10.6 Temporally Varying Logic – Finite State Machines

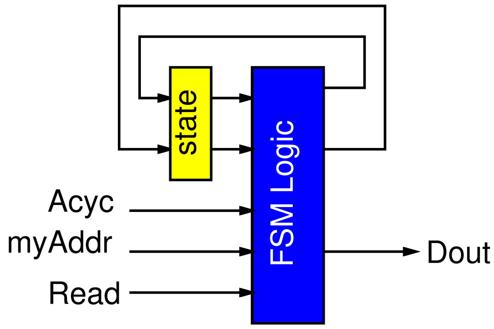
As we noted in Sections 10.3.2 and 10.3.3, the performance of a finite state machine is dictated by its latency rather than its throughput. Since the next state calculation must complete and be fed back to the input of the FSM before the next state behavior can begin, there is no benefit to be gained from spatial pipelining within the FSM logic. Temporal pipelining can be used within the FSM logic to increase gate and wire utilization as seen in Section 10.5.2. Finite state machines, however, happen to have additional structure over random logic which can be exploited. In particular, one never needs the full FSM logic at any point in time. During any cycle, the logic from only one state is active. In a traditional FPGA, we have to implement all of this logic at once in order to get the full FSM functionality. With multiple contexts, each context need only contain a portion of the state graph. When the machine transitions to a state whose logic resides in another context, we can switch contexts making a different portion of the FSM active. National Semiconductor, for example, exploits this feature in their multicontext programmable logic array (PLA), MAPL [Haw91].

10.6.1 Example

Figure 10.24 shows a simple, four-state FSM for illustrative purposes. The conventional, single-context implementation requires four 4-LUTs, one to implement each of `Dout` and `NS1` and two to calculate `NS0`. Figure 10.25 shows a two-context DPGA implementation of this same FSM. The design is partitioned into two separate circuits based on the original state variable `S1`. The two circuits are placed in separate contexts and `NS1` is used to select the circuit to execute as appropriate. Each circuit only requires three 4-LUTs, making the overall design smaller than the flat, single context implementation.

FSM Description

```
Idle  (00):
      if (Acyc & myAddr & Read)
          goto Wait1
      else
          goto Idle
Wait1 (01):
      goto Data
Data  (10):
      Assert Dout
      goto Wait2
Wait2 (11):
      goto Idle
```



FSM Logic

```
Dout = S1*/S0
NS0 =  /S1*/S0*Acyc*myAddr*Read + S1*/S0
NS1 = /S1*S0 + S1*/S0
```

Figure 10.24: Simple FSM Example

Context 0 `(S1=0)`

```
Dout  = 0
NS0  = /S0*Acyc*myAddr*Read
NS1  = S0
```

Context 1 `(S1=1)`

```
Dout  = 0
NS0  = /S0
NS1  = /S0
```

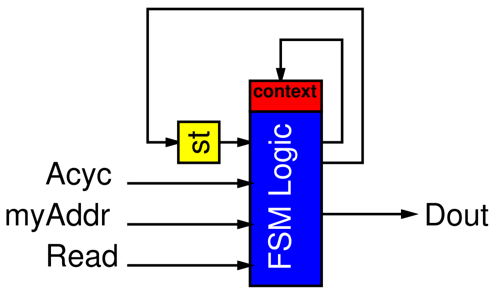


Figure 10.25: Two Context Implementation of Simple FSM Example

10.6.2 Full Temporal Partitioning

In the most extreme case, each FSM state is assigned its own context. The next state computation simply selects the appropriate next context in which to operate. Tables 10.17 and 10.18 show the reduction in area and path delay which results from state-per-context multiple context implementation of the MCNC FSM benchmarks. FSMs were mapped using `mustang` [DMNSV88]. Logic minimization and LUT mapping were done with `espresso`, `sis`, and `Chortle`. For single context FSM implementations, both one-hot and dense encodings were synthesized and the best mapping was selected. The multicontext FSM implementations use dense encodings so the state specification can directly serve as the context select. For multicontext implementations, delay and capacity are dictated by the logic required for the largest and slowest state. On average, the fully partitioned, multicontext implementation is 35-45% smaller that the single context implementation. Many FSMs are 3-5$\times$ smaller.

Timing From Tables 10.17 and 10.18, the multicontext FSM implementations generally have one or two fewer logic levels in their critical path than the single context implementations when mapped for minimum latency. The multicontext implementations have an even greater reduction in path length when mapped for minimum area. The multicontext FSMs, however, require additional time to distribute the context select and perform the multi-context read. *i.e.*

$$T_{single-context} = (\text{Levels}) \cdot t_{lut} \tag{10.6}$$
$$T_{multi-context} = (\text{Levels}) \cdot t_{lut} + t_{distribute} + t_{ctx-switch} \tag{10.7}$$

Recall $t_{ctx-switch} = 2.5$ ns from the prototype and $t_{lut} = 6$ ns with a typical amount of switching. Properly engineered, context distribution should take a few nanoseconds, which means the multicontext and single-context implementations run at comparable speeds when the multicontext implementation has one fewer LUT delays in its critical path than the single-context implementation.

10.6.3 Partial Temporal Partitioning

The capacity utilization and delay are often dictated by a few of the more complex states. It is often possible to reduce the number of contexts required without increasing the capacity required or increasing the delay. Tables 10.19 and 10.20 show the `cse` benchmark partitioned into various numbers of contexts and optimized for area or path delay, respectively. These partitions were obtained by partitioning along `mustang` assigned state bits starting with a four bit state encoding. Figures 10.26 and 10.27 plot the LUT count, area, and delay data from the tables versus the number of contexts employed.

One thing we note from both the introductory example (Figures 10.24 and 10.25) and the `cse` example is that the full state-per-context case is not always the most area efficient mapping. In the introductory example, once we had partitioned to two contexts, no further LUT reduction could be realized by going to four contexts. Consequently, the four context implementation would be larger than the two context implementation owing to the deeper context memories. In the `cse` example, the reduction in LUTs associated with going to going from 8 to 11 or 11 to 16 contexts saved less area than the cost of the additional context memories.

FSM	States	Single Context Levels	Single Context N_{4LUT}	Single Context **Area** [Mλ^2]	Context per State Levels	Context per State N_{4LUT}	Context per State **Area** [Mλ^2]	**Ratio** $\frac{A_{multi}}{A_{single}}$	**Delta Levels**
bbara	10	6	25	22.0	1	6	9.5	0.43	5
bbsse	16	4	50	43.9	3	12	24.6	0.56	1
bbtas	6	3	7	6.1	1	5	6.34	1.0	2
beecount	7	4	14	12.3	1	7	9.4	0.77	3
cse	16	6	83	72.9	2	15	30.7	0.42	4
dk14	7	4	58	50.9	1	8	10.8	0.21	3
dk15	4	12	25	22.0	1	7	7.8	0.35	11
dk16	27	5	80	70.2	1	8	23.2	0.33	4
dk17	8	6	19	16.7	1	6	8.5	0.51	5
dk512	15	2	20	17.6	1	7	13.8	0.79	1
donfile	24	2	46	40.4	1	6	16.0	0.40	1
ex1	20	7	120	105.4	2	26	61.4	0.58	5
ex4	14	7	21	18.4	1	13	24.6	1.33	6
ex6	8	5	57	50.0	1	11	15.7	0.31	4
keyb	19	7	112	98.3	4	14	32.0	0.32	3
mc	4	2	8	7.0	1	7	7.8	1.10	1
modulo12	12	6	12	10.5	1	5	8.7	0.82	5
planet	48	6	150	131.7	1	25	113.6	0.86	5
pma	24	6	82	72.0	2	15	40.1	0.56	4
s1	20	5	137	120.3	5	25	59.0	0.49	0
s1488	48	6	152	133.5	3	27	122.7	0.92	3
s1a	20	5	72	63.2	7	21	49.6	0.78	-2
s208	18	4	38	33.4	1	7	15.4	0.46	3
s27	6	2	5	4.4	1	4	5.1	1.20	1
s386	13	5	42	36.9	2	12	21.8	0.59	3
s420	18	3	40	35.1	1	7	15.4	0.44	2
s510	47	5	54	47.4	1	13	58.1	1.22	4
s8	5	4	12	10.5	1	4	4.7	0.45	3
s820	25	6	92	80.8	3	30	82.5	1.02	3
sand	32	7	178	156.3	5	30	98.9	0.63	2
sse	16	4	50	43.9	3	12	24.6	0.56	1
styr	30	7	186	163.3	4	21	65.9	0.40	3
tbk	32	8	340	298.5	6	33	108.8	0.36	2
Average								0.64	3

Table 10.17: Full Partitioning of MCNC FSM Benchmarks (Area Target)

FSM	States	Single Context Levels	Single Context N_{4LUT}	Single Context **Area** [Mλ^2]	Context per State Levels	Context per State N_{4LUT}	Context per State **Area** [Mλ^2]	**Ratio** $\frac{A_{multi}}{A_{single}}$	**Delta Levels**
bbara	10	3	40	35.1	1	6	9.5	0.27	2
bbsse	16	3	60	52.7	2	14	28.7	0.54	1
bbtas	6	2	9	7.9	1	5	6.3	0.80	1
beecount	7	2	19	16.7	1	7	9.4	0.57	1
cse	16	4	97	85.2	2	15	30.7	0.36	2
dk14	7	3	67	58.8	1	8	10.8	0.18	2
dk15	4	3	37	32.5	1	7	7.8	0.24	2
dk16	27	3	83	72.9	1	8	23.2	0.32	2
dk17	8	2	26	22.8	1	6	8.5	0.37	1
dk512	15	2	20	17.6	1	7	13.8	0.79	1
donfile	24	2	46	40.4	1	6	16.0	0.40	1
ex1	20	4	151	132.6	2	26	61.4	0.46	2
ex4	14	2	25	22.0	1	13	24.6	1.12	1
ex6	8	3	62	54.4	1	11	15.7	0.29	2
keyb	19	4	150	131.7	3	26	59.3	0.45	1
mc	4	2	8	7.0	1	7	7.8	1.10	1
modulo12	12	1	13	11.4	1	5	8.7	0.76	0
planet	48	4	172	151.0	1	25	113.6	0.75	3
pma	24	4	139	122.0	2	15	40.1	0.33	2
s1	20	4	195	171.2	3	30	70.8	0.41	1
s1488	48	4	183	160.7	2	28	127.2	0.79	2
s1a	20	3	107	93.9	4	30	70.8	0.75	-1
s208	18	3	40	35.1	1	7	15.4	0.44	2
s27	6	2	5	4.4	1	4	5.0	1.16	1
s386	13	4	54	47.4	2	12	21.8	0.46	2
s420	18	3	40	35.1	1	7	15.4	0.44	2
s510	47	3	76	66.7	1	13	58.1	0.87	2
s8	5	2	13	11.4	1	4	4.8	0.42	1
s820	25	3	137	120.3	3	30	82.5	0.69	0
sand	32	4	224	196.7	3	43	141.7	0.72	1
sse	16	3	60	52.7	2	14	28.7	0.54	1
styr	30	5	285	250.2	3	23	72.2	0.29	2
tbk	32	5	510	447.8	4	42	138.4	0.31	1
Average								0.56	1.36

Table 10.18: Full Partitioning of MCNC FSM Benchmarks (Delay Target)

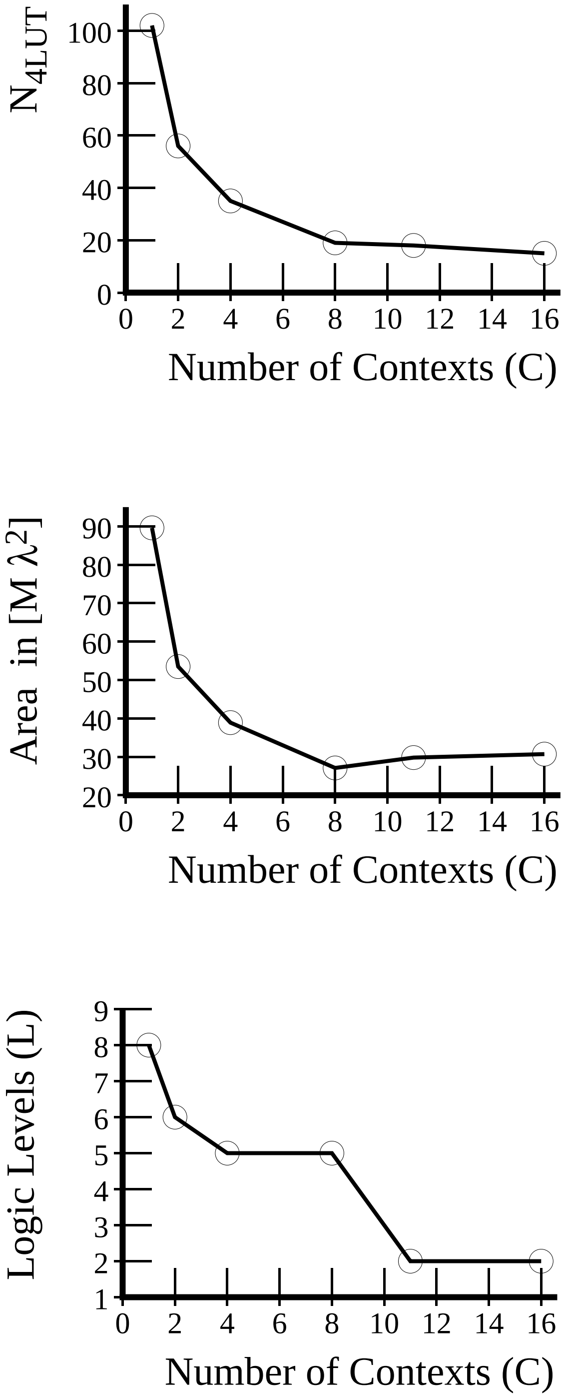


Figure 10.26: Area and Delay versus Number of Contexts for `cse` FSM Benchmark (Area Target)

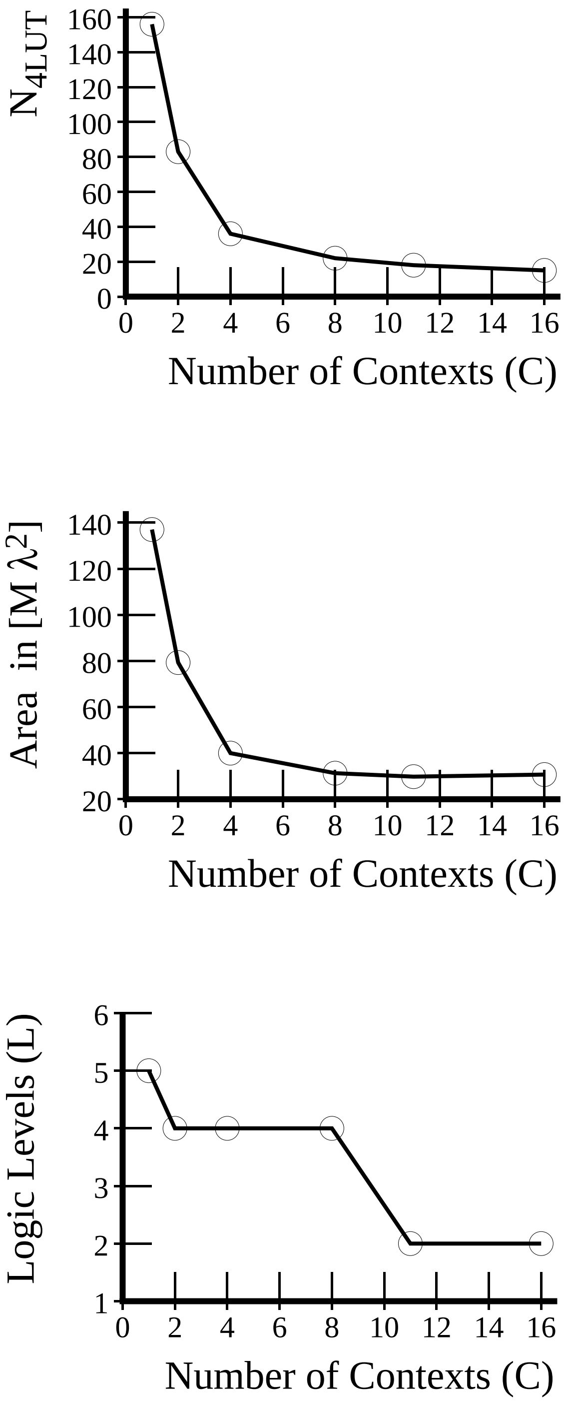


Figure 10.27: Area and Delay versus Number of Contexts for `cse` FSM Benchmark (Delay Target)

Multicontext Implementations for CSE FSM					
Contexts	Levels	N_{4LUT}	**Area**	$\frac{A_{mc}}{A_{single}}$	**Delta**
(one-hot)					
1	6	83	72.9	1.00	0
(dense)					
1	8	102	89.6	1.23	-2
2	6	56	53.5	0.73	0
4	5	35	38.9	0.53	1
8	5	19	27.1	0.37	1
11	2	18	29.8	0.41	4
16	2	15	30.7	0.42	4

Table 10.19: Area and Delay versus Number of Contexts for `cse` FSM Benchmark (Area Target)

Multicontext Implementations for CSE FSM					
Contexts	Levels	N_{4LUT}	**Area**	$\frac{A_{mc}}{A_{single}}$	**Delta**
(one-hot)					
1	4	97	85.2	1.00	0
(dense)					
1	5	156	137.0	1.60	-1
2	4	83	79.3	0.93	0
4	4	36	40.0	0.47	0
8	4	22	31.3	0.37	0
11	2	18	29.8	0.35	2
16	2	15	30.7	0.36	2

Table 10.20: Area and Delay versus Number of Contexts for `cse` FSM Benchmark (Delay Target)

Tables 10.21 through 10.25 show the benchmark set mapped to various multicontext implementations for minimum area. All partitioning is performed along `mustang` state bits. For these results, we examined all possible state bits along which to split and chose the best set. On average across the benchmark set, the 8-context mapping saves over 40% in area versus the best single-context case. The best multicontext mapping is often 3-5$\times$ smaller than the best single context mapping.

FSM	States	**Best Single Context**	**LUTs by Number of Context** Dense Encodings 1	2	4	8	16	32	64
bbara	10	25	25	19	12	8	6	6	6
bbsse	16	50	68	39	20	15	12	12	12
bbtas	6	7	7	5	5	5	5	5	5
beecount	7	14	14	11	7	7	7	7	7
cse	16	83	102	56	35	19	15	15	15
dk14	7	58	58	22	8	8	8	8	8
dk15	4	25	25	7	7	7	7	7	7
dk16	27	80	162	57	27	8	8	8	8
dk17	8	19	19	6	6	6	6	6	6
dk512	15	20	21	7	7	7	7	7	7
donfile	24	46	162	57	31	6	6	6	6
ex1	20	120	193	85	59	39	31	26	26
ex4	14	21	21	20	14	13	13	13	13
ex6	8	57	83	31	17	11	11	11	11
keyb	19	112	173	65	37	22	17	14	14
mc	4	8	8	7	7	7	7	7	7
modulo12	12	12	12	5	5	5	5	5	5
planet	48	150	346	122	80	38	29	27	25
pma	24	82	82	78	45	24	18	15	15
s1	20	137	196	144	57	44	28	25	25
s1488	48	152	305	153	129	52	34	28	27
s1a	20	72	136	73	51	38	24	21	21
s208	18	38	55	35	14	9	8	7	7
s27	6	5	5	5	4	4	4	4	4
s386	13	42	64	35	19	13	12	12	12
s420	18	40	54	33	15	10	8	7	7
s510	47	54	133	95	35	18	13	13	13
s8	5	12	12	13	9	4	4	4	4
s820	25	92	245	92	62	45	32	30	30
sand	32	178	358	139	95	44	33	30	30
sse	16	50	68	39	20	15	12	12	12
styr	30	186	387	133	67	40	28	21	21
tbk	32	340	513	137	75	48	35	33	33

Table 10.21: MCNC FSM Benchmarks LUTs v/s Number of Contexts (Area Target)

FSM	States	**Best Single Context**	**Area [Mλ^2] by Number of Context** Dense Encodings 1	2	4	8	16	32	64
bbara	10	21.9	21.9	18.2	13.3	11.4	12.3	19.8	34.8
bbsse	16	43.9	59.7	37.3	22.2	21.4	24.6	39.6	69.5
bbtas	6	6.1	6.1	4.8	5.6	7.1	10.2	16.5	29.0
beecount	7	12.3	12.3	10.5	7.8	10.0	14.3	23.1	40.5
cse	16	72.9	89.6	53.5	38.9	27.1	30.7	49.4	86.9
dk14	7	50.9	50.9	21.0	8.9	11.4	16.4	26.4	46.3
dk15	4	21.9	21.9	6.7	7.8	10.0	14.3	23.1	40.5
dk16	27	70.2	142.2	54.5	30.0	11.4	16.4	26.4	46.3
dk17	8	16.7	16.7	5.7	6.7	8.5	12.3	19.8	34.8
dk512	15	17.6	18.4	6.7	7.8	10.0	14.3	23.1	40.5
donfile	24	40.4	142.2	54.5	34.5	8.5	12.3	19.8	34.8
ex1	20	105.4	169.5	81.3	65.6	55.5	63.5	85.7	150.6
ex4	14	18.4	18.4	19.1	15.6	18.5	26.6	42.8	75.3
ex6	8	50.0	72.9	29.6	18.9	15.7	22.5	36.3	63.7
keyb	19	98.3	151.9	62.1	41.1	31.3	34.8	46.1	81.1
mc	4	7.0	7.0	6.7	7.8	10.0	14.3	23.1	40.5
modulo12	12	10.5	10.5	4.8	5.6	7.1	10.2	16.5	29.0
planet	48	131.7	303.8	116.6	89.0	54.1	59.4	89.0	144.8
pma	24	72.0	72.0	74.6	50.0	34.2	36.9	49.4	86.9
s1	20	120.3	172.1	137.7	63.4	62.7	57.3	82.4	144.8
s1488	48	133.5	267.8	146.3	143.4	74.0	69.6	92.3	156.4
s1a	20	63.2	119.4	69.8	56.7	54.1	49.2	69.2	121.6
s208	18	33.4	48.3	33.5	15.6	12.8	16.4	23.1	40.5
s27	6	4.4	4.4	4.8	4.4	5.7	8.2	13.2	23.2
s386	13	36.9	56.2	33.5	21.1	18.5	24.6	39.6	69.5
s420	18	35.1	47.4	31.5	16.7	14.2	16.4	23.1	40.5
s510	47	47.4	116.8	90.8	38.9	25.6	26.6	42.8	75.3
s8	5	10.5	10.5	12.4	10.0	5.7	8.2	13.2	23.2
s820	25	80.8	215.1	88.0	68.9	64.1	65.5	98.9	173.8
sand	32	156.3	314.3	132.9	105.6	62.7	67.6	98.9	173.8
sse	16	43.9	59.7	37.3	22.2	21.4	24.6	39.6	69.5
styr	30	163.3	339.8	127.1	74.5	57.0	57.3	69.2	121.6
tbk	32	298.5	450.4	131.0	83.4	68.4	71.7	108.8	191.1

Table 10.22: MCNC FSM Benchmarks Area v/s Number of Contexts (Area Target)

FSM	States	**Best Single Context**	**Delay by Number of Context** Dense Encodings 1	2	4	8	16	32	64
bbara	10	6	6	4	3	3	1	1	1
bbsse	16	4	6	6	4	3	3	3	3
bbtas	6	3	3	1	1	1	1	1	1
beecount	7	4	4	4	1	1	1	1	1
cse	16	6	8	6	5	5	2	2	2
dk14	7	4	4	4	1	1	1	1	1
dk15	4	12	12	1	1	1	1	1	1
dk16	27	5	4	5	8	1	1	1	1
dk17	8	6	6	1	1	1	1	1	1
dk512	15	2	10	1	1	1	1	1	1
donfile	24	2	4	6	5	1	1	1	1
ex1	20	7	4	7	7	5	4	2	2
ex4	14	7	7	3	2	1	1	1	1
ex6	8	5	8	6	4	1	1	1	1
keyb	19	7	4	7	5	4	5	4	4
mc	4	2	2	1	1	1	1	1	1
modulo12	12	6	6	1	1	1	1	1	1
planet	48	6	4	6	5	5	4	2	1
pma	24	6	6	8	5	5	3	2	2
s1	20	5	5	5	6	5	4	5	5
s1488	48	6	9	7	5	5	5	3	3
s1a	20	5	5	6	5	5	5	7	7
s208	18	4	4	6	3	3	2	1	1
s27	6	2	2	2	1	1	1	1	1
s386	13	5	5	5	4	3	2	2	2
s420	18	3	5	5	4	3	2	1	1
s510	47	5	6	5	3	3	1	1	1
s8	5	4	4	3	3	1	1	1	1
s820	25	6	5	7	7	6	4	3	3
sand	32	7	5	7	6	6	5	5	5
sse	16	4	6	6	4	3	3	3	3
styr	30	7	5	7	7	6	6	4	4
tbk	32	8	6	11	10	7	7	6	6

Table 10.23: MCNC FSM Benchmarks Delay v/s Number of Contexts (Area Target)

FSM	States	Best Single Context	Area Ratio by Number of Context Dense Encodings 1	2	4	8	16	32	64
bbara	10	1.00	1.00	0.83	0.61	0.52	0.56	0.90	1.58
bbsse	16	1.00	1.36	0.85	0.51	0.49	0.56	0.90	1.58
bbtas	6	1.00	1.00	0.78	0.90	1.16	1.67	2.68	4.71
beecount	7	1.00	1.00	0.86	0.63	0.81	1.17	1.88	3.30
cse	16	1.00	1.23	0.73	0.53	0.37	0.42	0.68	1.19
dk14	7	1.00	1.00	0.41	0.17	0.22	0.32	0.52	0.91
dk15	4	1.00	1.00	0.30	0.35	0.45	0.65	1.05	1.85
dk16	27	1.00	2.02	0.78	0.43	0.16	0.23	0.38	0.66
dk17	8	1.00	1.00	0.34	0.40	0.51	0.74	1.19	2.08
dk512	15	1.00	1.05	0.38	0.44	0.57	0.82	1.31	2.31
donfile	24	1.00	3.52	1.35	0.85	0.21	0.30	0.49	0.86
ex1	20	1.00	1.61	0.77	0.62	0.53	0.60	0.81	1.43
ex4	14	1.00	1.00	1.04	0.84	1.00	1.44	2.32	4.08
ex6	8	1.00	1.46	0.59	0.38	0.31	0.45	0.72	1.27
keyb	19	1.00	1.54	0.63	0.42	0.32	0.35	0.47	0.82
mc	4	1.00	1.00	0.95	1.11	1.42	2.04	3.28	5.77
modulo12	12	1.00	1.00	0.45	0.53	0.68	0.97	1.56	2.75
planet	48	1.00	2.31	0.89	0.68	0.41	0.45	0.68	1.10
pma	24	1.00	1.00	1.04	0.70	0.47	0.51	0.69	1.21
s1	20	1.00	1.43	1.14	0.53	0.52	0.48	0.69	1.20
s1488	48	1.00	2.01	1.10	1.07	0.55	0.52	0.69	1.17
s1a	20	1.00	1.89	1.10	0.90	0.86	0.78	1.09	1.92
s208	18	1.00	1.45	1.00	0.47	0.38	0.49	0.69	1.22
s27	6	1.00	1.00	1.09	1.01	1.30	1.87	3.00	5.28
s386	13	1.00	1.52	0.91	0.57	0.50	0.67	1.07	1.88
s420	18	1.00	1.35	0.90	0.47	0.41	0.47	0.66	1.15
s510	47	1.00	2.46	1.92	0.82	0.54	0.56	0.90	1.59
s8	5	1.00	1.00	1.18	0.95	0.54	0.78	1.25	2.20
s820	25	1.00	2.66	1.09	0.85	0.79	0.81	1.22	2.15
sand	32	1.00	2.01	0.85	0.68	0.40	0.43	0.63	1.11
sse	16	1.00	1.36	0.85	0.51	0.49	0.56	0.90	1.58
styr	30	1.00	2.08	0.78	0.46	0.35	0.35	0.42	0.74
tbk	32	1.00	1.51	0.44	0.28	0.23	0.24	0.36	0.64
average		1.00	1.51	0.86	0.63	0.56	0.70	1.09	1.92

Table 10.24: MCNC FSM Benchmarks Area Ratio v/s Number of Contexts (Area Target)

FSM	States	**Best Single Context**	**Delay Reduction by Number of Context** Dense Encodings 1	2	4	8	16	32	64
bbara	10	0	0	2	3	3	5	5	5
bbsse	16	0	-2	-2	0	1	1	1	1
bbtas	6	0	0	2	2	2	2	2	2
beecount	7	0	0	0	3	3	3	3	3
cse	16	0	-2	0	1	1	4	4	4
dk14	7	0	0	0	3	3	3	3	3
dk15	4	0	0	11	11	11	11	11	11
dk16	27	0	1	0	-3	4	4	4	4
dk17	8	0	0	5	5	5	5	5	5
dk512	15	0	-8	1	1	1	1	1	1
donfile	24	0	-2	-4	-3	1	1	1	1
ex1	20	0	3	0	0	2	3	5	5
ex4	14	0	0	4	5	6	6	6	6
ex6	8	0	-3	-1	1	4	4	4	4
keyb	19	0	3	0	2	3	2	3	3
mc	4	0	0	1	1	1	1	1	1
modulo12	12	0	0	5	5	5	5	5	5
planet	48	0	2	0	1	1	2	4	5
pma	24	0	0	-2	1	1	3	4	4
s1	20	0	0	0	-1	0	1	0	0
s1488	48	0	-3	-1	1	1	1	3	3
s1a	20	0	0	-1	0	0	0	-2	-2
s208	18	0	0	-2	1	1	2	3	3
s27	6	0	0	0	1	1	1	1	1
s386	13	0	0	0	1	2	3	3	3
s420	18	0	-2	-2	-1	0	1	2	2
s510	47	0	-1	0	2	2	4	4	4
s8	5	0	0	1	1	3	3	3	3
s820	25	0	1	-1	-1	0	2	3	3
sand	32	0	2	0	1	1	2	2	2
sse	16	0	-2	-2	0	1	1	1	1
styr	30	0	2	0	0	1	1	3	3
tbk	32	0	2	-3	-2	1	1	2	2
average		0.00	-0.27	0.33	1.27	2.18	2.70	3.03	3.06

Table 10.25: MCNC FSM Benchmarks Delta Delay v/s Number of Contexts (Area Target)

Tables 10.21 through 10.25 show the benchmark set mapped to various numbers of context implementations with delay minimization as the target. All partitioning is performed along `mustang` state bits. For these results, we examined all possible state bits along which to split and chose the best set. On average across the benchmark set, the 8 context mapping are half the area of the best single-context case while achieving comparable delay.

FSM	States	**Best Single Context**	**Delay by Number of Context** Dense Encodings 1	2	4	8	16	32	64
bbara	10	3	4	3	3	3	1	1	1
bbsse	16	3	4	4	3	3	2	2	2
bbtas	6	2	3	1	1	1	1	1	1
beecount	7	2	3	2	1	1	1	1	1
cse	16	4	5	4	4	4	2	2	2
dk14	7	3	4	3	1	1	1	1	1
dk15	4	3	5	1	1	1	1	1	1
dk16	27	3	4	4	5	1	1	1	1
dk17	8	2	4	1	1	1	1	1	1
dk512	15	2	5	1	1	1	1	1	1
donfile	24	2	4	5	5	1	1	1	1
ex1	20	4	4	4	4	3	3	2	2
ex4	14	2	3	3	2	1	1	1	1
ex6	8	3	4	4	3	1	1	1	1
keyb	19	4	4	6	5	4	4	3	3
mc	4	2	2	1	1	1	1	1	1
modulo12	12	1	3	1	1	1	1	1	1
planet	48	4	4	5	3	3	3	2	1
pma	24	4	4	5	4	3	3	2	2
s1	20	4	5	5	4	4	3	3	3
s1488	48	4	5	4	7	4	3	3	2
s1a	20	3	5	6	5	5	4	4	4
s208	18	3	4	5	3	2	2	1	1
s27	6	2	2	2	1	1	1	1	1
s386	13	4	4	4	3	2	2	2	2
s420	18	3	5	5	3	2	2	1	1
s510	47	3	4	4	3	2	1	1	1
s8	5	2	2	3	3	1	1	1	1
s820	25	3	5	4	4	4	3	3	3
sand	32	4	5	5	4	4	4	3	3
sse	16	3	4	4	3	3	2	2	2
styr	30	5	5	4	4	4	4	3	3
tbk	32	5	6	7	6	6	5	4	4

Table 10.26: MCNC FSM Benchmarks Delay v/s Number of Contexts (Delay Target)

FSM	States	**Best Single Context**	**LUTs by Number of Context** Dense Encodings 1	2	4	8	16	32	64
bbara	10	40	33	24	12	8	6	6	6
bbsse	16	60	85	53	24	15	14	14	14
bbtas	6	9	7	5	5	5	5	5	5
beecount	7	19	18	12	7	7	7	7	7
cse	16	97	156	83	36	22	15	15	15
dk14	7	67	58	26	8	8	8	8	8
dk15	4	37	38	7	7	7	7	7	7
dk16	27	83	162	82	35	8	8	8	8
dk17	8	26	31	6	6	6	6	6	6
dk512	15	20	52	7	7	7	7	7	7
donfile	24	46	162	199	31	6	6	6	6
ex1	20	151	193	136	80	47	32	26	26
ex4	14	25	28	20	14	13	13	13	13
ex6	8	62	97	42	18	11	11	11	11
keyb	19	150	173	202	37	22	22	26	26
mc	4	8	8	7	7	7	7	7	7
modulo12	12	13	21	5	5	5	5	5	5
planet	48	172	346	202	93	38	30	27	25
pma	24	139	97	148	67	33	18	15	15
s1	20	195	196	144	80	56	40	30	30
s1488	48	183	455	264	99	57	35	28	28
s1a	20	107	136	73	51	38	33	30	30
s208	18	40	55	64	14	11	8	7	7
s27	6	5	5	5	4	4	4	4	4
s386	13	54	84	46	20	15	12	12	12
s420	18	40	54	33	20	11	8	7	7
s510	47	76	185	97	35	20	13	13	13
s8	5	13	13	13	9	4	4	4	4
s820	25	137	245	154	90	60	39	30	30
sand	32	224	358	219	130	67	61	43	43
sse	16	60	85	53	24	15	14	14	14
styr	30	285	387	211	129	60	34	23	23
tbk	32	510	513	676	353	71	52	42	42

Table 10.27: MCNC FSM Benchmarks LUTs v/s Number of Contexts (Delay Target)

FSM	States	**Best Single Context**	**Area [Mλ^2] by Number of Context** Dense Encodings 1	2	4	8	16	32	64
bbara	10	35.1	29.0	22.9	13.3	11.4	12.3	19.8	34.8
bbsse	16	52.7	74.6	50.7	26.7	21.4	28.7	46.1	81.1
bbtas	6	7.9	6.1	4.8	5.6	7.1	10.2	16.5	29.0
beecount	7	16.7	15.8	11.5	7.8	10.0	14.3	23.1	40.5
cse	16	85.2	137.0	79.3	40.0	31.3	30.7	49.4	86.9
dk14	7	58.8	50.9	24.9	8.9	11.4	16.4	26.4	46.3
dk15	4	32.5	33.4	6.7	7.8	10.0	14.3	23.1	40.5
dk16	27	72.9	142.2	78.4	38.9	11.4	16.4	26.4	46.3
dk17	8	22.8	27.2	5.7	6.7	8.5	12.3	19.8	34.8
dk512	15	17.6	45.7	6.7	7.8	10.0	14.3	23.1	40.5
donfile	24	40.4	142.2	190.2	34.5	8.5	12.3	19.8	34.8
ex1	20	132.6	169.5	130.0	89.0	66.9	65.5	85.7	150.6
ex4	14	21.9	24.6	19.1	15.6	18.5	26.6	42.8	75.3
ex6	8	54.4	85.2	40.2	20.0	15.7	22.5	36.3	63.7
keyb	19	131.7	151.9	193.1	41.1	31.3	45.1	85.7	150.6
mc	4	7.0	7.0	6.7	7.8	10.0	14.3	23.1	40.5
modulo12	12	11.4	18.4	4.8	5.6	7.1	10.2	16.5	29.0
planet	48	151.0	303.8	193.1	103.4	54.1	61.4	89.0	144.8
pma	24	122.0	85.2	141.5	74.5	47.0	36.9	49.4	86.9
s1	20	171.2	172.1	137.7	89.0	79.7	81.9	98.9	173.8
s1488	48	160.7	399.5	252.4	110.1	81.2	71.7	92.3	162.2
s1a	20	93.9	119.4	69.8	56.7	54.1	67.6	98.9	173.8
s208	18	35.1	48.3	61.2	15.6	15.7	16.4	23.1	40.5
s27	6	4.4	4.4	4.8	4.4	5.7	8.2	13.2	23.2
s386	13	47.4	73.8	44.0	22.2	21.4	24.6	39.6	69.5
s420	18	35.1	47.4	31.5	22.2	15.7	16.4	23.1	40.5
s510	47	66.7	162.4	92.7	38.9	28.5	26.6	42.8	75.3
s8	5	11.4	11.4	12.4	10.0	5.7	8.2	13.2	23.2
s820	25	120.3	215.1	147.2	100.1	85.4	79.9	98.9	173.8
sand	32	196.7	314.3	209.4	144.6	95.4	124.9	141.7	249.1
sse	16	52.7	74.6	50.7	26.7	21.4	28.7	46.1	81.1
styr	30	250.2	339.8	201.7	143.4	85.4	69.6	75.8	133.2
tbk	32	447.8	450.4	646.3	392.5	101.1	106.5	138.4	243.3

Table 10.28: MCNC FSM Benchmarks Area v/s Number of Contexts (Time Target)

FSM	States	**Best Single Context**	**Delay Reduction by Number of Context** Dense Encodings 1	2	4	8	16	32	64
bbara	10	0	0	0	0	0	0	0	0
bbsse	16	0	-1	-1	0	0	1	1	1
bbtas	6	0	-1	1	1	1	1	1	1
beecount	7	0	-1	0	1	1	1	1	1
cse	16	0	-1	0	0	0	2	2	2
dk14	7	0	-1	0	2	2	2	2	2
dk15	4	0	-2	2	2	2	2	2	2
dk16	27	0	0	0	0	0	0	0	0
dk17	8	0	-2	1	1	1	1	1	1
dk512	15	0	-3	1	1	1	1	1	1
donfile	24	0	-2	-3	-3	1	1	1	1
ex1	20	0	0	0	0	1	1	2	2
ex4	14	0	-1	-1	0	1	1	1	1
ex6	8	0	-1	-1	0	2	2	2	2
keyb	19	0	0	-2	-1	0	0	1	1
mc	4	0	0	1	1	1	1	1	1
modulo12	12	0	-2	0	0	0	0	0	0
planet	48	0	0	-1	1	1	1	2	3
pma	24	0	0	-1	0	1	1	2	2
s1	20	0	-1	-1	0	0	1	1	1
s1488	48	0	-1	0	-3	0	1	1	2
s1a	20	0	-2	-3	-2	-2	-1	-1	-1
s208	18	0	-1	-2	0	1	1	2	2
s27	6	0	0	0	1	1	1	1	1
s386	13	0	0	0	1	2	2	2	2
s420	18	0	-2	-2	0	1	1	2	2
s510	47	0	-1	-1	0	1	2	2	2
s8	5	0	0	-1	-1	1	1	1	1
s820	25	0	-2	-1	-1	-1	0	0	0
sand	32	0	-1	-1	0	0	0	1	1
sse	16	0	-1	-1	0	0	1	1	1
styr	30	0	0	1	1	1	1	2	2
average		0.00	-0.91	-0.48	0.06	0.64	0.91	1.15	1.21

Table 10.29: MCNC FSM Benchmarks Delta Delay v/s Number of Contexts (Delay Target)

FSM	States	**Best Single Context**	**Area Ratio by Number of Context** Dense Encodings 1	2	4	8	16	32	64
bbara	10	1.00	0.83	0.65	0.38	0.32	0.35	0.56	0.99
bbsse	16	1.00	1.42	0.96	0.51	0.41	0.54	0.88	1.54
bbtas	6	1.00	0.78	0.60	0.70	0.90	1.30	2.09	3.66
beecount	7	1.00	0.95	0.69	0.47	0.60	0.86	1.38	2.43
cse	16	1.00	1.61	0.93	0.47	0.37	0.36	0.58	1.02
dk14	7	1.00	0.87	0.42	0.15	0.19	0.28	0.45	0.79
dk15	4	1.00	1.03	0.21	0.24	0.31	0.44	0.71	1.25
dk16	27	1.00	1.95	1.08	0.53	0.16	0.22	0.36	0.64
dk17	8	1.00	1.19	0.25	0.29	0.37	0.54	0.87	1.52
dk512	15	1.00	2.60	0.38	0.44	0.57	0.82	1.31	2.31
donfile	24	1.00	3.52	4.71	0.85	0.21	0.30	0.49	0.86
ex1	20	1.00	1.28	0.98	0.67	0.50	0.49	0.65	1.14
ex4	14	1.00	1.12	0.87	0.71	0.84	1.21	1.95	3.43
ex6	8	1.00	1.56	0.74	0.37	0.29	0.41	0.67	1.17
keyb	19	1.00	1.15	1.47	0.31	0.24	0.34	0.65	1.14
mc	4	1.00	1.00	0.95	1.11	1.42	2.04	3.28	5.77
modulo12	12	1.00	1.62	0.42	0.49	0.62	0.90	1.44	2.54
planet	48	1.00	2.01	1.28	0.68	0.36	0.41	0.59	0.96
pma	24	1.00	0.70	1.16	0.61	0.39	0.30	0.41	0.71
s1	20	1.00	1.01	0.80	0.52	0.47	0.48	0.58	1.01
s1488	48	1.00	2.49	1.57	0.69	0.51	0.45	0.57	1.01
s1a	20	1.00	1.27	0.74	0.60	0.58	0.72	1.05	1.85
s208	18	1.00	1.38	1.74	0.44	0.45	0.47	0.66	1.15
s27	6	1.00	1.00	1.09	1.01	1.30	1.87	3.00	5.28
s386	13	1.00	1.56	0.93	0.47	0.45	0.52	0.83	1.47
s420	18	1.00	1.35	0.90	0.63	0.45	0.47	0.66	1.15
s510	47	1.00	2.43	1.39	0.58	0.43	0.40	0.64	1.13
s8	5	1.00	1.00	1.09	0.88	0.50	0.72	1.16	2.03
s820	25	1.00	1.79	1.22	0.83	0.71	0.66	0.82	1.44
sand	32	1.00	1.60	1.06	0.74	0.49	0.64	0.72	1.27
sse	16	1.00	1.42	0.96	0.51	0.41	0.54	0.88	1.54
styr	30	1.00	1.36	0.81	0.57	0.34	0.28	0.30	0.53
tbk	32	1.00	1.01	1.44	0.88	0.23	0.24	0.31	0.54
average		1.00	1.45	1.05	0.59	0.50	0.62	0.95	1.67

Table 10.30: MCNC FSM Benchmarks Area Ratio v/s Number of Contexts (Delay Target)

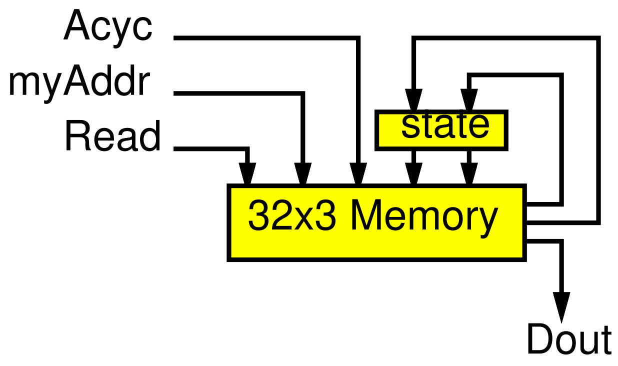


Figure 10.28: Memory-based Implementation for Simple FSM Example

10.6.4 Comparison with Memory-based FSM Implementations

A memory and a state register can also be used to implement finite-state machines. The data inputs and current state are packed together and used as addresses into the memory, and the memory outputs serve as machine outputs and next state outputs. Figure 10.28 shows a memory implementation of our simple FSM example from Figures 10.24 and 10.25.

Used for finite-state machines, the DPGA is a hybrid between a purely gate (FPGA) implementation and a purely memory implementation. The DPGA takes advantage of the memory to realize smaller, state-specific logic than an FPGA which must implement all logic simultaneously. The DPGA uses the restructurable interconnect in the array to implement next-state and output computations out of gates. As we noticed in Section 4.5, the gate implementation allow the DPGA to exploit regularities in the computational task. In this case, we avoid the necessarily exponential area increase associated with additional inputs in a memory implementation, the linear-log increase associated with additional states, and the linear increase associated with additional outputs.

Assuming we could build just the right sized memory for a given FSM, the area would be:

$$A_{memory-fsm} = \left(N_{state} \cdot 2^{N_{inputs}}\right) \times \left(N_{outputs} + \lceil \log_2 (N_{states}) \rceil\right) \times A_{mem\_cell}$$

Table 10.31 summarizes the areas of the best memory-based FSM implementations along with the areas for FPGA and 8-context DPGA implementations. The "Min area" column indicates the area assuming a memory of exactly the right size is used, while the "Memory Area" field shows the area for the smallest memory with an integral number of address bits as shown in the "organization" column. When the total number of state bits and input bits is less than 11, the optimal memory implementations can be much smaller than the FPGA or DPGA implementation. Above 11 input and state bits, the DPGA implementation is smaller. Since the DPGA implementation size increases with task complexity rather than number of inputs, while the memory implementation is exponential in the number of inputs and state bits, the disparity grows substantially as the number of inputs and state bits increase.

FSM	states	ins	outs	**Min area** [Mλ^2]	Integral Addr. & Data Organization	**Memory area** [Mλ^2]	**FPGA area** [Mλ^2]	**8-ctx DPGA area** [Mλ^2]
bbtas	6	2	2	0.1	$2^5 \times 5$	0.2	6.1	7.1
dk15	4	3	5	0.3	$2^5 \times 7$	0.3	21.9	10.0
dk17	8	2	3	0.2	$2^5 \times 6$	0.2	16.7	8.5
dk512	15	1	3	0.3	$2^5 \times 7$	0.3	17.6	10.0
mc	4	3	5	0.3	$2^5 \times 7$	0.3	7.0	10.0
modulo12	12	1	1	0.1	$2^5 \times 5$	0.2	10.5	7.1
beecount	7	3	4	0.5	$2^6 \times 7$	0.5	12.3	10.0
dk14	7	3	5	0.5	$2^6 \times 8$	0.6	50.9	11.4
dk16	27	2	3	1.0	$2^7 \times 8$	1.3	70.2	11.4
donfile	24	2	1	0.7	$2^7 \times 6$	0.9	40.4	8.5
s27	6	4	1	0.5	$2^7 \times 4$	0.6	4.4	5.7
s8	5	4	1	0.4	$2^7 \times 4$	0.6	10.5	5.7
bbara	10	4	2	1.2	$2^8 \times 6$	1.8	21.9	11.4
ex6	8	5	8	3.4	$2^8 \times 11$	3.4	50.0	15.7
ex4	14	6	9	14.0	$2^{10} \times 13$	16.0	18.4	18.5
bbsse	16	7	7	27.0	$2^{11} \times 11$	27.0	43.9	21.4
cse	16	7	7	27.0	$2^{11} \times 11$	27.0	72.9	27.1
tbk	32	6	3	19.7	$2^{11} \times 8$	19.7	298.5	68.4
sse	16	7	7	27.0	$2^{11} \times 11$	27.0	43.9	21.4
s386	13	7	7	22.9	$2^{11} \times 11$	27.0	36.9	18.5
keyb	19	7	2	20.4	$2^{12} \times 7$	34.4	98.3	31.3
planet	48	7	19	184.3	$2^{13} \times 25$	245.8	131.7	54.1
pma	24	8	8	95.8	$2^{13} \times 13$	127.8	72.0	34.2
s1	20	8	6	67.6	$2^{13} \times 11$	108.1	120.3	62.7
s1a	20	8	6	67.6	$2^{13} \times 11$	108.1	63.2	54.1
ex1	20	9	19	294.9	$2^{14} \times 24$	471.9	105.4	55.5
s1488	48	8	19	368.6	$2^{14} \times 25$	491.5	133.5	74.0
styr	30	9	10	276.5	$2^{14} \times 15$	294.9	163.3	57.0
s208	18	11	2	309.7	$2^{16} \times 7$	550.5	33.4	12.8
sand	32	11	9	1101.0	$2^{16} \times 14$	1101.0	156.3	62.7
s820	25	18	19	188743.7	$2^{23} \times 24$	241591.9	80.8	64.1
s420	18	19	2	79272.3	$2^{24} \times 7$	140928.6	35.1	14.2
s510	47	19	7	384408.9	$2^{25} \times 13$	523449.1	47.4	25.6

N.b. – benchmarks reordered by the sum of the number of inputs and densely encoded state bits

Table 10.31: Memory Implementations for MCNC FSM Benchmarks

10.6.5 Areas for Improvement

Timing In this section, we assumed that the context read occurred in series with execution within the target context and state. It is possible to overlap context reads with execution by using a more sophisticated FSM mapping model. On state transition, instead of reading a context with the target state logic, we read a context with all the logic for any state which may *follow* the target state. This can be viewed as speculatively fetching just the set of logic which may be needed by the time it has been read a cycle later. Using this scheme, we can reduce the time to the actual delay in the context rather than the context delay plus the read time. For heavily branching FSMs, the target logic will often have to include more state logic per context than with this style of mapping than it was with the simple division described here. As we see here, including more state logic increases the delay so it is not immediately obvious which case will generally have superior performance.

Partitioning For the partial temporal partitioning above, we partitioned strictly along `mustang` state bits. This is likely to give less than optimal partitions since `mustang`'s cost model is aimed at multi-level logic implementations. It assumes all logic must be available at once and is not trying to maximize the independence among state groups. A more sophisticated mapping would go back to the original state-transition graph and partition states explicitly to minimize the logic required in each partition. Informally, the goal would be to group states with similar logic together and separate states performing disparate logical functions.

10.6.6 General Technique

While demonstrated in the contexts of FSMs, the basic technique used here is fairly general. When we can predict which portions of a netlist, circuit, or computational task are needed at a given point in time, we can generate a more specialized design which only includes the required logic. The specialized design is often smaller and faster than the fully general design. With a multicontext component, we can use the contexts to hold many specialized variants of a design, selecting them as needed.

In the synthesis of computations, it is common to synthesize a controller along with datapaths or computational elements. The computations required are generally different from state to state. Traditional, spatial implementations must have hardware to handle all the computations or generalize a common datapath to the point where it can be used for all the computations. Multicontext devices can segregate different computational elements into different contexts and select them only as needed.

For example, in both dbC [GM93] and PRISM-II [AWG94] a general datapath capable of handling all of the computational subtasks in computation is synthesized alongside the controller. At any point in time, however, only a small portion of the functionality contained in the datapath is actually needed and enabled by the controller. The multicontext implementation would be smaller by folding the disparate logic into memory and reusing the same active logic and interconnect to implement them as needed.