

ECE 510 OCE  
BDDs and Their Applications

**Lecture 18.**  
**Implicit Multi-Output Decomposition (2)**  
**Finite State Machine Decomposition**

May 25, 2000  
Alan Mishchenko

---

---

---

---

---

---

---

---

**Overview**

- Implicit multi-output decomposition (part 2) (Wurth/Eckl/Legl, DAC'95)
  - finding the global partition
  - implicitly selecting next preferable function
  - Lmax algorithm (T.Kam, DAC'94)
- Parallel and serial FSM decomposition
  - deriving the lattice of S.P. partitions
  - find good partitions (partition pairs)
  - performing decomposition implicitly using transition/output relations

May 25, 2000 ECE 510 OCE: BDDs and Their Applications 2

---

---

---

---

---

---

---

---

**Operations on Partitions**

- The product of partitions  $\pi_1$  and  $\pi_2$  on  $S$  is the partition  $\pi_1 \cdot \pi_2$  on  $S$  such that  $s \equiv t(\pi_1 \cdot \pi_2)$  iff  $s \equiv t(\pi_1)$  and  $s \equiv t(\pi_2)$ .
- The sum of partitions  $\pi_1$  and  $\pi_2$  on  $S$  is the partition  $\pi_1 + \pi_2$  on  $S$  such that  $s \equiv t(\pi_1 + \pi_2)$  iff there is a sequence in  $S$ ,  $s = s_0, s_1, \dots, s_n$ , such that  $s_n = t$  and either  $s_i \equiv s_{i+1}(\pi_1)$  or  $s_i \equiv s_{i+1}(\pi_2)$ ,  $0 \leq i \leq n-1$ .

May 25, 2000 ECE 510 OCE: BDDs and Their Applications 3

---

---

---

---

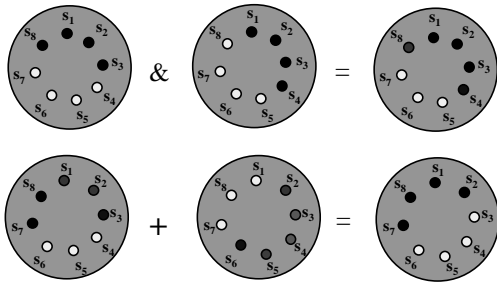
---

---

---

---

## Interpretation of Operations



May 25, 2000

ECE 510 OCE: BDDs and Their Applications

4

---

---

---

---

---

---

---

---

## Implicit Partition Manipulation

- Partitions can be manipulated similar to sets, using characteristic functions
- The product of partitions is the product of their char functions
- The sum of partitions is the sum of their char functions followed by computation of transitive closure

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

5

---

---

---

---

---

---

---

---

## Example: Product of Partitions

	0 0 1 1 0 1 1 0	&	0 0 1 1 0 1 1 0	=	0 0 1 1 0 1 1 0
00	1 0 0 0		1 1 0 0		1 0 0 0
01	0 1 1 1		1 1 0 0		0 1 0 0
11	0 1 1 1		0 0 1 1		0 0 1 1
10	0 1 1 1		0 0 1 1		0 0 1 1

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

6

---

---

---

---

---

---

---

---

### Example: Sum of Partitions

	$s_1$	$s_2$	$s_3$	$s_4$
$s_{1,00}$	1	1	0	0
$s_{2,01}$	1	1	0	0
$s_{3,11}$	0	0	1	0
$s_{4,10}$	0	0	0	1

	$s_1$	$s_2$	$s_3$	$s_4$
$s_{1,00}$	1	0	0	0
$s_{2,01}$	0	1	1	0
$s_{3,11}$	0	1	1	0
$s_{4,10}$	0	0	0	1

	$s_1$	$s_2$	$s_3$	$s_4$
$s_{1,00}$	1	1	0	0
$s_{2,01}$	1	1	1	0
$s_{3,11}$	0	1	1	0
$s_{4,10}$	0	0	0	1

after transitive closure

	$s_1$	$s_2$	$s_3$	$s_4$
00	1	1	1	0
01	1	1	1	0
11	1	1	1	0
10	0	0	0	1

May 25, 2000 ECE 510 OCE: BDDs and Their Applications 7

---

---

---

---

---

---

---

---

	000	001	010	011	100	101	110	111
00	0	0	0	1	0	1	1	1
01	1	1	1	1	1	1	1	0
10	1	1	1	1	1	1	1	0
11	0	0	0	1	0	1	1	0

$F_1(x)$

	000	001	010	011	100	101	110	111
00	0	0	0	1	0	1	0	1
01	0	1	1	1	1	1	1	0
10	0	1	1	1	1	1	1	0
11	1	1	1	0	1	0	1	0

$F_2(x)$

May 25, 2000 ECE 510 OCE: BDDs and Their Applications 8

---

---

---

---

---

---

---

---

$D_2(x)$   $D_1(x)$   $D_3(x)$   $F_1(x)$   $F_2(x)$

May 25, 2000 ECE 510 OCE: BDDs and Their Applications 9

---

---

---

---

---

---

---

---

## Decomposition Algorithm

- Find the global partition of all outputs
- Find the characteristic function of all preferable (constructable and assignable) decomposition functions
- Use Lmax algorithm to select the best preferable function (function which can be used to decompose the most outputs)
- Iterate the above steps until the decomposition is selected

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

10

---

---

---

---

---

---

---

---

## Finding Global Partition

- Wurth et al does not say how they implement this step
- One of the ways is to find the char func of a output partition as a column compatibility relation and take the product all partitions
  - for completely specified functions
$$CR(b,b') = \forall f [ F(b,f) \leftrightarrow F(b',f) ]$$
  - for incompletely specified functions
$$CR(b,b') = \forall f \exists v [ R(b,f,v) \leftrightarrow R(b',f,v) ],$$
$$R(b,f,v) = v' \& F^0(b,f) + v \& F^1(b,f) + F^2(x)$$

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

11

---

---

---

---

---

---

---

---

## Assignability and Constructability

- A function is assignable at a certain encoding step, if selecting it as one of the decomposed function can lead to a decomposition
- A function is constructable if it can be expressed in terms of the blocks of the global partitions: by assigning some block to the ON-set, some blocks to the OFF-set of this function
- It is proved (Wurth et al) that to find all optimal decompositions it is enough to consider preferable (assignable and constructable) functions

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

12

---

---

---

---

---

---

---

---

## Implicit Preferable Function Selection

- If the global partition is composed of  $p$  blocks, the number of decomposition functions is  $q \geq \lceil \log_2(p) \rceil$
- Let us select the next decomposition function in such a way that it is shared by the max number of outputs
- For this purpose, we implicitly build the characteristic function of all preferable decomposition functions as a relation  $\chi(z,o)$  over variables  $z$  that encode decomposition function in terms of global partition blocks and variables  $o$  representing network outputs
- Using Lmax algorithm, the assignment of  $z$  is selected that corresponds to the maximum number of  $o$ 's.

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

13

---

---

---

---

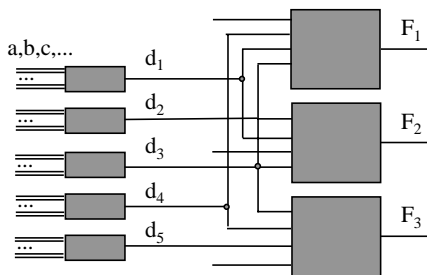
---

---

---

---

## Example: 3 outputs, 5 functions



May 25, 2000

ECE 510 OCE: BDDs and Their Applications

14

---

---

---

---

---

---

---

---

## Lmax Operator

- Given the binary relation  $F(r,c)$  as a BDD, the problem is to find a subset of the assignments of  $c$ , each of which relates to the maximum number of assignment of  $r$  in  $F(r,c)$
- The algorithm takes the relation  $F(r,c)$  and set of variables  $r$  as a bdd cube
- The algorithm returns (1) the bdd representing all assignments of  $c$  that are connected to the maximum number of  $r$  and (2) the maximum number itself

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

15

---

---

---

---

---

---

---

---

## Pseudo-code for Lmax algorithm

```

(bdd, int) Lmax( bdd F, bdd r )
{
  if ( F == bddfalse ) return ( bddfalse, 0 );
  if ( F == bddtrue ) return ( bddtrue, 2^|support(r)| );
  // check the cache for results
  v is the top-most variable in F;
  if ( v ∈ r ) return ( bddtrue, bdd_count_onset(F) );
  (F0, Count_F0) = Lmax( bdd_low(F), r );
  (F1, Count_F1) = Lmax( bdd_high(F), r );
  Count = max( Count_F0, Count_F1 );
  if ( Count_F0 == Count_F1 ) Res = ITE( v, F1, F0 );
  else if ( Count == Count_F0 ) Res = ITE( v, 0, F0 );
  else if ( Count == Count_F1 ) Res = ITE( v, F1, 0 );
  // insert the result into cache
  return ( Res, Count );
}

```

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

16

---

---

---

---

---

---

---

---

---

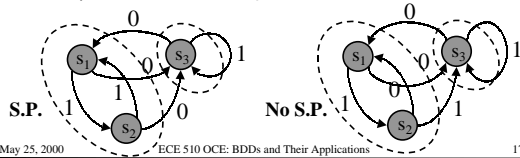
---

## FSM Decomposition

- This approach to FSM decomposition is based on partitions with substitution property (S.P. partitions)
- A partition on the states of FSM  $M = \{S, I, O, \delta, \lambda\}$  is said to satisfy S.P. iff

$$\forall s, t \in S: s \equiv t(\pi) \Rightarrow \forall a \in I: \delta(a, s) \equiv \delta(a, t)(\pi)$$

- Example:  $\pi = \{ \{s_1, s_2\}, \{s_3\} \}$



May 25, 2000

ECE 510 OCE: BDDs and Their Applications

17

---

---

---

---

---

---

---

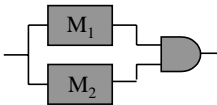
---

---

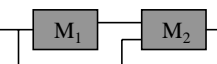
---

## Existence of Decomposition

- Property 1. An FSM has a non-trivial parallel decomposition iff there exist two S.P. partitions  $\pi_1$  and  $\pi_2$  such that  $\pi_1 \cdot \pi_2$ .



- Property 2. An FSM has a non-trivial serial decomposition iff there is an S.P. partitions  $\pi_1$ .



May 25, 2000

ECE 510 OCE: BDDs and Their Applications

18

---

---

---

---

---

---

---

---

---

---

## Computation of S.P. Partitions

- Find the transition/output relation of the FSM
- Find the char func of all subsets of two states
- Find the set of all minimal S.P. partitions (partitions induced by the two-state subsets)
- Find the set of all S.P. partitions (sub-lattice) by iteratively summing the minimal S.P. partitions first among themselves, next with the results of summing at previous iterations

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

19

---

---

---

---

---

---

---

---

## Char Function of Tuples "k out of n"

- Problem: Given the set S of n elements, build a BDD for the characteristic function of the set of subjects containing exactly k out n elements of set S
- The brute force approach results in creating the sum of  $n!/k!(n-k)!$  cubes, each of which represents a characteristic function of a subset
- The intelligent approach uses a recursive BDD procedure to build the char function

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

20

---

---

---

---

---

---

---

---

## Building Char Function of Tuple Set

```
bdd Tuples( int k, int n)
{
  if ( k < 0 || n < k )      return bddfals;
  if ( k == 0 && n == 0 )   return bddtrue;
  // check cache for results
  bdd F0 = Tuples( k,  n-1 );
  bdd F1 = Tuples( k-1, n-1 );
  bdd Res = bdd_ite( bdd_ithvar(n), F1, F0 );
  // insert into cache
  return Res;
}
```

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

21

---

---

---

---

---

---

---

---

## Computing One Minimal S.P. Partition

```
bdd FindSPPartition( bdd T(i,x,z), bdd InitialPartition(x,y) )
{ // compute the partition block characterization function
  bdd CF(x,y) = CProjection(InitialPartition(x,y) , CubeYVars );
  bdd AllBlocks = bddfalse; bdd NewBlocks = CF(x,y);
  // iteratively compute the char func of induced partitions
  do {
    AllBlocks(x,y) |= NewBlocks(x,y);
    NewBlocks(x,y) =  $\exists x \exists z$  [ T(i,x,z) & OldBlocks(x,y) ] |z→x;
  } while ( AllBlocks != NewBlocks );
  // check whether this is a non-trivial partition
  if (  $\forall y$  [ AllBlocks (x,y) ] == AllStates(x) ) return 1-Partition(x,y);
  else return Partition( AllBlocks (x,y) );
}
```

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

22

---

---

---

---

---

---

---

---

---

---

## Deriving Sub-Lattice of S.P. Partitions

- Given the char function of the set of minimal S.P. partitions, MinimalPartitions(a,x,y), the complete sub-lattice of S.P. partitions can be computed

```
bdd FindLattice( bdd MinimalPartition )
{ All(a,x,y) = MinimalPartitions(a,x,y);
  do { Sum(a,a',x,y) = ADD-partitions[ All(a,x,y), All(a',x,y) ];
    Sum(a,x,y) = Re-Encode( Sum(a,a',x,y) );
    New(a,x,y) = Sum(a,x,y) - All(a',x,y);
    All(a,x,y) |= New(a,x,y);
  } while ( New != bddfalse );
  return All(a,x,y);
}
```

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

23

---

---

---

---

---

---

---

---

---

---

## Re-Encoding

- Given relation  $R(x,y)$ , it is possible to re-encode it in such a way that it uses the minimum number of variables  $z$  to encode true assignments of variables  $y$

```
bdd Re-Encode( bdd R, bdd y )
{ if ( R = 0 || R = 1 ) return R; // caching does not work!!!!
  for all variables in y
    if ( Ry'(x,y) == 0 ) R = Ry'(x,y);
    else if ( Ry(x,y) == 0 ) R = Ry(x,y);
  yi is a variable in y such that Ry' and Ry have as close ON-set
  minterm count as possible;
  bdd R0 = Re-Encode( Ry' );
  bdd R1 = Re-Encode( Ry );
  return ITE( NewVar, R1, R0 );
}
```

May 25, 2000

ECE 510 OCE: BDDs and Their Applications

24

---

---

---

---

---

---

---

---

---

---