Cube Calculus Machine

PLAN OF EVOLVABLE AND LEARNING HARDWARE LECTURES

- **Our hardware** : the **DEC-PERLE-1** board.
  - Programming/designing environment for DEC-PERLE/XILINX.
  - Two different concepts of designing Learning Hardware using the DEC-PERLE-1 board.
- Compare **logic** versus ANN and GA approaches to learning.
- Introduce the concept of **Learning Hardware**
- Methods of **knowledge representation** in the **Universal Logic Machine (ULM):**
  - **variants of Cube Calculus**.

We are here →

- A general-purpose computer with instructions specialized to operate on logic data: **Cube Calculus Machine**.
  - Variants of cube calculus - arithmetics for combinatorial problems
  - Our approach to Cube Calculus Machine
- A processor for only one application: **Curtis Decomposition Machine.**

CUBE CALCULUS and other representations

STANDARD BINARY CUBE CALCULUS

- Represents product terms as cubes where the state of each input variable is specified by a symbol:
  - positive (1),
  - negative (0),
  - non-existing (a don't care) (X),
  - or contradictory (epsilon).
- Each of these symbols is encoded in **positional notation** with two bits as follows: 1 = 01, 0 = 10, X = 11, epsilon = 00.
- Positional notation for cube 0X1 is 10-11-01.
- Each position represents a state of the variable by the presence of "one" in it: left bit - value 0, right bit - value 1.
- This encoding presents simple reduction to **set-theoretical** representations

STANDARD BINARY CUBE CALCULUS

- A cube can represent :
  - a product, a sum,
  - a set of symmetry coefficients of a symmetric function,
  - a spectrum of the function,
  - or another piece of data on which some symbol-manipulation (usually set-theoretical) operations are executed.
- Usually the cube corresponds to a product term of literals.
- For instance, assume the following order of binary variables: **age** , **sex** and **color_of_hair.** Assume also that the discretization of variable **age** is:age = 0 for person's **age < 18** and **age = 1** otherwise
- Men are encoded by value 0 of attribute **sex** and women by value 1.
- **color_of_hair** is 0 for black and 1 for blond.
- A blond woman of age 19 is denoted by 110 and a black-hair seven-years old person of unknown sex is described by cube 0X1.
- Cube XXX is the set of all possible people for the selected set of attribute variables and their discretized values.

STANDARD BINARY CUBE CALCULUS (3)

- **Two-dimensional representation** is just a set of cubes where the connecting operator is implicitly understood as:
  - OR for SOP;
  - EXOR for ESOP;
  - concatenation for a spectrum,
  - or other.
- For instance, assuming each cube corresponding to AND operator and the OR being the connecting operator;
  - the list {0X1,110} is the SOP which represents the above mentioned two people (or a set of all people with these properties).
- Multi-valued and integer data can be encoded with binary strings in this representation,
  - so that next all operations are executed in binary (we use this model in the decomposition machine)

STANDARD BINARY CUBE CALCULUS (4)

- For instance, if there were three age categories, **young**, **medium** and **old**, they can be encoded as values 0, 1 and 2 of the ternary variable *age*, respectively.
- Variable **age** could be next represented in hardware as pair of variables **age_1** and **age_2**, where

  0 = 00, 1 = 01, 2 = 10,

- thus encoding:

young = NOT{**age_1**} NOT{**age_2**},

medium = NOT{**age_1**} {**age_2**},

old = age_1 NOT{age_2}.

MULTI-VALUED CUBE CALCULUS (MVCC)

- **A superset of CC.**
- It represents product terms as cubes where each input variable can have a **subset** of a finite set of all possible values that this variable can take.
- Each element of the set is represented by a single bit, which makes this representation not efficient for large sets of values.
- In the above example we could have for instance a 5-valued variable **age** for five age categories, and a quaternary variable **color_of_hair**
- Each position of a variable corresponds to its possible value.

MULTI-VALUED CUBE CALCULUS (MVCC)

- For instance, 10000-10-0100 describes a 7-year old boy with black hair
- This is an example of a minterm cube, i.e. with single values in each variable.
- 01100-11-1100 describes group **G_1** of people, men and women, that are either in second or in third age category and have either blond or black hair.
- This is an example of a cube that is not a minterm.
- 100000-00-1000 describes a first-category-of-age person with blond hair who has some conflicting information in **sex** attribute, for instance a missing value (this is also how contradictions are signalized during cube calculus calculations).
- The hardware operations in MVCC are done directly on such MV variable cubes so that the separate encoding to binary variables is not necessary.

GENERALIZED MV CUBE CALCULUS

- A superset of MVCC.
  - Each **output variable** can be also a subset of values.
  - Such cubes can be directly used to represent MV relations, as in **Table 2**
- Its operations are more general than MVCC, because more interpretations can be given to cubes
- This calculus has more descriptive power, but the respective hardware processors are much **more complicated.**

SIMPLIFIED BINARY CUBE CALCULUS

- A subset of CC. It operates only on **minterms** .
- It has application in decomposition of functions. **Minterms** can be of different dimensions.
- The hardware is much simplified: operations are only set-theoretical.
- This is **the simpliest virtual machine** realized by us, so larger data can be processed by it because more of a machine can be fit to the **limited FPGA Array resources** of DEC-PERLE-1.

SIMPLIFIED MV CUBE CALCULUS

- Cubes where for every input variable either
  - **only a single value** of its possible values is selected (which is denoted by a binary code (such as a byte) of a symbol corresponding to this value),
  - the variable is missing (which is denoted by a selected symbol, X),
  - or the variable is contradictory (another symbol, emptyset).
- Used for Rough Sets (Pawlak) and variable-valued logic (Michalski).
- For instance, assuming 10 age categories,
  - **0 = 0 - 10 years,**
  - **1 = 10 - 19 years,**
  - **2 = 20 - 29 years, etc,**
  - **and 3 hair categories: 0 = blond, 1 = black, 2 = red,**
  - the 7-year old boy with black hair is described as 0-0-1,
  - the 18-year old girl with black hair is described as 1-1-1,
  - the 28-year old woman with red hair is described as 2-1-2, and
  - a set of all people with red hair is X-X-2

SIMPLIFIED MV CUBE CALCULUS

- There is no way now to describe in one cube people below 19 with red or black hair, which was possible in MVCC or GMVCC.
- This simplification of the language brings however big speedup of algorithms and storage reduction when applied for data with many values of attributes.
- The control of algorithms becomes more complicated, while the data path is simplified.

SPECTRAL REPRESENTATIONS

- Examples: Reed-Muller FPRM and GRM spectra, Walsh spectrum, various orthogonal spectra.
- These representations represent function as a sequence of spectral coefficients or selected coefficient values with their numbers.
- Some spectral representations are useful to represent data for **genetic algorithms**: the sequence of spectral coefficients is a chromosome.
- For instance, in the Fixed-Polarity Reed-Muller (FPRM) canonical AND/EXOR forms for n variables, every variable can have two polarities, 0 and 1.
- Thus there are $\mathbf{2^n}$ different polarities for a function and the GA algorithm has to search for the polarity that has the minimum number of ones in the chromosome.

SPECTRAL REPRESENTATIONS

- This way, every solution is correct, and the fitness function is used only to evaluate the cost of the design (100% correctness of the circuit is in general very difficult to achieve in GA.
- Therefore our approaches to logic synthesis based on GA are to **have a representation that provides you with 100 % correctness** and have the GA search only for net minimization.
- This approach involves however a more difficult fitness function to be calculated in hardware than the pure GA or Genetic Programming approaches.
- Similarly, the other AND/EXOR canonical form called the Generalized Reed-Muller form (GRM) has **$n\ 2^{\{n-1\}}$** binary coefficients, so there are **$2^{\{n\ 2\wedge\{n-1\}}$** various GRM forms.

SPECTRAL REPRESENTATIONS

- Because there are more GRM forms, it is more probable to find a shorter form among them than among the FPRM forms.
- But the chromosomes are much longer and the evaluation is more difficult.
- This kind of trade-offs is quite common in spectral representations.
- Spectral methods allow for high degree of parallelism.

ROUGH PARTITIONS AND LABELED ROUGH PARTITIONS

- Rough Partitions (RP) represented as Bit Sets (Luba).
- This representation stores the two-dimensional table column-wise, and not row-wise as MVCC does.
- In r-partition every variable (a column of a table) induces a partition of the set of rows (cubes) to blocks, one block for each value the variable can take (there are two blocks for a binary variable, and **k** blocks for a **k**-valued variable).
- Rough Partitions are a good idea but they don't really form a representation of a function.
- Since the values of a variable are not stored together with partition blocks, the essential information on the function is lost and the original data can not be recovered from it.
- This is kind of an abstraction of a function, useful for instance in various decomposition algorithms.

LABELED ROUGH PARTITIONS

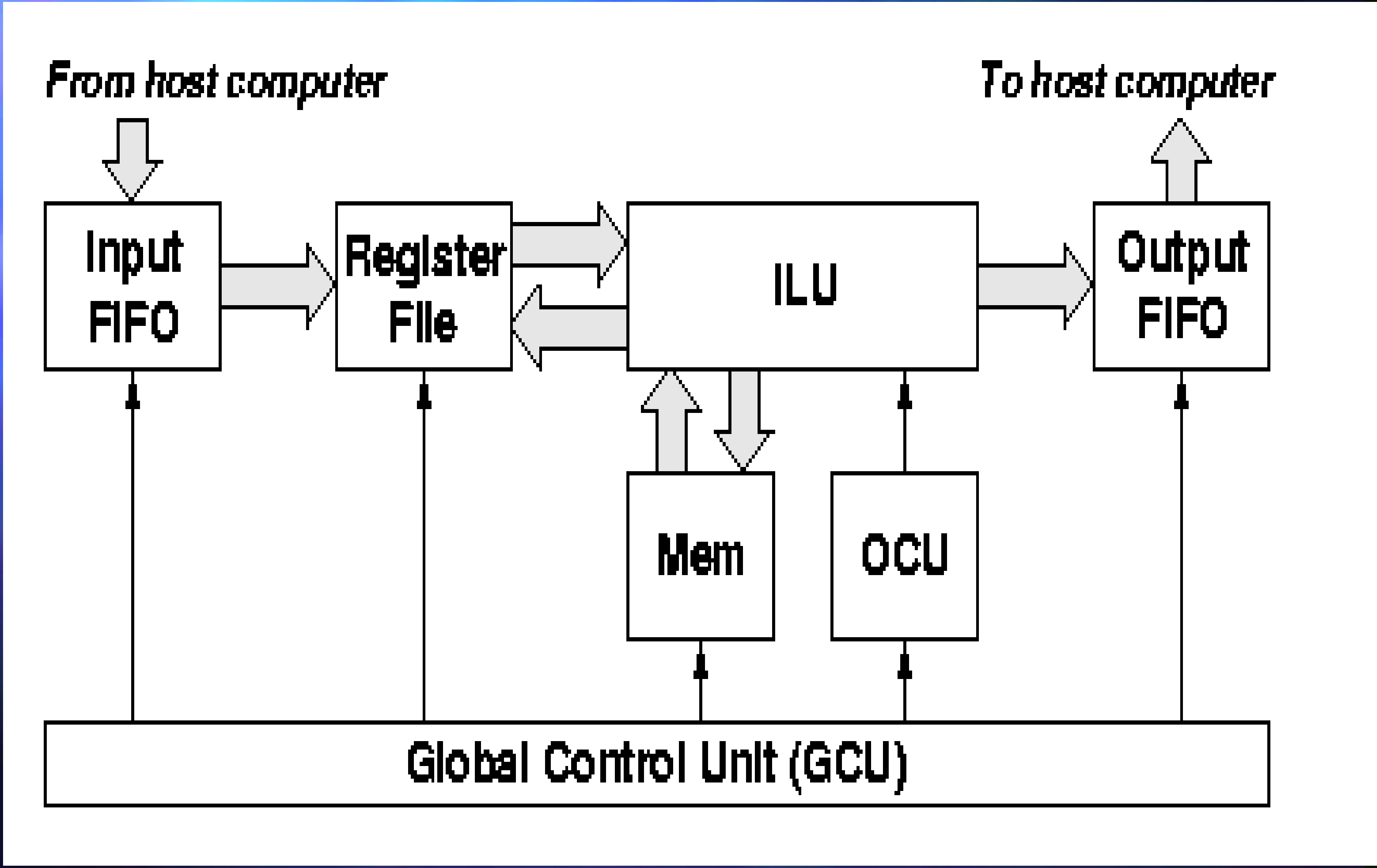
- A generalization of RS which has very interesting properties and allows to find different kind of patterns in data.
- It is useful for decomposition of MV relations and it preserves all information about the relation or function.
  - It can be also made canonical, when created for special cubes.
- Most of its operations are reduced to set-theoretical operations, so hardware realization is relatively easy.
- Relations happen in tables created from real data-base and features from images,for instance, MV relations are benchmarks **hayes**, **flare1**, flare2 from Irvine

LABELED ROUGH PARTITIONS (2)

- An example of application of relation in logic synthesis area is a modulo-3 counter (a non-deterministic state machine is a special case of multiple-valued, multi-output, relation) that counts in sequence **s0 -> s1 -> s2 -> s0** and if the state **s3** happens to be the initial state of the counter, counter should transit to any of the states **s0,s1,s2**, but not to the state **s3** itself.
- Generalized values for input variables are already known from cube calculus but **generalized values for output variables** are a new concept which allows for representation and manipulation of relations in LRP.

CUBE
CALCULUS
MACHINE

Simplified idea of the Cube Calculus Machine



CUBE CALCULUS MACHINE

•In our design, the Cube Calculus Machine is a coprocessor to the host computer and is realized as a virtual processor in DEC-PERLE-1.

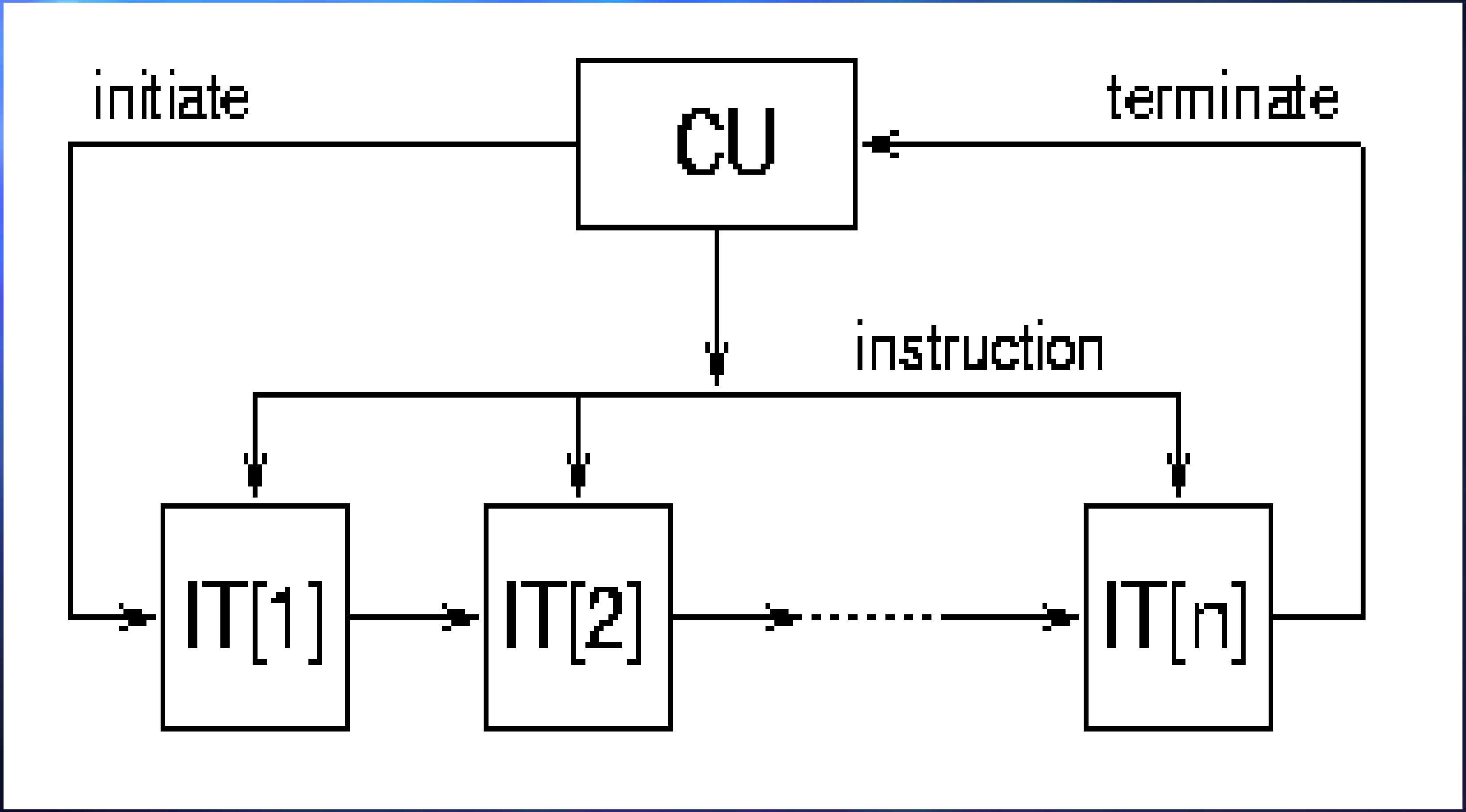
the CCM communicates with the host computer through the input and the output FIFO.

The Iterative Logic Unit (ILU) is realized using a one-dimensional iterative network of combinational modules and cellular automata.

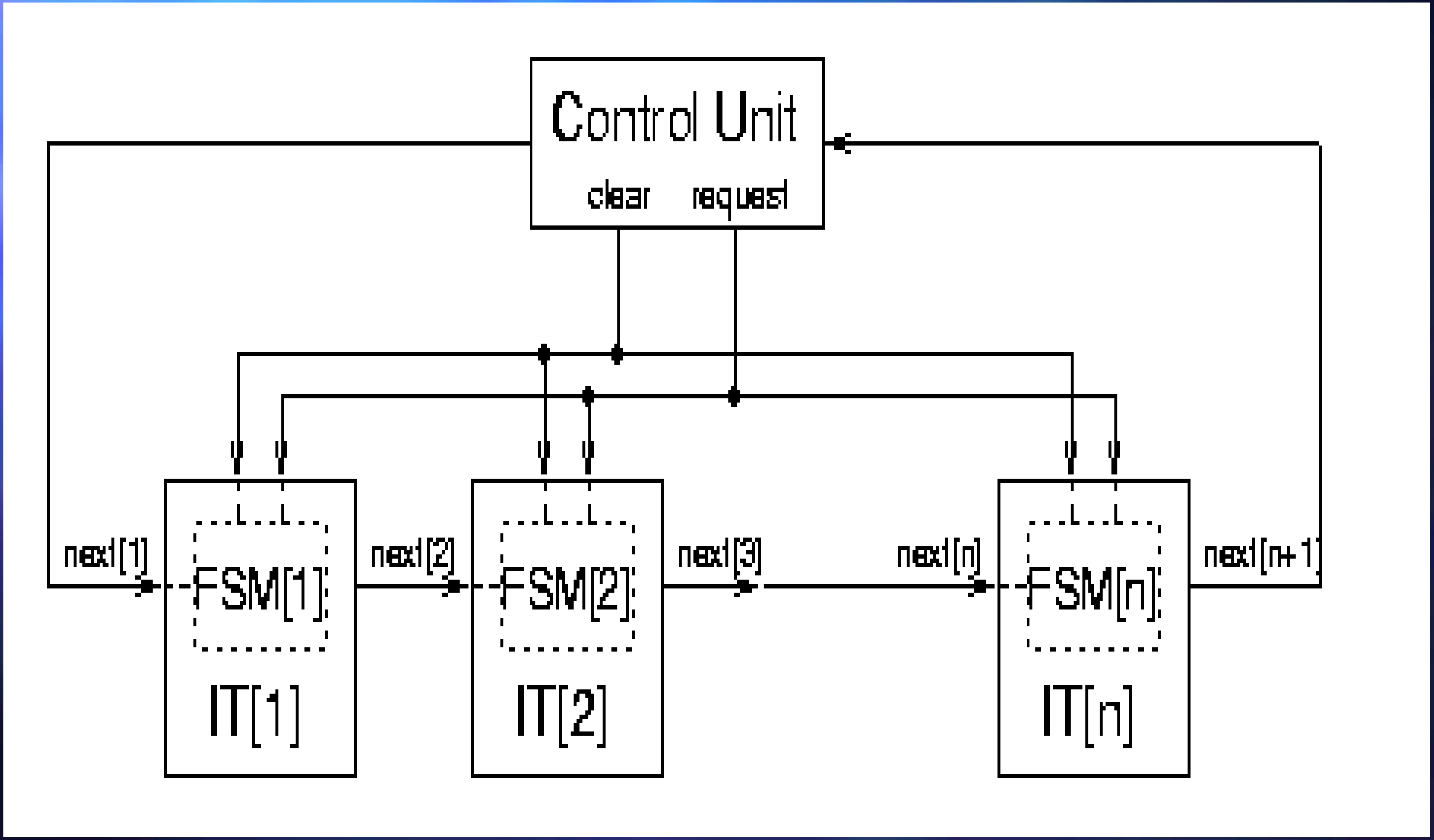
ILU is composed from ITs, each of them processes a single binary variable or two values of a multi-valued variable.

Any even number of variables can be processed, and only size of the board as well as bus limitations are the limits (it is the total of 32 values now, which is at most 16 binary variables, 8 quaternary variables, or 4 8-valued variables, or any mixture of even-valued variables).

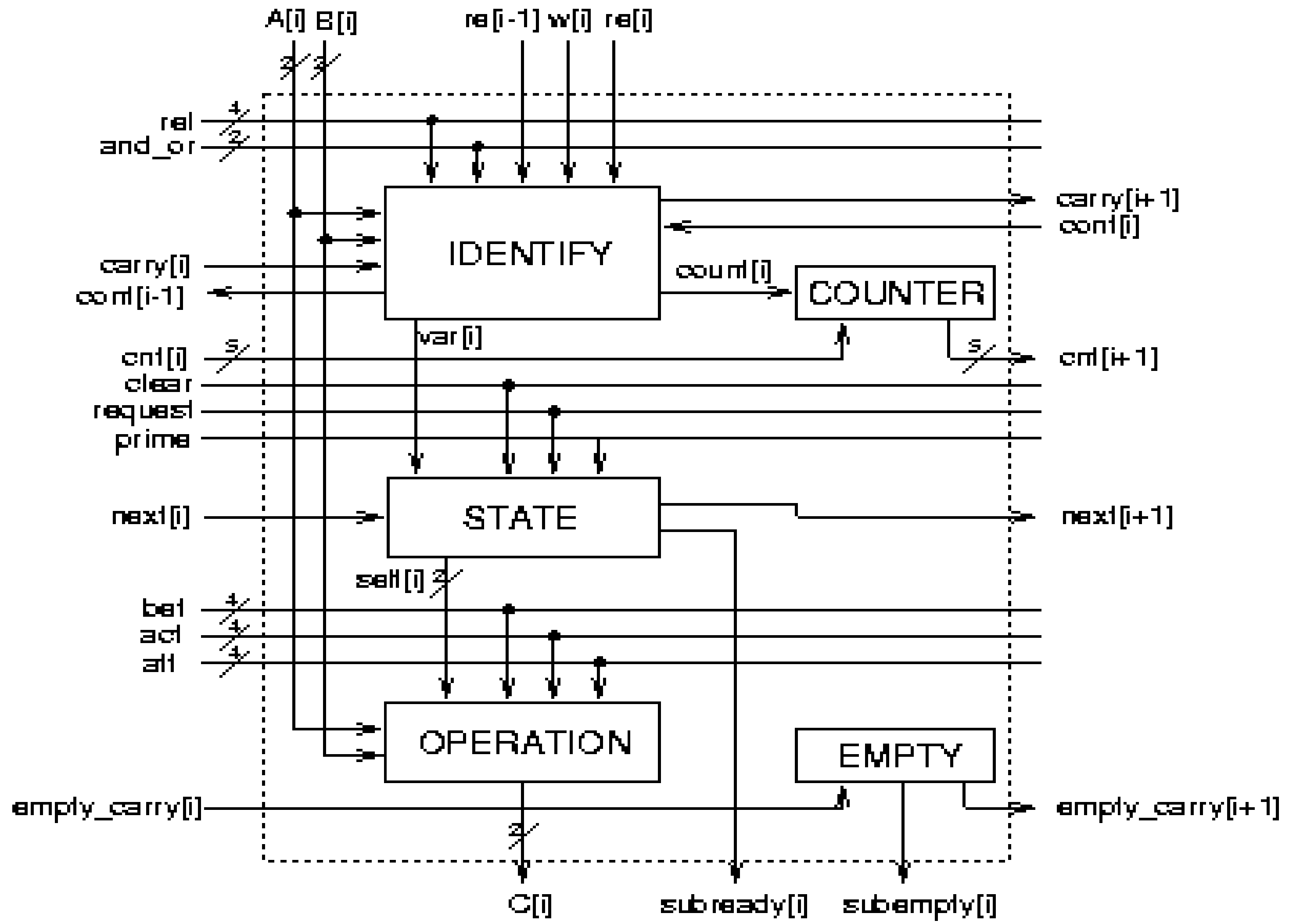
CCM as a SIMD machine



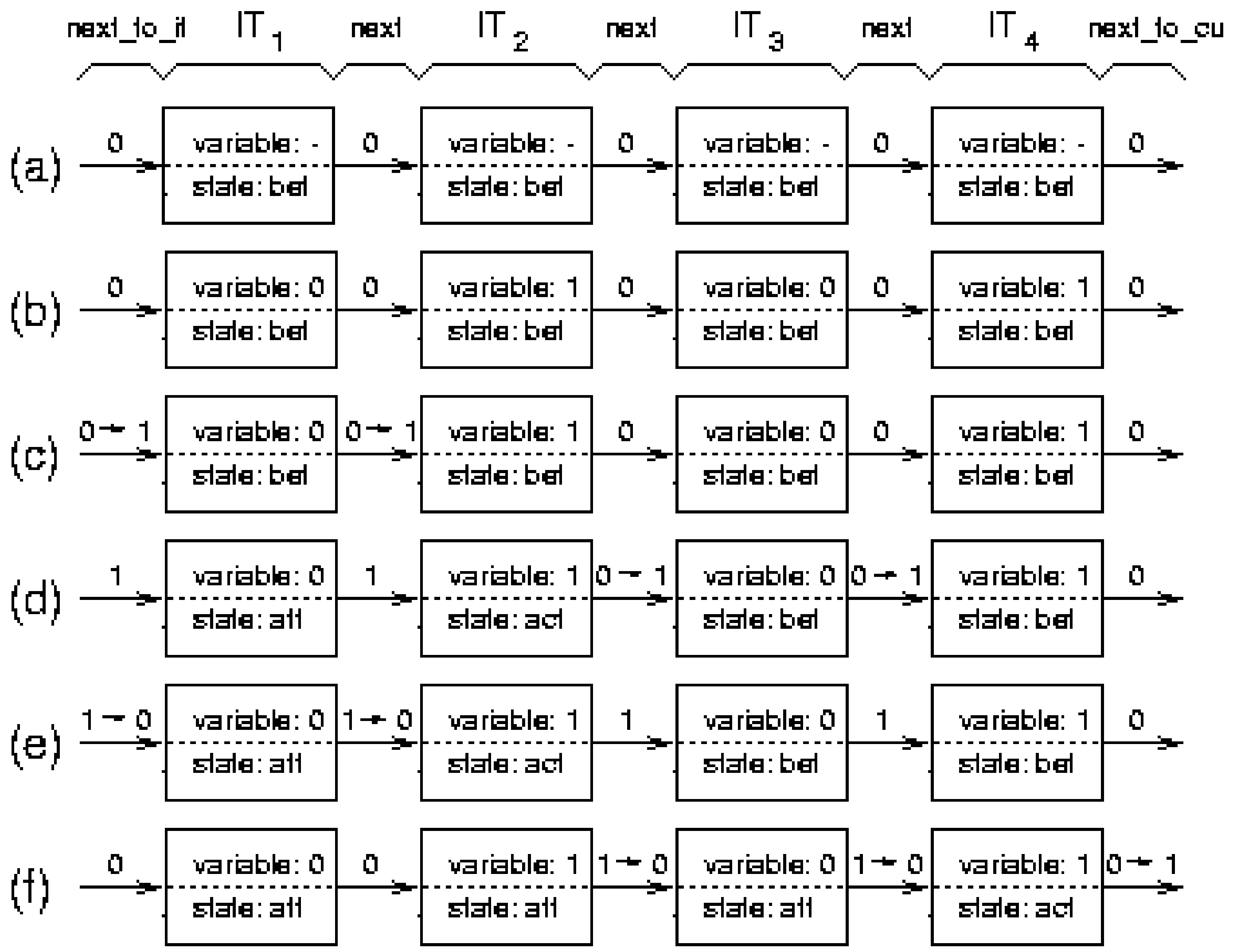
CCM as a Programmable Cellular Automaton



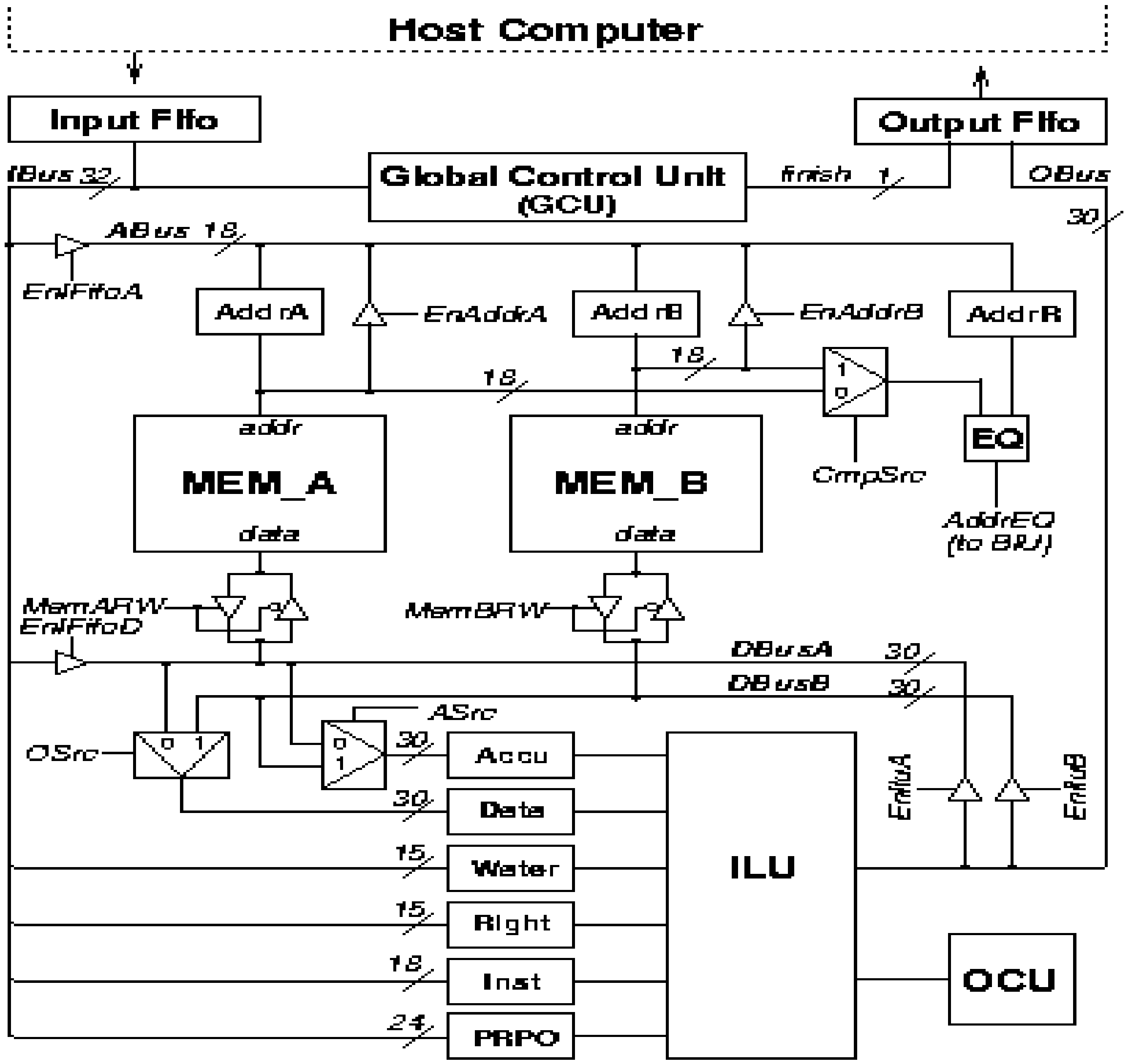
Details of the single cell of Iterative Logic Unit of CCM



Propagation of information in Cellular structure



General Architecture of CCM, new version



The ILU can take the input from register file and memory, and can write output to the register file, the memory, and the output FIFO.

The ILU executes the cube operation under the control of **Operation Control Unit** (OCU).

The **Global Control Unit** (GCU) controls all parts of the CCM and let them work together.

- The machine realizes the set of operations from Table 3.

- The Table shows also their programming information. Each row of Table describes one cube operation.
- Each operation is specified in terms of:
    - **rel** - the elementary relation type between input values,
    - and/or - the global relation type, and the internal state of the elementary cellular automaton - before,active and after .
- The operation name, notation, the output value of rel (partial relation) function in every IT, and_or (relation type), the output values of before , active and after functions are listed from left to right.

- Partial relation rel is an elementary relation on elementary piece of data (pair of bits).

- These set theoretical relations such as inclusion, equality, etc.

- The value of and_or equals to 1 means that the relation type is of AND type; otherwise, the relation type is of OR type.

- This relation is created by composing elementary relations from ITs and variables.

The Output Values of Bitwise Functions Used in Cube Operations

Operation	Notation	Relation		Output Function		
		rel	*and/or*	*before*	*active*	*after*
crosslink	$A \uplus B$	1110	1	0011	0111	0101
sharp	$A \#_{basic} B$	0010	0	0011	0010	0011
disjoint sharp	$A \# d_{basic} B$	0010	0	0011	0010	0001
consensus	$A *_{basic} B$	1111	1	0001	0111	0001
intersection	$A \cap B$	–	–	0001	–	–
super cube	$A \cup B$	–	–	0111	–	–
prime	$A' B$	0001	0	0011	0111	–
cofactor	$A \|_{basic} B$	1011	1	0001	1111	–

•The machine is microprogrammable both in its OCU control unit part (by use of CCM Assembly Language) and in Data Path, as achieved by ILU operations programmability.

For instance, each operation is described by the binary pattern corresponding to it in the respective row of Table 3.

By creating other binary patterns in the fields of Table 3, new operations can be programmed to be executed by ILU.

As the reader can appreciate, there are very many such combinations, and thus CCM micro-operations.

•We call this **horizontal data-path microprogramming** .

Higher order CCM operations are created by sequencing low-level operations.

This is called **vertical control microprogramming** and is executed by OCU (within ILU) and GCU (for operations with memories and I/O).

Thus, the user has many ways to (micro) program sequences of elementary instructions.

This is done in CCM Assembly language.

Evaluation.

- For comparing the performance of the CCM and that of the software approach, a program to execute the disjoint sharp operation on two arrays of cubes was created using C language.
- Then this program and the CCM are used to solve the following problems:
  - (1) Three variables problem: $1 \#$ (all minterm with 3 binary variables).
  - (2) Four variables problem: $1 \#$ (all minterm with 4 binary variables).
  - (3) Five variables problem: $1 \#$ (all minterm with 5 binary variables).
- The C program is compiled by GNU C compiler version 2.7.2, and is run on Sun Ultra5 workstation with 64MB real memory.

Evaluation.

- The CCM is simulated using QuickHDL software from Mentor Graphics.
- We simulated the VHDL model of CCM, got the number of clocks used to solve the problem, then calculated the time used by CCM using formula: clock *$ clock-period.
- A clock of 1.33 MHz (clock period: 750 ns) is used as the clock of the CCM.

Compare CCM (1.33 MHz) with software approach

Problem	3 variables	4 variables	5 variables
Ultra5	111 usec	268 usec	812 usec
CCM	546×0.75 $= 409$ usec	1285×0.75 $= 963.75$ usec	3405×0.75 $= 2553.75$ usec
speedup	0.27	0.28	0.32

Evaluation.

- It can be seen from Table that our CCM is about 4 times slower than the software approach.
- But, the clock of the CPU of Sun Ultra5 workstation is 270 MHz, which is 206 times faster than the clock of the CCM.
- Therefore, we still can say that the design of the CCM is very efficient for cube calculus operations.

Evaluation.

- It also can be seen from Table that the more variables the input cubes have, the more efficient the CCM is.
- This is due to the software approach need to iterate through one loop for each variable that is presented in the input cubes.

Evaluation.

- However, the clock period of 750ns is too slow.
- From the state diagram of the GCU, it can be found that the delays of **empty carry path** and **counter carry path** only occur in a few states.
- Thus, if we can just give more time to these states, then we can speedup the clock of the whole CCM.
- This is very easy to achieve: for example, the state P2 of GCU needs more time for the delay of counter carry path, so add two more states in series between states P2 and P3.

Evaluation.

- These two extra states do nothing but give the CCM two more clock periods to evaluate the signal **prel_res,** which means that the CCM has 3 clock periods to evaluate signal **prel_res** in state P2 after adding two more ``delay'' states.
- After making similar modifications to all these kind of states, the CCM can run against a clock of 4 Mhz (clock period of 250 ns).

COMPARE CCM (4MHZ) WITH SOFTWARE APPROACH

Problem	3 variables	4 variables	5 variables
Ultra5	111 usec	268 usec	812 usec
CCM	611×0.25 $= 152.75$ usec	1486×0.25 $= 371.5$ usec	4078×0.25 $= 1019.5$ usec
speedup	0.72	0.72	0.80

Evaluation.

- It is very hard to increase the clock frequency again with this mapping because some other paths like **memory path** have delays greater than 150 ns.

RESULTS OF COMPARISON

- A design like CCM with a complex control unit and complex data path is not good for the architecture of the DEC-PERLE-1 board.
- It can be seen from our CCM mapping that since a lot of signals must go through multiple FPGA chips, this leads to greater signal delays.
- For instance, if we can connect the memory banks and the registers directly, then the memory path has a delay of only 35 ns. But our current memory path has a delay of 160 ns.
- Another issue is that XC3090 FPGA is kind of ``old'' now (6 to 8 years old technology).
- The latest FPGAs from Xilinx or other vendors have more powerful CLBs and more routing resource, and they are made using deep sub-micron process technology.

POSSIBLE IMPROVEMENTS

- Mapping the entire CCM inside one FPGA chip would speedup the CCM:
- If we map entire CCM into one FPGA chip, the signals do not need to go through multiple chips again, which means the routing delay is reduced.
- Since the new FPGA chip has more powerful CLBs and routing resource, we can map the CCM denser. This also reduces the routing delays.
- Since new FPGA chips are made using deep sub-micron technology, the delay of CLB and routing wires are both reduced.
- For example, the delay of the CLB of XC3090A is 4.5 ns while the delay of CLB of XC4085XL (0.35 micron technology) is only 1.2 ns. This means that it is very easy to achieve 3 times faster mapping.

NEW FPGA CHIPS FOR NEW VERSION

- XC4085XL FPGA from Xilinx has a CLB matrix of **56 * 56** and up to 448 user I/O pins.
- The CCM should be able to map into one XC4085XL FPGA.
- It should not be difficult to run the CCM against a clock of 20 MHz (clock period: 50 ns).
- This means that our CCM will be about 4 times faster than the software approach while the system clock of the CCM is still 5 times slower than that of the workstation.

CONCLUSIONS

- Principles of Learning Hardware as a competing approach to Evolvable Hardware, and also as its generalization.
- Data Mining machines.
- Universal Logic Machine with several virtual processors.
- DEC-PERLE-1 is a good medium to prototype such machines, its XC3090A chip is now obsolete.
- This can be much improved by using XC4085XL FPGA and redesigning the board.
- Massively parallel architectures such as CBM based on Xilinx series 6000 chips will allow even higher speedups.