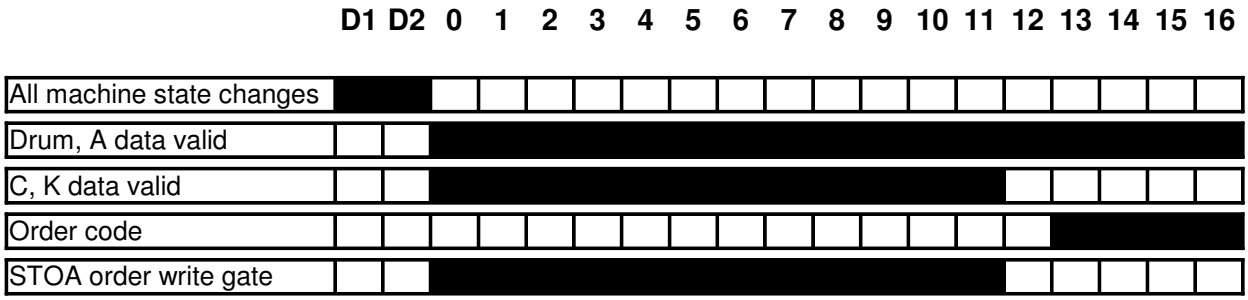
Opcode functions

				ALU op					ALU-A			ALU-B				SKIPS						GATES										
assembly	opcode	ALU func	ALU A, B in	OP-ADD	OP-OR	OP-AND	OP-CC	OP-CCY	OP-ALUA-A	OP-ALUA-AE	OP-ALUA-AI	OP-ALUB-A	OP-ALUB-D	OP-ALUB-K	OP-ALUB-SW	OP-SZ	OP-SNZ	OP-SC	OP-SNC	OP-STZ	OP-STNZ	OP-NOMREF	OP-ACIRC	OP-CCIRC	OP-KCIRC	OP-AWRITE	OP-CWRITE	OP-KWRITE	OP-HALTSW	STOA-GATE	STO-GATE	OP-TTYW	15 Dec 2001
IDLE	0			1																		1	1	1	1							Internal use
HALT	1			1					1													1	1	1	1			1				Machine control, TBD
SWA	2			1											1							1		1	1	1						Gate switches --> A
JZ	3	add	A, z	1					1							1						1	1		1		1					K --> C if Z
JNZ	4	add	A, z	1					1								1					1	1		1		1					K --> C if not Z
JC	5	add	A, z	1					1									1				1	1		1		1					K --> C if Cy
JNC	6	add	A, z	1					1										1			1	1		1		1					K --> C if not Cy
RLC	7	add	a, a	1			1		1			1										1		1	1	1							A + A + Cy --> A
SR	8	add	ae, k	1			1	1		1				1								1	1	1	1							AE + K --> A
COM	9	add	ai, k	1			1	1			1			1								1	1	1	1							1's com if K=0; 2's if K=1
JTZ	10	add	A, z	1					1											1		1	1		1		1					K --> C if TTI
JTNZ	11	add	A, z	1					1												1	1	1		1		1					K --> C if not TTI
TTO	12			1					1													1	1	1	1						1	Gates A to TTO FF
ADC k	13	add	a, k	1			1		1					1								1		1	1	1							A + k + cy --> A
ADD k	14	add	a, k	1			1	1	1					1								1		1	1	1							A + k --> A
AND k	15	and	a, k			1	1	1	1					1								1		1	1	1							A & k --> A
OR k	16	or	a, k		1		1	1	1					1								1		1	1	1							A . k --> A
LD k	17	add	z, k	1										1								1		1	1	1							K --> A, K not address
JUMP k	18	add	A, z	1					1													1	1		1		1					K --> C
ADCM k	19	and	a, d	1			1		1				1											1	1	1							A + (k) + Cy --> A
ADDM k	20	add	a, d	1			1	1	1				1											1	1	1							A + (k) --> A
ANDM k	21	and	a, d			1	1	1	1				1											1	1	1							A & (k) --> A
ORM k	22	or	a, d		1		1	1	1				1											1	1	1							A . (k) --> A
LDM k	23	add	z, d	1									1											1	1	1							(k) --> A
LDA k	24	add	z, k	1										1								1		1	1	1							K --> A, K is address
STOA k	25	add	A, z	1					1														1	1	1				1				D-write early pickoff
STO k	26	add	A, z	1					1														1	1	1					1			D-write early pickoff
mn population				###	2	2	###	8	20	1	1	1	5	8	1	1	1	1	1	1	1	###	###	###	###	###	7	0	1	1	1	1	

Word timing



Tom Jennings, 2350 Allesandro Street, Los Angeles CA 90039, USA

17

Drum geometry

Diameter	5.1	Major variable
Bit clock, Kbits/sec	100	Major variable
Bits/word	20	
Max memory	4096	Major variable
Sector skew	5	
Next-instruction, mS	1.00	

bpi	**bits per track**	**words per track**	**rpm**	**mS per track**	**worst case latency, mS**	**average latency, mS**	**tracks for max words**	bit spacing, mils
50	801	40	7490	8.01	8.0	4.0	102	0.0200
55	881	44	6809	8.81	8.8	4.4	93	0.0182
60	961	48	6241	9.61	9.6	4.8	85	0.0167
65	1041	52	5761	10.41	10.4	5.2	79	0.0154
70	1122	56	5350	11.22	11.2	5.6	73	0.0143
75	1202	60	4993	12.02	12.0	6.0	68	0.0133
80	1282	64	4681	12.82	12.8	6.4	64	0.0125
85	1362	68	4406	13.62	13.6	6.8	60	0.0118
90	1442	72	4161	14.42	14.4	7.2	57	0.0111
95	1522	76	3942	15.22	15.2	7.6	54	0.0105
100	1602	80	3745	16.02	16.0	8.0	51	0.0100
105	1682	84	3567	16.82	16.8	8.4	49	0.0095
110	1762	88	3404	17.62	17.6	8.8	46	0.0091
115	1843	92	3256	18.43	18.4	9.2	44	0.0087
120	1923	96	3121	19.23	19.2	9.6	43	0.0083
125	2003	100	2996	20.03	20.0	10.0	41	0.0080
130	2083	104	2881	20.83	20.8	10.4	39	0.0077
135	2163	108	2774	21.63	21.6	10.8	38	0.0074
140	2243	112	2675	22.43	22.4	11.2	37	0.0071
145	2323	116	2583	23.23	23.2	11.6	35	0.0069
150	2403	120	2497	24.03	24.0	12.0	34	0.0067
155	2483	124	2416	24.83	24.8	12.4	33	0.0065
160	2564	128	2341	25.64	25.6	12.8	32	0.0063

Minimum latency is word time * skew factor.

Control machine logic

Control machine logicexample assumes word width of 6 bits for simplicity (K is 3 bits). It is assumed there is D1-clock, D2-clock and other obvious control signals.	**CLOCK**	**Run**	**Exec**	**gate C to addr comp**	**gate K to addr comp**	**addr comp out 0=mis match**	**drum or reg data valid**	**write I gate**	**write K gate**	**write A gate**	**write C gate**	**write D gate**	**INCR-C**
	d1			1		1							
	d2			1		1							
FETCH-SEARCHis always address (C)	0			1		1	1						
line up drum head with address (C)	1			1		1	1						
	2			1		1	1						
	3			1		1	1						
(an addr mismatch would clear addr	4			1		1	1						
comp out)	5			1		1	1						
state change conditional address match	**d1**		1	-		1							
	d2		1	-		-							
FETCH-EXECUTE	0		1	1		-	1		1				1
load D into K then I	1		1	1		-	1		1				
incr C this state & bit0	2		1	1		-	1		1				
	3		1	1		-	1	1					
	4		1	1		-	1	1					
	5		1	1		-	1	1					
state change unconditional	**d1**	1			1	1							
(add'l state change in D2 if /MREF)	**d2**	1			1	1							
RUN-SEARCH is always address (K)	0	1			1	1	1						
line up drum head with address (K)	1	1			1	1	1						
	2	1			1	1	1						
	3	1			1	1	1						
	4	1			1	1	1						
	5	1			1	1	1						
state change unconditional	**d1**	1	1		1	1							
	d2	1	1		1	-							
RUN-EXECUTElda (k)	0	1	1		1	-	1			1			
clock D into A	1	1	1		1	-	1			1			
	2	1	1		1	-	1			1			
	3	1	1		1	-	1			1			
	4	1	1		1	-	1			1			
	5	1	1		1	-	1			1			
state change unconditional	**d1**				1	1							
	d2				1	1							
	d1	1	1		1	-							
sto, stoa selects Ae, asserts DWRITE	**d2**	1	1		1	-						1	
RUN-EXECUTEsto (k)	0	1	1		1	-	1					1	
	1	1	1		1	-	1					1	
	2	1	1		1	-	1					1	
	3	1	1		1	-	1					1	
	4	1	1		1	-	1					1	
	5	1	1		1	-	1					1	
unconditional state change	**d1**				1	1	1						

Control machine logic

	d2				**1**	**1**	**1**						
unconditional state change	**d1**		**1**	**1**		**1**							
	d2		**1**	**1**		**1**							
FETCH-EXECUTE sz	0		**1**	**1**		**1**	**1**						**1**
	1		**1**	**1**		**1**	**1**						
	2		**1**	**1**		**1**	**1**						
	3		**1**	**1**		**1**	**1**						
	4		**1**	**1**		**1**	**1**						
	5		**1**	**1**		**1**	**1**						
unconditional state change	**d1**	**1**			**1**	**1**							
MREF cond. state change skip **RUN-SEARCH**	**d2**	**1**	**1**		**1**	**1**							
RUN-EXECUTEsz	0	**1**	**1**		**1**	**1**	**1**						**1**
assume condition true	1	**1**	**1**		**1**	**1**	**1**						
	2	**1**	**1**		**1**	**1**	**1**						
	3	**1**	**1**		**1**	**1**	**1**						
	4	**1**	**1**		**1**	**1**	**1**						
	5	**1**	**1**		**1**	**1**	**1**						
unconditional state change	**d1**			**1**		**1**							
	d2			**1**		**1**							