Internship report

Memory design with Mnemosyne (./internship)

1. Installation
   Guide: https://github.com/chrpilat/mnemosyne/blob/master/README.md
   Clone project: `git clone https://github.com/chrpilat/mnemosyne`

```
mkdir obj && cd obj
cmake -DCMAKE_INSTALL_PREFIX=/opt/mnemosyne ..
make && make install
cd /opt/mnemosyne/examples/batch_init
make -f Makefile.mnemosyne
```

   →finish Mnemosyne installation

2. Generate a BRAM component with Mnemosyne
   Go to the installed directory on your computer and make:

```
cd /opt/mnemosyne/examples/batch_init/
make -f Makefile.mnemosyne
```

   The output Verilog file will be generated in directory output_virtex7_verilog

3. Encode rules and initiate Mnemosyne memory block
   In this trial, we aim to make a scalable AmoebaSAT through utilizing multi-port memory block for bounceback rules storing and checking.
   Go back to the amoebasat/internship/ directory.
   Encode the bounceback rules:

```
g++ preprocess_bram.cpp -o preprocess
./preprocess "filename.cnf"
```

   After running, a new file "bram.init" will be generated in the same directory.
   Copy this file into the installed directory of Mnemosyne.
   Configure the details of memory design in file "config.cfg":
   - name: name of the output Verilog file
   - width: width of the memory block (by default, 32 bit). This width must match the width of one line in file "bram.init"
   - height: length of the memory block (by default, 1024 lines). Number of lines in "bram.init" corresponds to this height.
   - interface: list all the ports of this memory block, first port is by default a write port. If you want to make 1 write and 3 reads, put as "w, r, r, r"

   Note:

   - width and height of "bram.init" are editable in file "preprocess_bram.cpp" (line 9,10).
   - In this trial, only INTER rules are encoded. The COLLAPSE rules can be omitted based on a new bounceback scheme

Original

Bounceback Signal $L_{i,b} \in \{0,1\}$:

$$L_{i,b}(t+1) = \begin{cases} 1 & \text{(if } (B \ni (P, Q) \text{ such that } Q \ni (i,b)) \text{ and } \forall\ (j,c) \in P\ (X_{j,c}(t+1) = 1)), \\ 0 & \text{(otherwise)}, \end{cases}$$

where **Bounceback Set B = *INTRA* ∪ *INTER* ∪ *CONTRA*.**

CONTRA-omitted

Bounceback Signal $L_{i,b} \in \{0,1\}$:

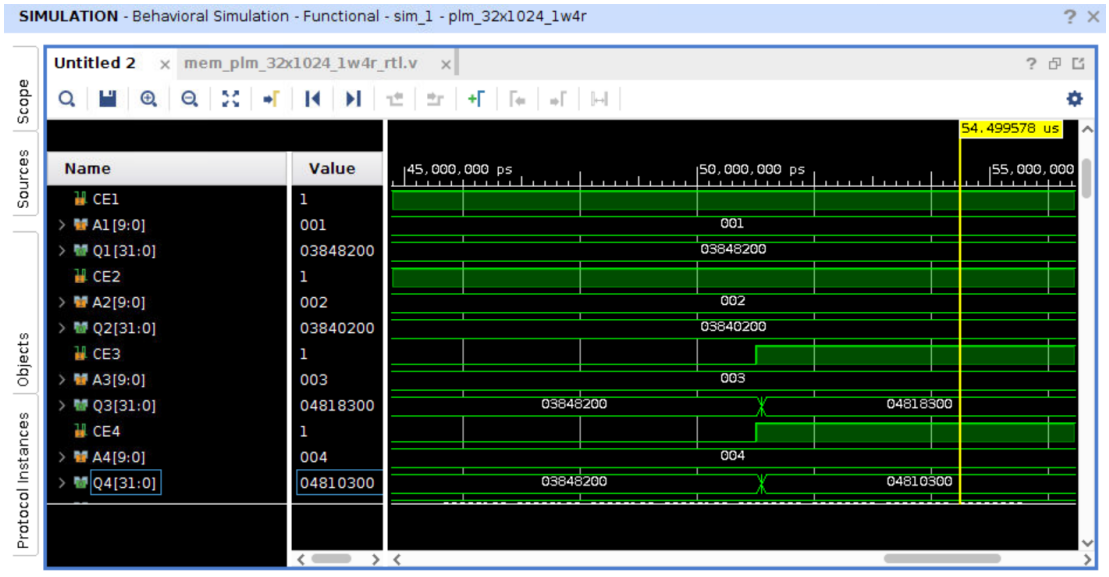
$$L_{i,b}(t') = \begin{cases} 1 & \text{(if } (B \ni (P, Q) \text{ such that } Q \ni (i,b)) \text{ and } \forall\ (j,c) \in P\ (X_{j,c}(t+1) = 1)), \\ 0 & \text{(otherwise)}. \end{cases}$$

If $L_{i,0}(t')=L_{i,1}(t')=1$, then $(L_{i,0}(t+1)=L_{i,1}(t+1))= (0,1)$ with a probability 0.5, otherwise **(1,0)**,
else $L_{i,0}(t+1)=L_{i,0}(t')$, $L_{i,1}(t+1)=L_{i,1}(t')$,
where **Bounceback Set B = *INTRA* ∪ *INTER*** ~~∪ *CONTRA*.~~

If you want to encode all rules (INTER + CONTRA), uncomment the code block between line 310-330 and comment out line 333-343 in file "preprocess_bram.cpp"

4. Generate the multiport memory block with Vivado & export it into IP
   Create a project in Vivado.
   Include the generated Verilog file (.v)
   Include a corresponding wrapper file (.v), location: /opt/mnemosyne/share/tech/virtex7/
   Run Vivado synthesis.

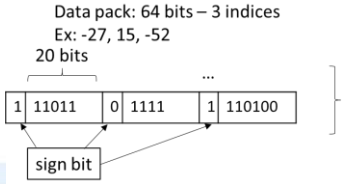
   Check if the data initialization is correct through opening wave viewer (Vivado)



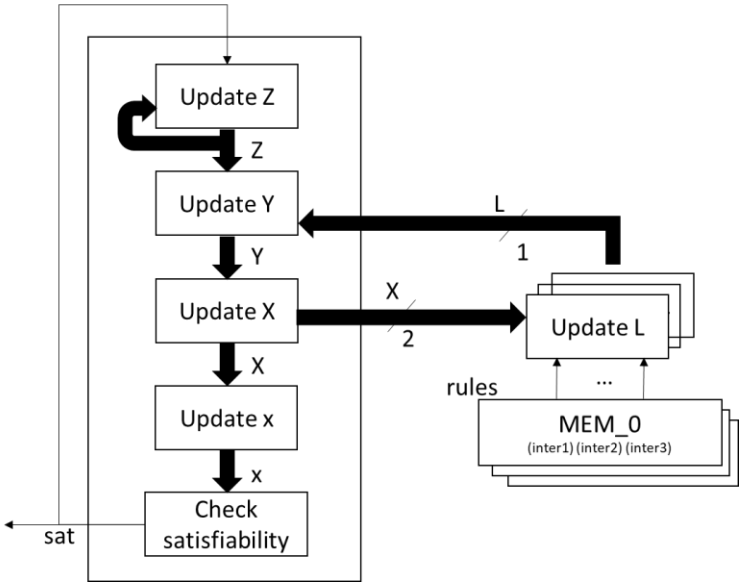
5. Synthesize the AmoebaSAT solver with generated multiport memory (on-going)

   Modify the INTER rules checking function in AmoebaSAT solver (added rules decoding)

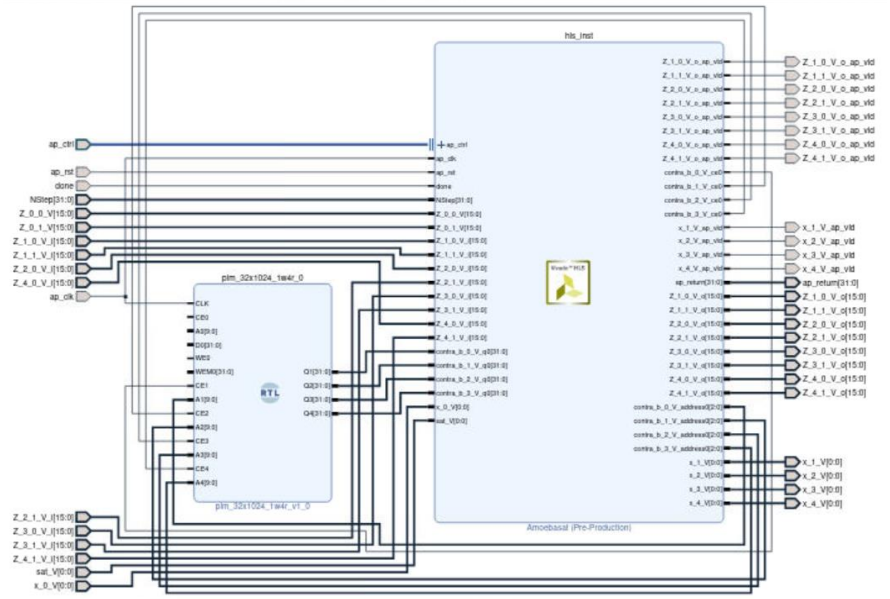
```
Loop_inter:
    for(i=0;i<size_bc;i++){
//#pragma HLS UNROLL factor=10

#pragma HLS PIPELINE
        //L[inter2][inter_sign2] = L[inter2][inter_sign2] | conflict;
        //L[inter3][inter_sign3] = L[inter3][inter_sign3] | conflict;
        //extract index
        f_t inter_row = inter_b[i];
        inter_sign1 = inter_row[0];
        inter1 = inter_row.range(20,1);
        inter_sign2 = inter_row[21];
        inter2 = inter_row.range(41,22);
        inter_sign3 = inter_row[42];
        inter3 = inter_row.range(62,43);

        conflict = (LargeX[inter2][inter_sign2]>0) &
                (LargeX[inter3][inter_sign3]>0);

        L[inter1][inter_sign1] = L[inter1][inter_sign1] | conflict;

        sat_temp = sat_temp & (!conflict);
    }

    return sat_temp;
}
```



Block diagram of AmoebaSAT with multiport memory:



In Vivado, create a block design including: Package AmoebaSAT IP (generated through Vivado HLS), multiport memory block (generated through Mnemosyne & synthesized with Vivado in part 4).



After including IPs into the block design, please manually connect the corresponding ports between AmoebaSAT IP and Mnemosyne blocks.

Run synthesis & implementation.

6. Preliminary result

   Synthesis on test.cnf between Original (centralized_hls) and this version

	Original	This
Slice	339	269
LUTRAM	0	13
LUT	831	790
FF	802	442
Clk (ns)	3.464	4.724
ADP	1174.3	1270.7

Others:

Performance of CONTRA-omitted version compared with others:

Instance	N_vars	Original (intra, inter, contra)	Ours-C (intra, inter, contra, collapse)	AmoebaSAT Slim	Intra + inter (CONTRA-omitted)
Uf50-01	50	2286 (0.82x)	1,558 (0.56x)		2,786 (1x)
Uf50-0100 (unique)	50	14268 (1.71x)	12,699 (1.52x)		8,352 (1x)
Uf225-028	225	618,973 (6.28x)	16,572 (0.09x)	303,775 (1.65x)	184,227 (1x)
Uf250-01	250	83,591 (1.10x)	11,120 (0.15)	12,625 (0.17x)	75,826 (1x)
Uf250-0100	250	4,813,976 (24.47x)	142,482 (0.72x)		196,696 (1x)
Flat30-1	90	3134 (1.02x)	2119 (0.69x)		3084 (1x)
Flat50-1	150	11911 (1.26x)	4168 (0.44x)		9454 (1x)
Flat75-10	225	16177 (1.19x)	6787 (0.50x)		13629 (1x)