Extracting a simplified view of design functionality based on vector simulation

Presenter: Hillel Miller, Freescale

Onur Guzey, Charles Wen, Li-C. Wang, University of CA – Santa Barbara

Magdy S. Abadir, Freescale

Presentation Outline

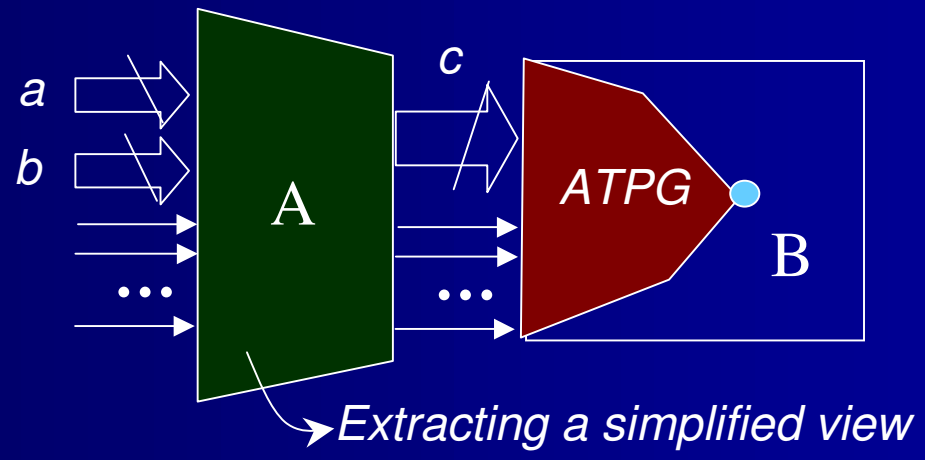
- Introduction
- Boolean learning
- Extension to Hybrid learning
- Word level learning
- An application on OpenRISC 1200
- Future directions

Introduction

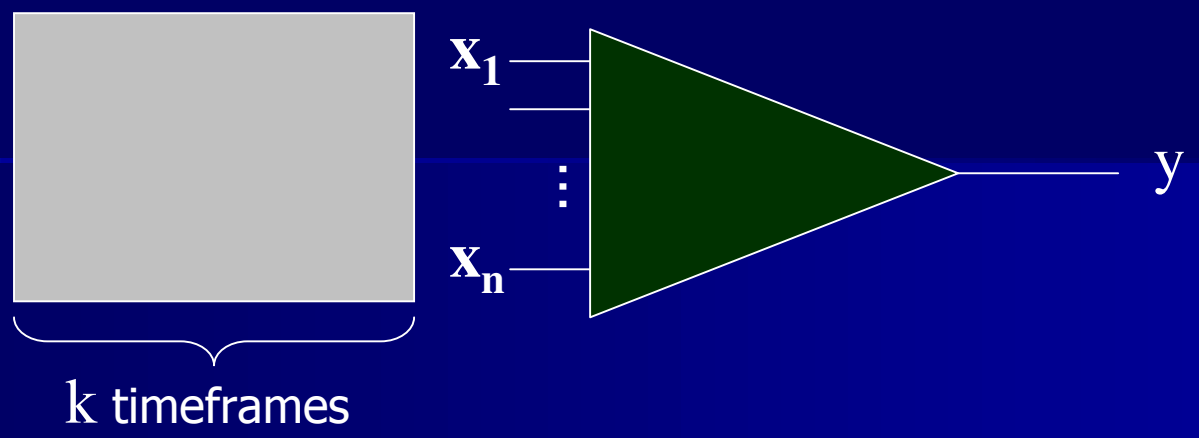
- Objective: Using simulation data to extract a "simplified view" of design functionality from a set of inputs to a set of outputs
- Once a simplified view is extracted, it provides a "short-cut" to control the outputs from the inputs
- One possible application is Test Pattern Justification (TPJ)

Enhanced controllability

- **A** can be complex and/or be the software implementing the I/O mapping, i.e. hard to search on
- Our interest is the target in module **B**
- We find a simplified view for module **A**
- ATPG works on module B to obtain a test at the inputs of the module. Then, this test is *mapped back* to the inputs of **A** by working on the *simplified view*



A "simplified view"



- The implementation of the function $y = f(x_1 \ldots x_n)$ can be quite complex (C++ program, etc.)
- A "simplified view" is a function *g* such that *g* is learned from simulation data
  - *g* is determined by the learning algorithm
- The input dimension of the learning problem is k*n
  - But we are hoping that there are only a few decision variables
  - Hence, learning needs to identify these variables

Simplified view

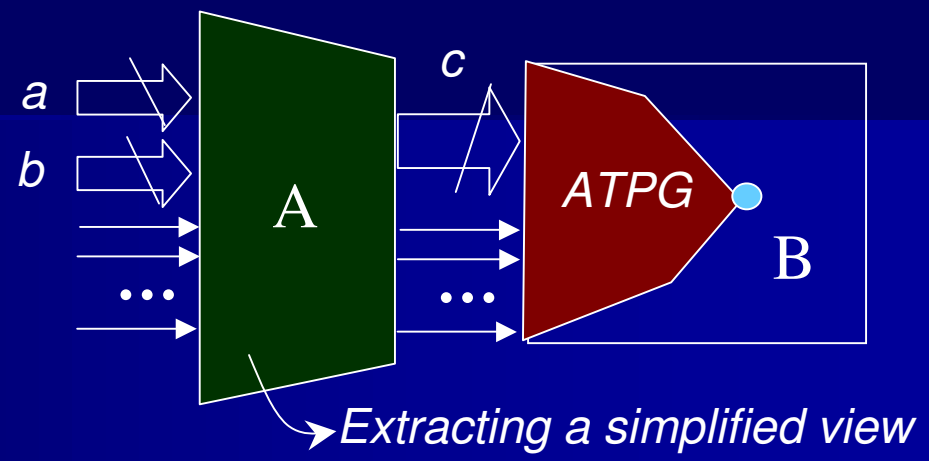
- If part of the functionality is too complex to be learned, it should not be included in the simplified view
  - Our goal is NOT trying to learn the design functionality of a module
- Our goal is merely to find a short-cut to provide better controllability

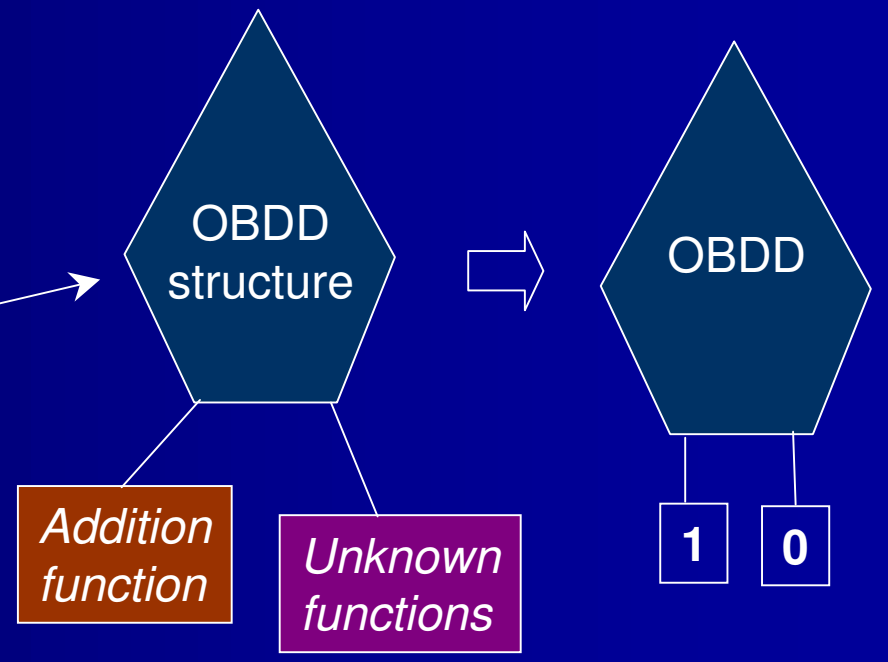
Simple view extraction, a learning problem

- Can a learner request data for different input values as it sees needed?
  - Yes: Query-based learning
  - No: Non-query-based learning
- We know that query-based is more powerful than non-query-based
- This work focuses on query-based
  - So that if certain functionality cannot be learned by query-based method, we know that it cannot be learned by non-query based method either

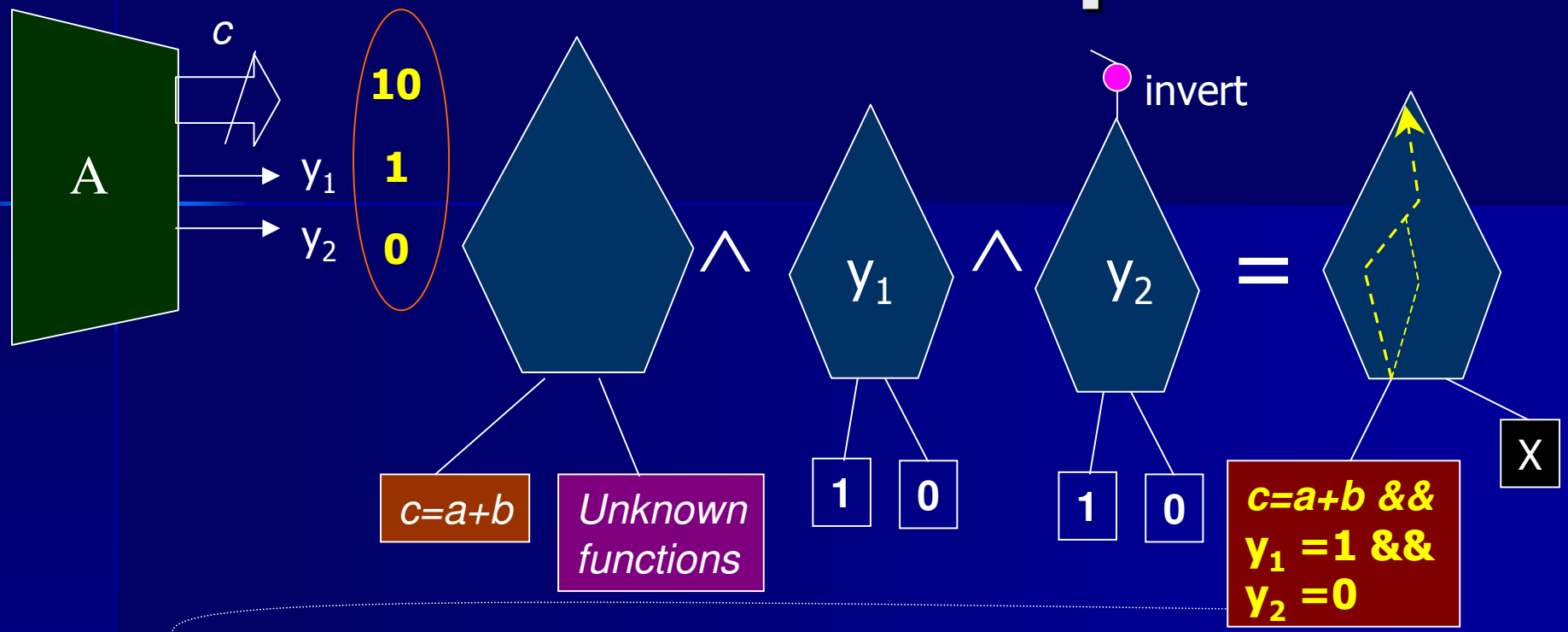
Simplified view as a decision diagram

- Assume that we want to extract a simplified view based on a given word-level function
  - **"c = a + b"**
- Module A may perform many other functions but we are interested only in the + function
- Figure on the right presents a way of modeling this problem as a Boolean learning problem.





Combine multiple OBDDs



Then, select one or more: {a=2, b=8}, {a=5, b=5}, …

- There is no guarantee that the resulting OBDD will contain a solution vector
  - We may drop an output variable and produce an approximate solution
- The justified vector(s) can be simulated to verify its validity

The Core of the Problem is Boolean learning

- <u>Boolean Learning:</u>

  Learn $\mathbf{y = g\,(x_1, \ldots, x_{n*k})}$

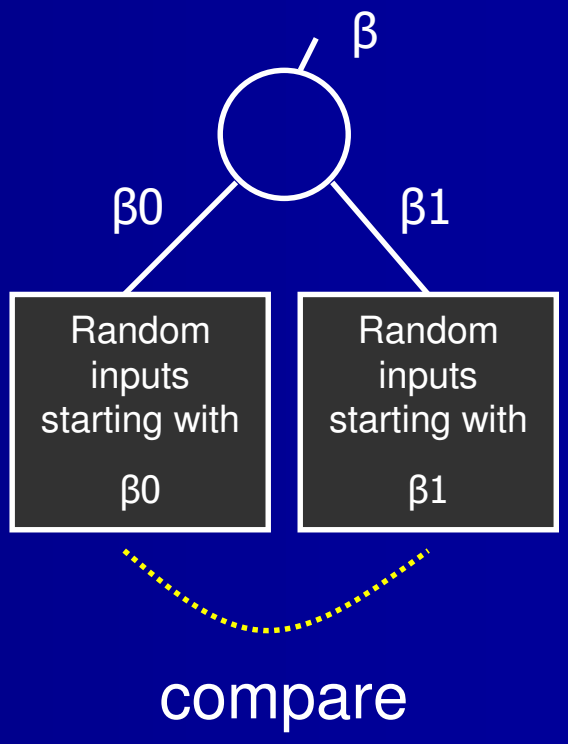
  where y and $x_1, \ldots, x_{n*k}$ are Boolean variables
- Boolean learning serves as the underlying engine for hybrid learning (which will be explained later on)
- Therefore successful development of an efficient Boolean learner is crucial.

OBDD-based Boolean learning

- Properties of the proposed Boolean learning algorithm:
  - Results are OBDDs: Therefore they can be used for test pattern justification
  - Query-based
  - Consistent: The algorithm guarantees that the resulting OBDD will not be bigger than the actual OBDD of the true function
  - Can work with limited resources: To prevent memory blow-up and long run-times, limit of OBDD size can be enforced (heuristic to ensure good learning result)

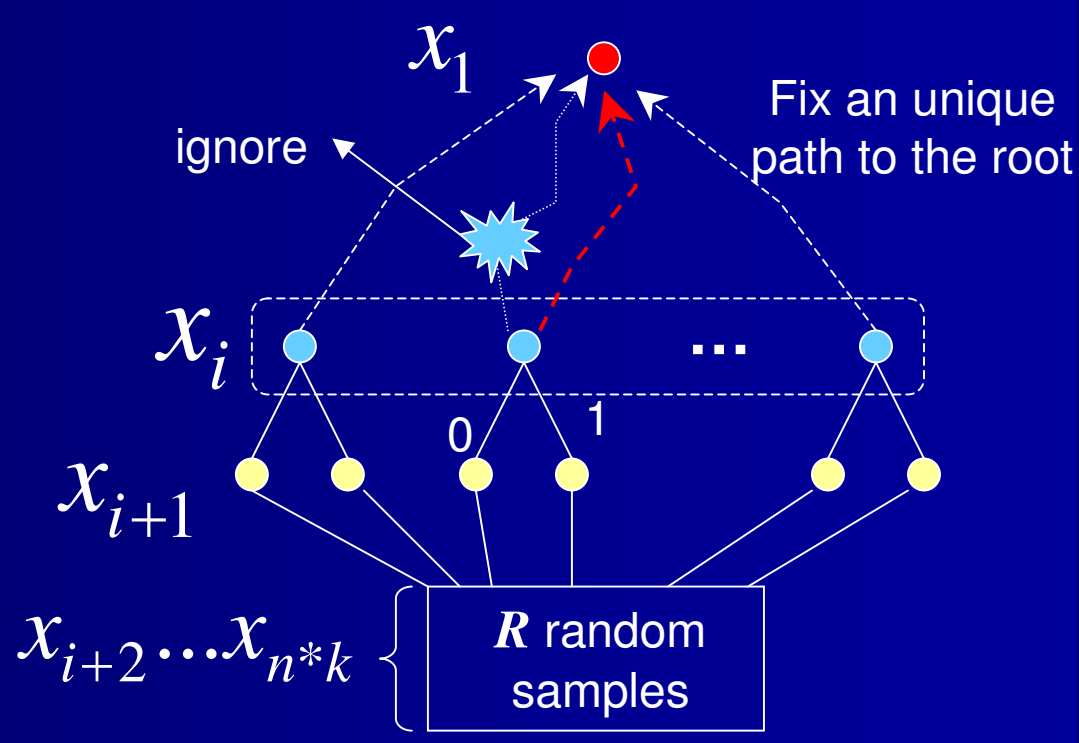
Basic idea

- Try to build the OBDD from the root.
- Decision to merge children of a node is made by comparing simulation results.
- If all the results are the same for both nodes, they are merged otherwise they are split.



Unique path selection

- When merging two (or more) nodes, one of the nodes is selected as the unique path to the root.
- This selection makes the learning algorithm *consistent* because it does not split two nodes that are not split in the actual OBDD.



Comparison to Fourier learning – a validation step

- Fourier analysis based learning is one of the most powerful query based learning algorithms known

OBDD-based learner

Circuit	Avg. Learn. Time	# of nodes	Accuracy	Circuit	Avg. Learn. Time	#of nodes	Accuracy
c432	1	280	99.77%	c499	12	3458	99.30%
c880	2	181	99.98%	c2670	8	28	98.14%
c1908	3	633	99.76%	c5315	67	1519	94.61%
c3540	48	6770	95.02%	c7552	44	362	99.52%
c6288**	2	227	100%	too large	38	5290	98.50%

Fourier-analysis based learner

Circuit	Avg. Learn. Time	# of Coeffs	Accu.	Circuit	Avg. Learn. Time	# of Coeffs	Accuracy
c432	33	53	69.10%	c499	5	1	99.00%
c880	21	29	85.00%	c2670	175	4	90.50%
c1908	5	8	99.70%	c5315	115	37	87.00%
c3540	18	31	85.60%	c7552	277	26	89.40%
c6288**	121	47	74.71%	too large	15	19	49.90%

Unique path selection

- The unique path selection ensures that the resulting OBDD will not be bigger than the actual OBDD of the true function
- This simple idea is crucial and allows the algorithm to perform better than the Fourier analysis method

Working with limited resources

- To keep run-time short and memory usage low, we can limit the maximum number of nodes allowed in any layer of the OBDD.
- Heuristic to select nodes to drop at each layer
- Although overall learning accuracy is lower, high learning accuracy can still be achieved within a restricted input space that consists of the selected nodes
- This is another perspective of simplification

The effect of restricted memory – another validation step

Experimental results for all outputs of benchmark circuit C3540

output	Unrestricted Accuracy	Restricted Accuracy	% of learned space	output	Unrestricted Accuracy	Restricted Accuracy	% of learned space
1	100	100	100	12	99	99	20.5
2	100	100	100	13	100	100	100
3	99.5	100	99.5	14	100	100	100
4	100	100	100	15	100	100	100
5	100	100	100	16	97.5	99	35.7
6	99	99	73.7	17	93.5	100	31.3
7	97.5	100	100	18	94.5	100	37.2
8	100	100	100	19	98.5	99	20.2
9	100	100	100	20	97.5	99.5	42.9
10	100	100	100	21	55	95	2.3
11	97.5	100	76.2	22	54	97.5	4.5

Accuracy in the restricted space only

Accuracy in the whole space

Extending learning to the Hybrid domain

- Suppose $x_1, \ldots, x_{n*k}$ are bit-level inputs and $op_1, \ldots, op_m$ are word-level inputs.
- To learn with a word-level output
  - We order these inputs as x1,…, x_{n*k}, op_1,…, op_m
  - The decision diagram is constructed with all the Boolean variables first
- Since the output is a word-level variable, comparisons are made over all the bits of the output rather than a single bit.

Boolean Vs. Hybrid learning

Boolean Learning

	Boolean variables	output
$f_{\alpha 0}$	0 0 1 … 1 0 1	1
	0 1 0 … 1 1 1	0
	0 0 1 … 1 0 0	1
	0 1 1 … 1 0 0	1
$f_{\alpha 1}$	1 0 1 … 1 0 1	1
	1 1 0 … 1 1 0	0
	1 0 1 … 1 1 1	1
	1 1 1 … 0 0 0	1

variable ordering →

Hybrid Learning

	Boolean variables	word-level variables	outputs
$f_{\alpha 0}$	0 0 1 0 0 0 … 0	1 0 … 1 0 1	1 … 1
	0 1 0 0 1 1 … 0	1 1 … 1 1 1	0 … 0
	0 0 1 1 1 0 … 1	1 0 … 1 0 0	1 … 0
	0 1 1 0 0 0 … 1	1 1 … 1 0 0	1 … 1
$f_{\alpha 1}$	1 0 1 0 0 0 … 0	1 0 … 1 0 1	1 … 1
	1 1 0 0 1 1 … 0	1 1 … 1 1 0	0 … 0
	1 0 1 1 1 0 … 1	1 0 … 1 1 1	1 … 0
	1 1 1 0 0 0 … 1	1 1 … 0 0 0	1 … 1

variable ordering →

The extension is quite natural

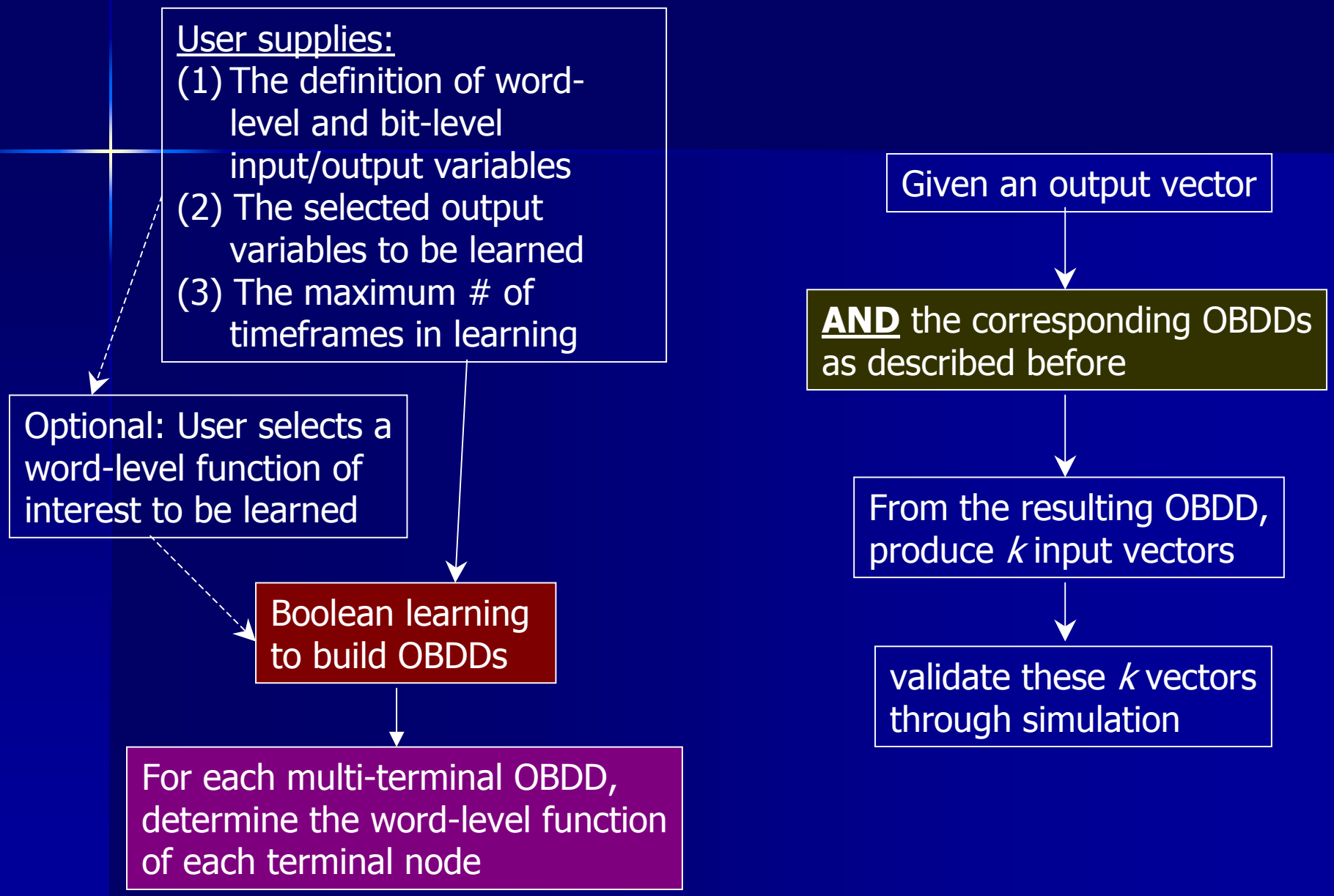
Extending learning to the Hybrid domain

- When we finish building the decision diagram for Boolean variables, we stop.
- At that point there may be more than two terminals so resulting diagram is a multi-terminal decision diagram (OMDD).
- Each terminal corresponds to a different word-level function
- In the next step, some of these word-level functions are determined.

Word-level learning

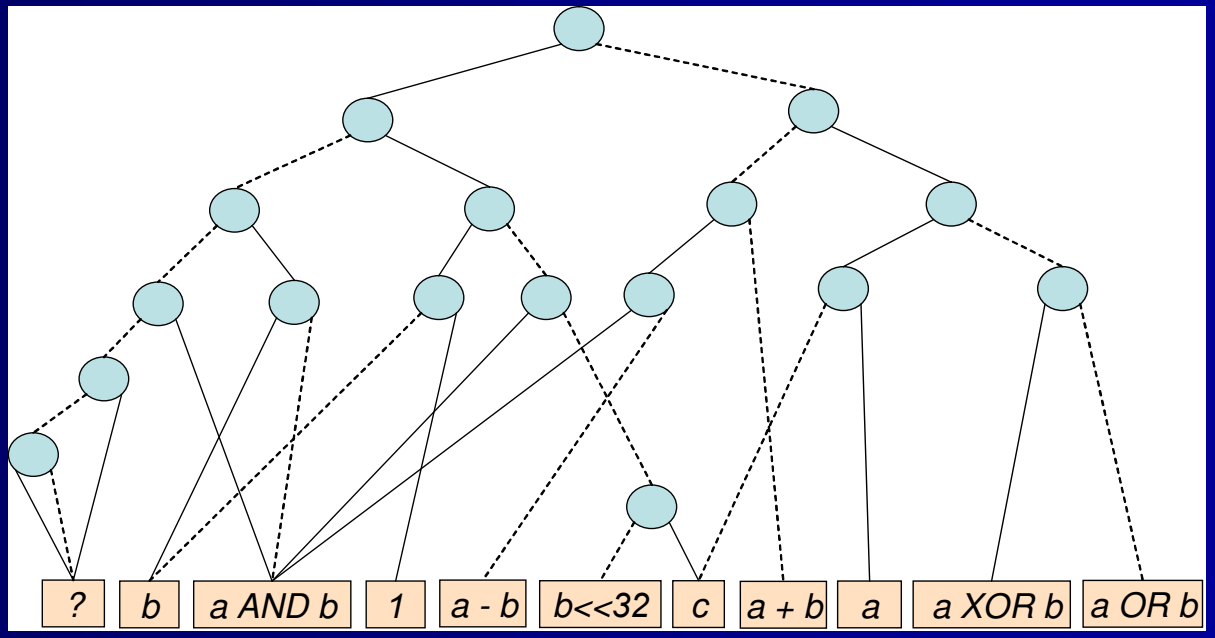
- For each terminal node, determine what function it performs
- Two possible solutions:
  - Template matching:
    - Commonly used word-level functions like addition, multiplication etc. are pre-built in a library
    - Data like "$<a,b> \rightarrow c$" are checked against each function to see "if c=a+b?" "if c=a*b?" "if c= $a \wedge b$?", etc.
  - Multivariate Polynomial interpolation:
    - MPI tries to find a polynomial to represent the function
    - MPI is applied only if template matching fails

Overall picture



Experiments with OpenRISC 1200

- A simplified view of the OpenRISC 1200 ALU



Accuracy of the view

- Learning rates using the simplified view
  - On randomly produced inputs

	Empirical learning accuracy
Word-level evaluation metric	83.70%
Bit-level evaluation metric	95.60%

Word-level evaluation metric: Percentage of correct word-level guesses over all guesses. A guess is considered correct only if all its bits are correct.

Bit-level evaluation metric: Percentage of correct bits over all guessed bits.

Future directions

- We currently work on:
  - A sequential data-mining engine for identifying sequential relations (certain timeframe relations between input and outputs etc.) that will serve as a preprocessing step
  - Try on Freescale processor
    - Mapping instructions to the boundary of a unit
  - Try on OpenSparc
    - Mapping from inputs to outputs of a unit