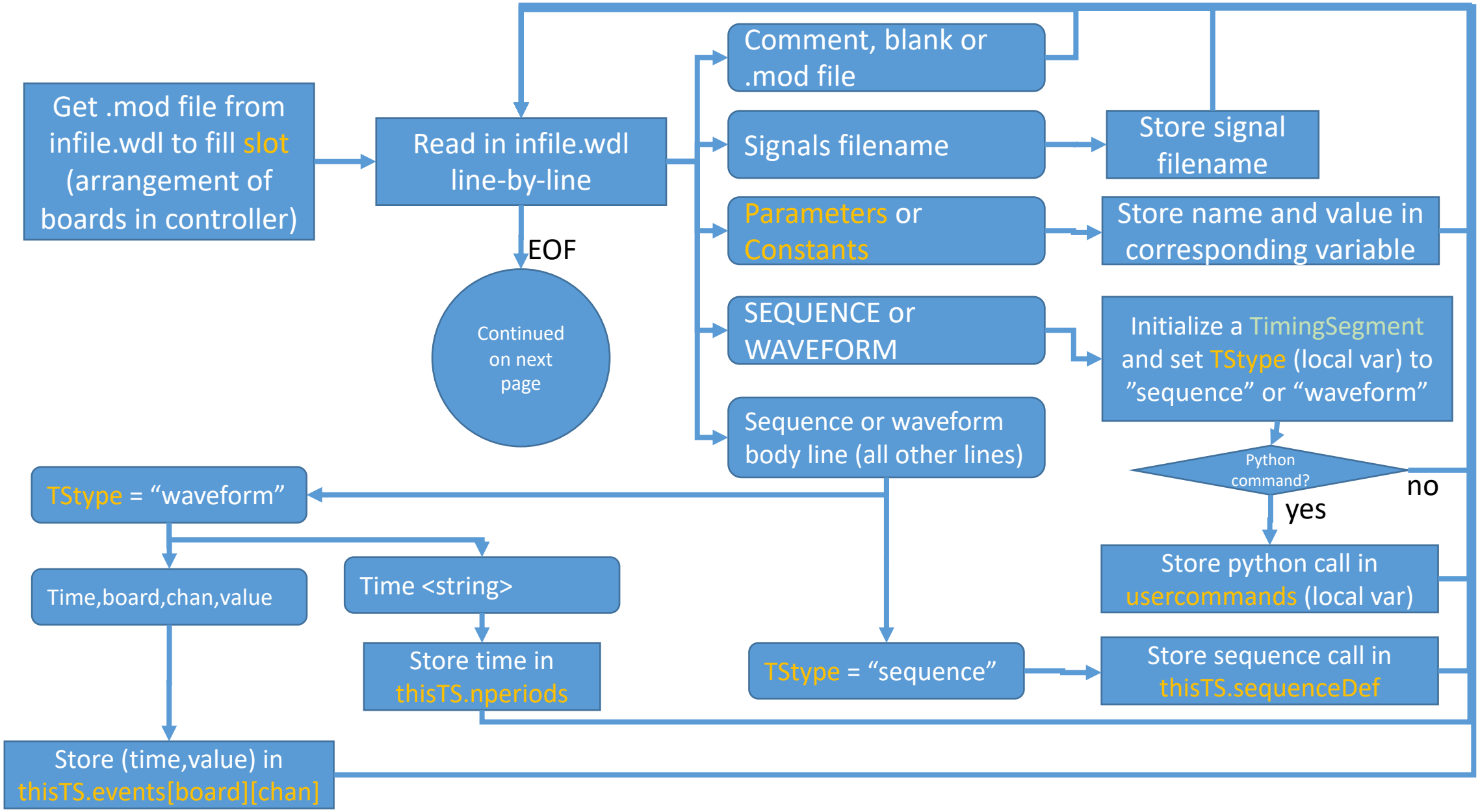
Global variables and “def” and “class” definitions in wavgen.py.

- A user only needs to know how to call loadWDL().
- The important global variables are UniqueStateArr and Catalog
- UniqueStateArr is an (# states)x(# channels) array, where each row is a unique state that the controller takes on.
- Catalog is a python list of TimingSegment objects
- After a TimingSegment is initialized (for example, as TS), and its events are defined, a call to TS.script() or TS.plot() generates an interal call to TS.__make_states(), which determines unique controller states and adds them to UniqueStateArr.
- After all TimingSegments are defined, the functions state() and script() are used to generate the proto-ACF texts.

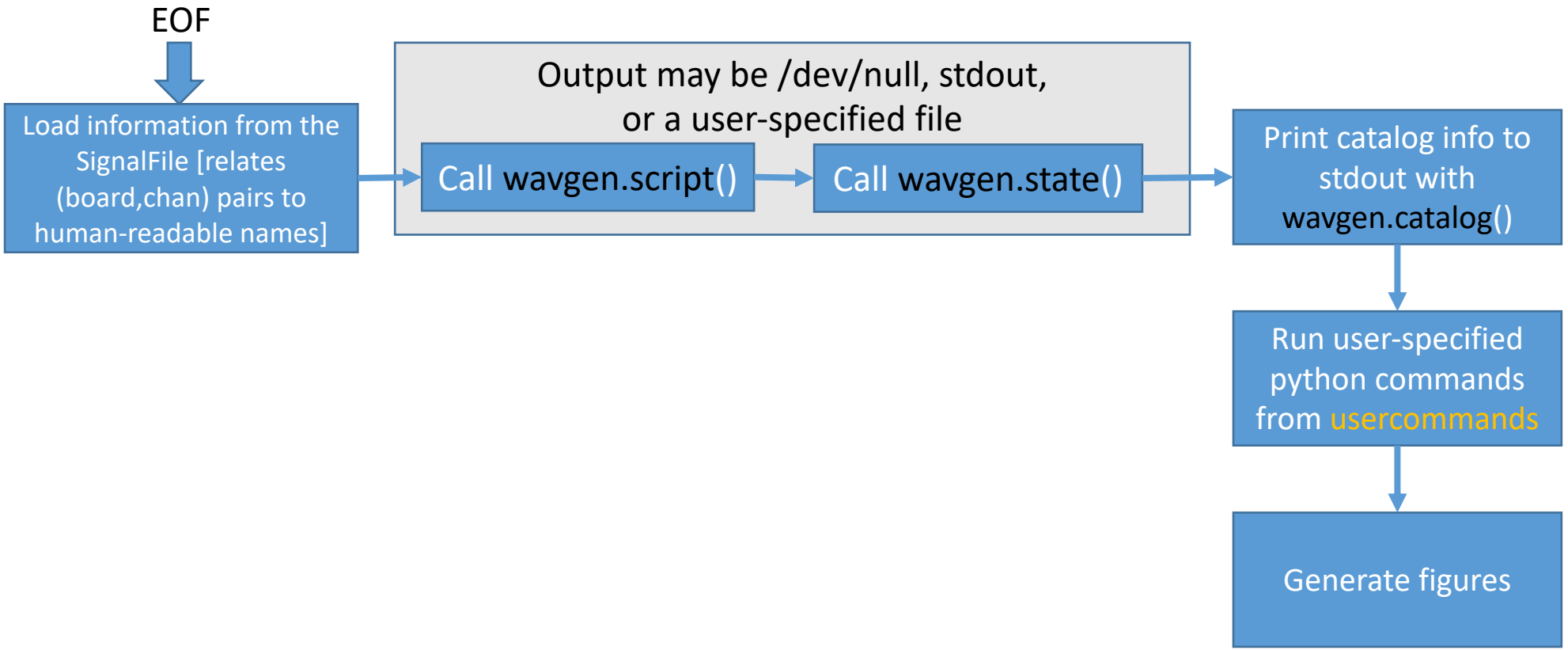
```
period_ns        = 10  # ns
slot = { 'drvr' : [],
         'lvds' : [],
         'adc'  : [],
         'back' : [0],
         'hvbd' : [] }
__boardTypes__     = ('drvr','lvds','adc','back','hvbd')
__chan_per_board__ = { 'drvr' : 2*8, # 2* to take care of level and slew flag
                       'lvds' : 20,
                       'adc'  : 1,
                       'back' : 6,
                       'hvbd' : 30 }
UniqueStateArr   = np.array([]);
Catalog          = [] # list of all TimingSegment objects
Parameters       = collections.OrderedDict() # all of the parameters
Constants        = collections.OrderedDict() # all of the constants
__SignalByName__ = {}
__SignalByIndx__ = {}
__seq_ID__       = 0
__TStypes__      = ('','waveform','sequence')
GenerateFigs     = False # set to True to plot waveforms when loadWDL is called.
__padmax__       = 25 # padding for comments in script
```

```
 34:def loadWDL(infile,outfile='/dev/null',verbose=1):
185:def __loadMod__(ModFile): #subroutine of loadWDL()
207:def __loadSignals__(__SignalFile__): #subroutine of loadWDL()
238:def __get_level_index_from_chan_slot__(slotnum, channel): # subroutine of __loadSignals__()
264:def __get_slot_chan_from_level_index__(levelColumnIndex): # only used in TimingSegment.plot()
283:def __index_of__(Name): # access Catalog elements by name instead of index
289:class TimingSegment(object):
292:    def __init__(self, name='', TStype = '', nperiods=1, endline=-2):
373:    def __tmax(self,reset=False): # subroutine of __make_states()
398:    def __fill_state(self, boardtype): # subroutine of __make_states()
443:    def __make_states(self):
521:    def script(self, outfile=sys.stdout):
658:    def __make_waveform(self, initialLevel=[]):
725:    def plot(self, cycles=2, initialLevel=[]):
814:def state(outfile=sys.stdout):
952:def script(outfile=sys.stdout, quiet=False):
999:def catalog(MagicNullArgument=None):
1012:def wplot(TimingObjectLabel):
1025:class modegen():
1028:    def __init__(self, modefile, acffile):
1065:    def __read_inputfile(self):
1103:    def __assign_defaults_from_acf(self):
1152:    def __index_modeKVpair(self):
1171:    def write(self,append=None):
```

loadWDL(infile.wdl)



loadWDL(infile.wdl)



UniqueStateArr

- 2D array of floats (# states) x (2 x # channels)
- For everything EXCEPT the DRIVER board, each DAC channel is represented by 2 columns of the USA.
  - Even columns represent DAC values
  - The +1 odd column represents the change flag
- For the DRIVER board, each DAC channel is represented by 4 columns of the USA:
  - (mod 4) = 0 columns represent the DAC value
  - The +1 and +3 odd columns represent the change flag
  - The +2 even column represents the FAST/SLOW flag
- STA Archon driver boards have 8 DACs – in wavgen, they are represented as having 16 channels, with the even channels being DACS and the +1 odd channels being the corresponding FAST/SLOW flags.
- Most of the work of filling the USA happens in TimingSegment.__make_states().
- State000, the top row of the USA is the “do nothing” state.

	driver DAC				driver DAC				DAC	
	value	change flag	“fast” flag	change flag	value	change flag	“fast” flag	change flag	value	change flag
000	0	0	0	0	0	0	0	0	0	0
001	1	1	0	1	0	0	0	0	2.3	1
002	0	1	1	1	0	0	0	0	0	1
003										
004										
005										
006										
etc										

class TimingSegment

- Variables:
  - Defined in self.__init__():
    - **endline**: integer indicating what to do at the end of this TimingSegment
    - **label**: integer index location of this TimingSegment in Catalog. defined to be the current value of the global variable __seq_ID__
    - **name**: This is unique in the Catalog. wavgen will overwrite Catalog entries without complaining.
    - **tstype**: '', 'sequence' or 'waveform'
    - **nperiods**: # clock cycles assuming every call is one clock cycle.
    - **sequenceDef**: sequence events (t, call). Filled by loadWDL.
    - **events**: waveform events (t,board,chan,value). Filled by loadWDL.
    - **Consts**: empty placeholder.
  - Defined in self.__make_states():
    - **do_anything_tt**: integer time when any event happens, including the start of the TimingSegment.
    - **do_anything_dt**: diff of **do_anything_tt** with a 0 at the end (zero time between the end of the TimingSegment and the end of the TimingSegment.)
    - **unique_state_ID**: 1x**nperiods** sparse array specifying the state of the system at every time point.
    - **sequence_times**: (for sequences) 1x#calls array indicating times at which call are made.
  - Defined in self.script():
    - **time**: # clock cycles to complete this TimingSegment.
    - **Params**: Dict of params and values. Used to track decrementing counters for time calculations and plotting
    - **ExitState**: Initialized to state000 in __init__(), set to the last state called in waveform or sequence.
    - **ExitLevel**: Initialized to state000's levels in __init__(), set to the levels defined after two passes through the TimingSegment.

class TimingSegment

- Methods:
  - __init__(self, name='', TStype = '', nperiods=1, endline=-2):
  - __tmax(self,reset=False): # subroutine of __make_states()
  - __fill_state(self, boardtype): # subroutine of __make_states()
  - __make_states(self):
  - script(self, outfile=sys.stdout):
    - generates the script
    - (if necessary) calls __make_states() to populate the UniqueStateArr
  - __make_waveform(self, initialLevel=[]):
  - plot(self, cycles=2, initialLevel=[]):
    - generates waveform plots using __make_waveform()
    - (if necessary) calls __make_states() to populate the UniqueStateArr