Figures:

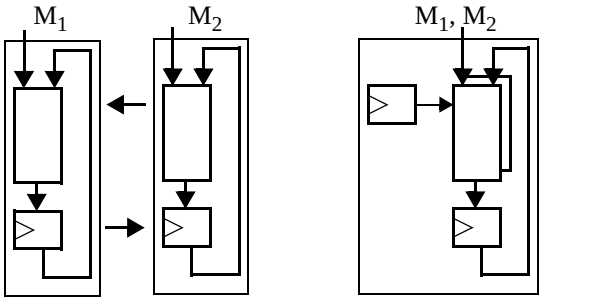

FIGURE 1. Structural decomposition of FSM

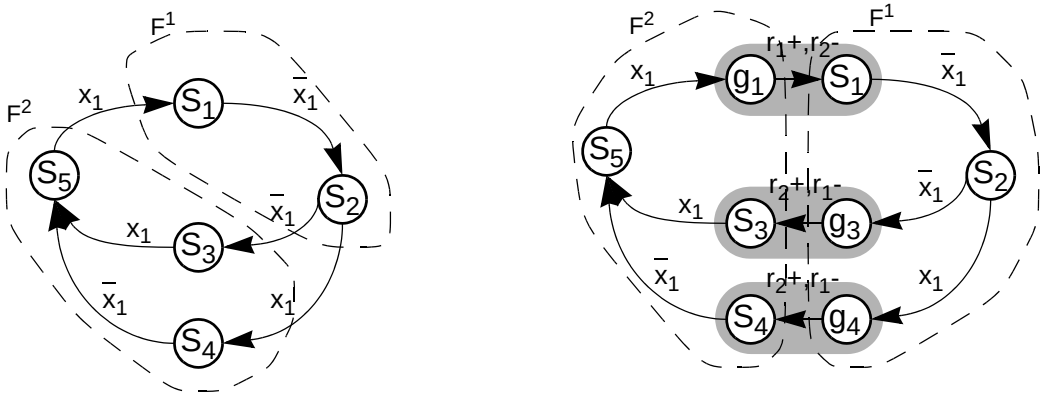

FIGURE 2. Example, a) Monolithic FSM with state partition indicated, b) Coupled states introduced

B:	b_1	b_2	b_3
F^1	g_1	s_3	s_4
F^2	s_1	g_3	g_4

FIGURE 3. Example, Coupled state table

```
struct subFSM {
    set of int S, G, Q;
}

set of struct subFSM F;
int sb[n, max(⌈ log2|U^m| ⌉)] ← null;

assignCoupledStates(set of struct subFSM F, int sb)

    int i,j ← 1;
    for all f ∈ F {
        for all q ∈ f.Q {
            i ← indexOf(f);
            sb[i,j] ← q;
            for all ft ∈ F\f {
                for all g ∈ ft.G {                    //g states in other subFSMs
                    if (indexOf(g) = indexOf(q))
                        sb[indexOf(ft),j] ← g;
                }
            }
            j ← j +1;
        }
    }
}
assignFreeStates(set of struct subFSM F, int sb)
{
    for all f ∈ F {
        int j ← 1;
        i ← indexOf(f);
        for all s ∈ f.S \f.Q {
            while (sb[i,j] ≠ null)
                j ← j +1;
            sb[i,j] ← s;
        }
    }
}
```

FIGURE 4. Pseudo code for bundling of the coupled (assignCoupledStates) and free states (assignFreeStates)

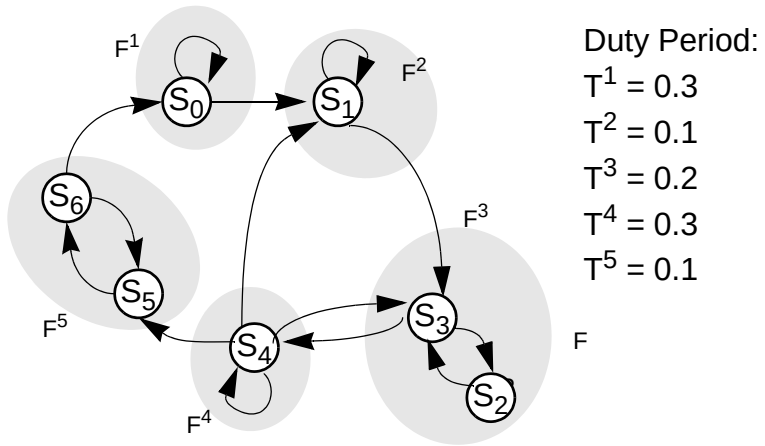

FIGURE 5. Example of a partitioned FSM with high *c*.

a) Initial coupled state table

B:	$\mathbf{b_0}$	$\mathbf{b_1}$	$\mathbf{b_2}$	$\mathbf{b_3}$	$\mathbf{b_4}$
F^1	s_0	g_1	-	-	-
F^2	-	s_1	g_3	-	-
F^3	-	-	s_3	g_4	-
F^4	-	g_1	g_3	s_4	g_5
F^5	g_0	-	-	-	s_5

b) Sorted table

B:	$\mathbf{b_0}$	$\mathbf{b_1}$	$\mathbf{b_2}$	$\mathbf{b_3}$	$\mathbf{b_4}$
F^1	s_0	g_1	-	-	-
F^4	-	g_1	g_3	s_4	g_5
F^3	-	-	s_3	g_4	-
F^2	-	s_1	g_3	-	-
F^5	g_0	-	-	-	s_5

c) After merging coupled-state

B:	$\mathbf{b_0}$	$\mathbf{b_1}$	$\mathbf{b_2}$	$\mathbf{b_3}$	$\mathbf{b_4}$
F^1	s_0	g_1	-	-	-
F^4	s_4	g_1	g_3	g_5	-
F^3	g_4	-	s_3	-	-
F^2	-	s_1	g_3	-	-
F^5	g_0	-	-	s_5	-

d) Final coupled state table

B:	$\mathbf{b_0}$	$\mathbf{b_1}$	$\mathbf{b_2}$	$\mathbf{b_3}$
F^1	s_0	g_1	-	-
F^4	s_4	g_1	g_3	g_5
F^3	g_4	-	s_3	-
F^2	-	s_1	g_3	-
F^5	g_0	-	-	s_5

FIGURE 6. Optimized state table

```
struct subFSM {
    set of int S, G, Q;
}

set of struct subFSM F;
int sb[n, max(⌈log2|U^m|⌉)];  //state bundle table
double probBundle[numberOf(F.G)]; //sum of static state probability of states in each state bundle
mergeCoupledStates(set of struct subFSM F, int sb, double probBundle)


    sort(sb);
     g_n ← numberOf(F.G);
    for (i ← 1; i < g_n; i ← i+1){
        max_gain ← 0;
        opt_b  ← 0;
        for (j ← i+1; j ≤ g_n; j← j+1){
            row  ← 1;
            while (sb[row, i]=null ||sb[row,j]=null)
                row  ← row+1;
            if (row=n){                    //column i and j can be merged
                gain ← probBundle [i]+probBundle [j];
                if (gain > max_gain){
                    max_gain ← gain;
                    opt_b ← j;
                }
            }
        }
        if (opt_b > 0){                    //find column obt_b can be merged into column i
            for (k ← 1; k ≤  n; k ← k+1){
                if (sb[k, i] = null)
                    sb[k, i] ← sb[k, obt_b];
            }
            “remove column opt_b in sb”;
            g_n ← g_n-1 ;
        }
    }
    sort(sb);
}
```

FIGURE 7. Pseudo code for g-state merging

B: **C:**	b_1 000	b_2 001	b_3 010	b_4 011	b_5 100	b_6 101	b_7 110	b_8 111	$\lceil \log_2 \lvert U^m \rvert \rceil$
F^1	s_1	g_4	s_2	s_3	-	-	-	-	2
F^2	s_6	s_4	s_5	g_7	-	-	-	-	2
F^3	g_1	-	-	s_7	-	-	-	-	2
F^4	g_1	s_8	g_5	s_9	s_{10}	s_{11}	s_{12}	s_{13}	3

FIGURE 8. State encoding in re-ordered state table

int old_sb[n, $max(\lceil \log_2 \lvert U^m \rvert \rceil)$]; //state bundle table before optimization

int new_sb[n, $max(\lceil \log_2 \lvert U^m \rvert \rceil)$]← null; //state bundle table after optimization

double b_matrix[numberOf(mergedCoupledState),numberOf(mergedCoupledState)];

optimiseCoupledStates(**int** old_sb, **double** b_matrix, **int** new_sb)

```
    int b[numberOf(mergedCoupledState)];        //state bundles
    struct sub_b;   //subset of state bundles
    for (i ← 1; i ≤ numberOf(mergedCoupledState); i ← i+1)
        b[i] ← the ith column of old_sb;
    lock(b[1]) ;
    for (i ← 1; i ≤ n; i ← i+1)
        new_sb[i, 1] ← b[1];
    for (i ← 1; i ≤ n; i ← i+1){
        sub_b ← ∅;
        for (j ← 1; j ≤ numberOf(mergedCoupledState); j ← j+1){
            if (old_sb[i, j] ≠ null)
                sub_b ← sub_b ∪ b[j];
        }
        b_n ← least state bits needed for sub_b in new_sb;
        for unlocked state bundle b[x]∈ sub_b{
            for each locked state bundle  b[y] in b
                "find b_matrix[x_i y_i] with maximal state bundle transition probability";
        }
        for (j ← 1; j ≤ 2^(b−n); j ← j+1)
            "find m is the column index of b[y_i] in new_sb,such that
             Hammingdistance(binaryCode(m),binaryCode(j)) is minimal";
        for (k ← 1; k ≤ n; k ← k+1)
            new_sb[k, j] ← b[x_i];
        lock(b[x_i]);
    }
}
```

FIGURE 9. Pseudo code for optimized coupled state encoding

a) Final coupled state table after optimization

B: **C:**	**b_0** **00**	**b_1** **01**	**b_3** **10**	**b_2** **11**
F^1	s_0	g_1	-	-
F^4	s_4	g_1	g_5	g_3
F^3	g_4	-	-	s_3
F^2	-	s_1	-	g_3
F^5	g_0	-	s_5	-

b) Final state table after free state optimization

B: **C:**	**b_0** **00**	**b_1** **01**	**b_3** **10**	**b_2** **11**	bits
F^1	s_0	g_1	-	-	1
F^4	s_4	g_1	g_5	g_3	2
F^3	g_4	s_2	-	s_3	2
F^2	-	s_1	-	g_3	2
F^5	g_0		s_5	s_6	2

c) State table before state encoding optimization

B:	**b_0** **000**	**b_1** **001**	**b_2** **010**	**b_3** **011**	**b_4** **100**	**bits**
F^1	s_0	g_1	-	-	-	1
F^2	-	s_1	g_3	-	-	2
F^3	s_2	-	s_3	g_4	-	2
F^4	-	g_1	g_3	s_4	g_5	3
F^5	g_0	s_6	-	-	s_5	3

FIGURE 10. Comparison of state bundle table before and after optimization

```
struct subFSM {
    set of int S, G, Q;
}

set of struct subFSM F;
int sb[n, max(⌈log2|U^m|⌉)];  //state bundle table before free states assignment
double s_matrix[numberOf(S),numberOf(S)];  //state transition probability matrix

optimizeFreeStates(set of struct subFSM F, int sb, double s_matrix)
{
    int b_n[n];     //minimun state code length in each subFSM
    int sb_backup[n, max(⌈log2|U^m|⌉)];
    sb_backup ← copy(sb);
    assignFreeStates(F,sb_backup);
    for (i ← 1; i ≤ n; i ← i+1)
        b_n[i] ← minimumLengthCode(sb_backup[i]);
    for all f ∈ F {
        i ← indexOf(f);
        A ← f.Q ∪f.G;                  //assigned states ,g states included
        D ← f.S \f.Q;                  //unassigned states
        do{
            count ← numberOf(D);   //unassigned state number
            if (count>0){
                for all a ∈ A {
                    for all d ∈ D
                        "find s_matrix[a_i,d_j] with highest state transition probability";
                }
                k ← 1;
                while (sb[i,k] ≠ a_i)
                    k ←  k+1;
                for (m ← 1; m ≤ 2^b_n[i]; m ← m+1){
                    if(sb[i, m] ≠ null)
                        "find position m_i with minimal Hammingdistance(binaryCode(m_i -1), binaryCode(k-1));"
                }
                sb[i, m_i] ← d_j;
                A← A ∪ d_j;
                D← D\d_j;
                count ← count-1;
            }
        }while (count>0)
    }
}
```

FIGURE 11. Pseudo code for free state encoding optimization

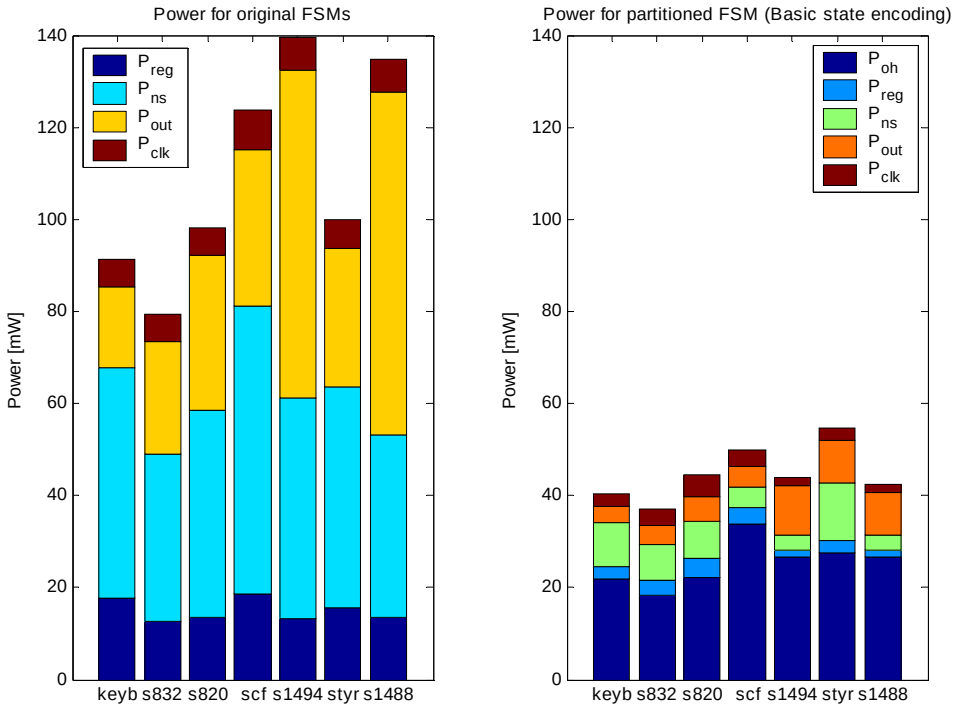

FIGURE 12. Power reductions for partitioned FSMs

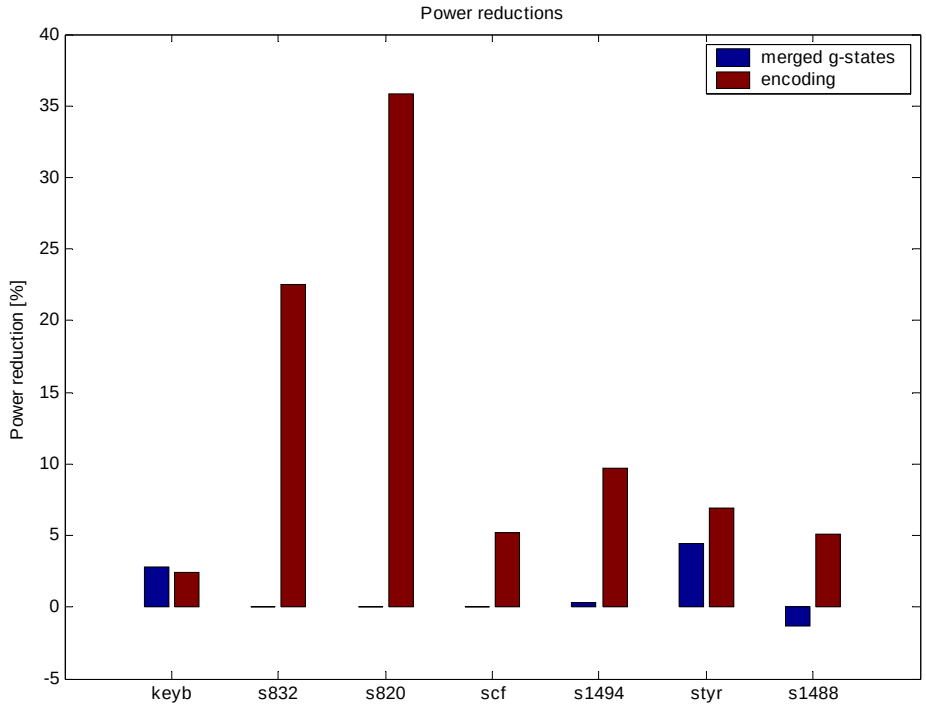

FIGURE 13. Power reductions in the sub-FSMs

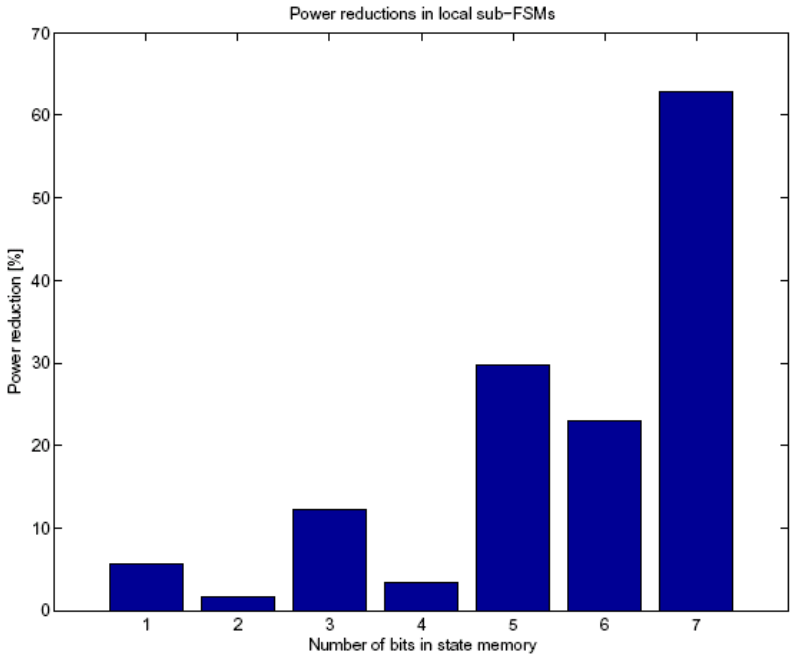

FIGURE 14. Power reductions versus number of bits in the state memory

TABLE 1. Structural information from the FSM decomposition

FSM	keyb	s832	s820	scf	s1494	styr	s1488
$\|S^1\|$	1	4	4	4	1	1	1
$\|U^1\|$	4	5	5	5	2	4	2
$\|PI^1\|$	3	6	6	1	3	5	3
$\|PO^1\|$	1	5	9	12	12	6	13
$\|T^1\|$	0.99	0.99	0.99	0.96	0.91	0.85	0.91
$\|S^2\|$	1	21	4	4	1	1	1
$\|U^2\|$	3	24	7	8	4	4	4
$\|PI^2\|$	6	18	9	3	3	5	3
$\|PO^2\|$	0	17	10	8	7	2	7
$\|T^2\|$	0.27	0.03	0.03	0.08	0.20	030	0.20
$\|S^3\|$	1		17	110	1	2	1
$\|U^3\|$	4		23	8	4	3	4
$\|PI^3\|$	7		17	3	6	6	6
$\|PO^3\|$	1		12	8	13	1	12
$\|T^3\|$	0.18		<0.01	0.02	0.08	0.20	0.08
$\|S^4\|$	1				1	4	1
$\|U^4\|$	4				2	8	3
$\|PI^4\|$	7				0	5	1
$\|PO^4\|$	1				5	5	4
$\|T^4\|$	0.09				0.02	0.08	0.02
$\|S^5\|$	15				1	8	1
$\|U^5\|$	16				3	16	2
$\|PI^5\|$	6				1	7	0
$\|PO^5\|$	2				4	10	3
$\|T^5\|$					0.03	0.03	0.03
$\|S^6\|$					1	14	42
$\|U^6\|$					3	21	46
$\|PI^6\|$					2	6	8
$\|PO^6\|$					7	10	19
$\|T^6\|$					0.02	<0.01	0.02
$\|S^7\|$					42		1
$\|U^7\|$					46		3
$\|PI^7\|$					8		2
$\|PO^7\|$					19		5
$\|T^7\|$					0.02		0.02