Functional description for bank switching Agnus card for SMD2000

Rev C (final draft following completion of prototype); 9/6/2024.

Description and Features

An Agnus card for the SMD2000 using 8375 2 MB Agnus from A500+/A600.

At addresses 00 0000-0F FFFF (the first 1 MB of chip ram) there will be ordinary chip ram. Amiga will see this ram as usual.

At addresses 10 0000-1F FFFF (usually the second 1 MB of chip ram) there will be provision for up to 16x 1MB banks of ram, selectable using bank switching. The initial prototype only has two banks (bank 0 and bank 1) to demonstrate proof of concept.

Bank selection is controlled by writing to a new register BNKCTR at DFF1F0. It may be possible to write to this register using the Copper as well as the CPU however this has not been tested.

The status of the ram expansion may be determined by reading a new register BNKSTA at DFF1F4, and the configuration may be determined by reading a new register BNKCON at DFF1F6

At power on the second MB is not connected so the OS does not attempt to use it. The expansion can be configured to connect a bank in “reset proof” mode; this allows the OS to detect the second megabyte of chip ram at startup and use the full 2 MB in the conventional way.

Key Parts:

2 MB 8375 Fat Agnus.

ATF1504AS-10AU100 CPLD (“BUDDY”), mouser part 556-ATF1504AS-10AU100

2x HYB 514171BJ-60 chips for each 1 MB bank.

SN74F244 gated buffer ICs – 1/2 an IC for each 1 MB bank

SN74LS31D delay elements IC

CPLD Signals:

External

Signal	Name	Source
CCK, CCKQ	Color Clock / Quadrature	Amiga (Agnus) – Input to CPLD
CCKD, CCKQD	Delayed Clocks	Delay Elements IC – Input to CPLD
RGA[1:8]	Register Address Bus	Amiga (Agnus) – Input to CPLD
DRD[0:15]	DRAM Data Bus	Amiga (Motherboard) – I/O for CPLD.
_RST	Reset	Amiga (Motherboard) – Input to CPLD. Output LED0
BANKSEL	Bank Select Lines	CPLD output
LED 0-4	LED Lines	CPLD Output

Pin designations are specified in schematics and pin designation file.

Internal

Signal	Name	Description
PHI1, PHI2	Internal Clocks	Calculated from Amiga clocks and delayed clocks
LRGA[0:8]	Latched RGA	RGA bus latched on the positive edge of PHI1
LDRDI[8:15]	Latched DRD	Latched input register for upper half of DRD bus
LRDO[0:7]	Latched output register	Output register for lower half of DRD bus
BCR[0:15]	Bank Control Register	Stores state of the card (which bank is connected)
BANKNUM[0:15]	Bank Number	Internal register that controls BANKSEL output lines. These lines are also connected to LEDs.
DSBNK	Disconnect Bank	Active when instruction 0 is received. Output LED1.
CNBNK	Connect Bank	Active when instruction 2 is received. Output LED2.
RPBNK	Reset Proof Bank	Active when instruction 6 is received. Output LED3.
STATUS	Internal control line	Active when the STATUS register is read
CONFIG	Internal control line	Active when the CONFIG register is read

Constants

Constant	Value	Description
addBANKC	000011111	Address for Bank Control Register
addCONFIG	001011111	Address for Config Register
addSTATUS	011011111	Address for Status Register
configuration	10000001	Configuration value for card*

*this is covered in *Status and Config Registers BNKSTA and BNKCON* below.

Clocks

Input clocks from Agnus:

CCK (C1); 3.546895 MHz

CCKQ; CCK phase shifted by 90 degrees.

Calculated clocks (same as in PAULA):

PHI1 Leading edge 45 degrees after CCK (or 45 degrees before CCKQ). Falling edge coincident with CCK rising edge.

PHI2 Leading edge 45 degrees after CCKQ (or 45 degrees before falling edge of CCK). Falling edge coincident with rising edge of CCK.

_CCK Inverted CCK (not calculated in CPLD, ~CCK is used instead)

Clock Logic

A nominal delay of 33.5 ns +/- 4.5 ns will be added to both CCK and CCKQ using a SN74LS31 (Mouser part 595-SN74LS31D); use elements 1+3 and 6+4. This will create two clocks called CCKD and CCKQD (the D is for delay).

PHI1 = AND(CCKD,!CCKQ)

PHI2 = OR(CCKQD,_CCK)

This was tested in Excel then simulated in circuits.js.

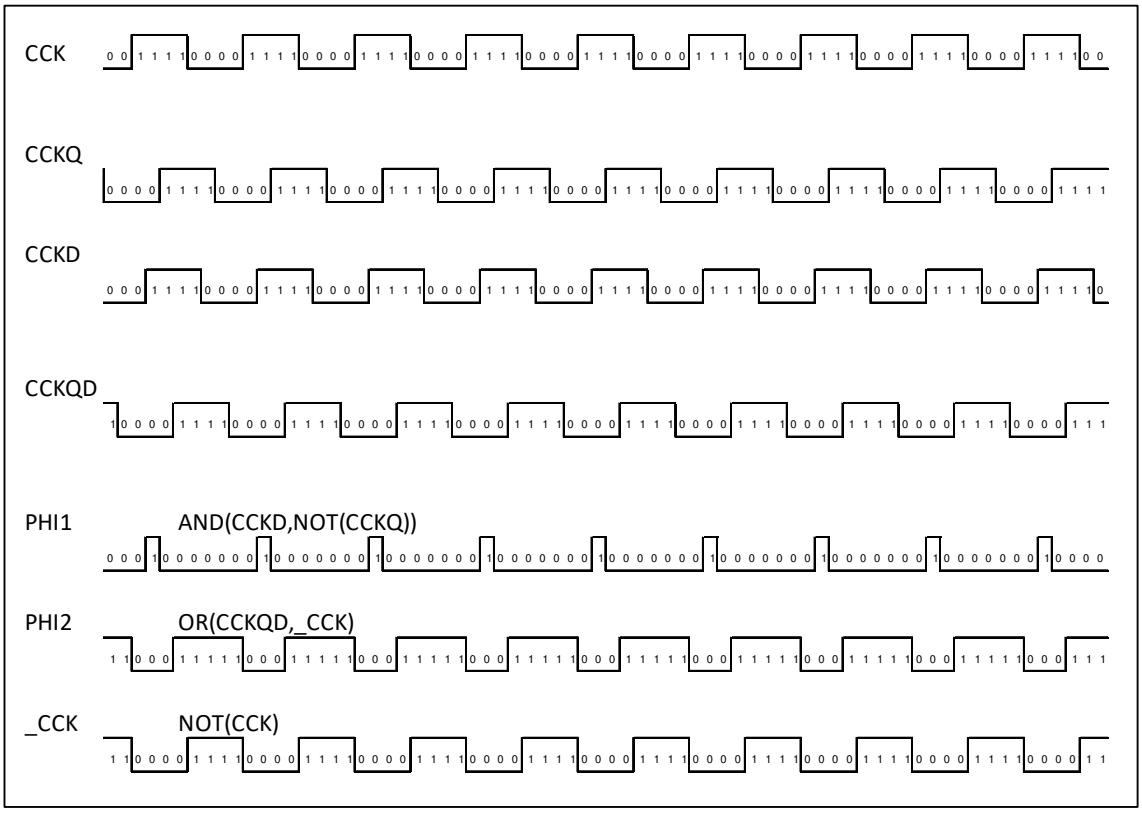


Figure 1: Clock Logic Simulation in Excel

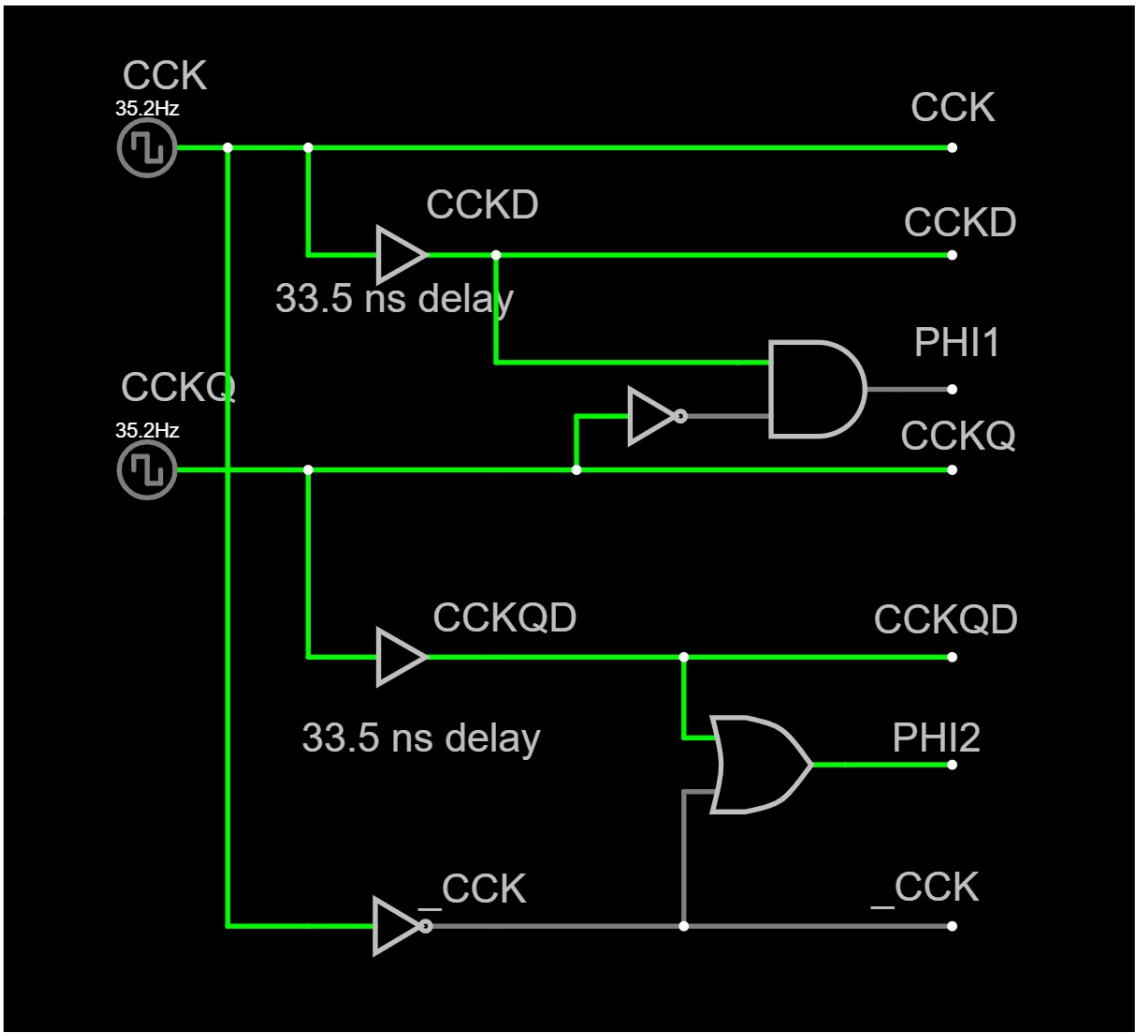


Figure 2: Simulation Circuit (frequency x10^7; time x10^-7)

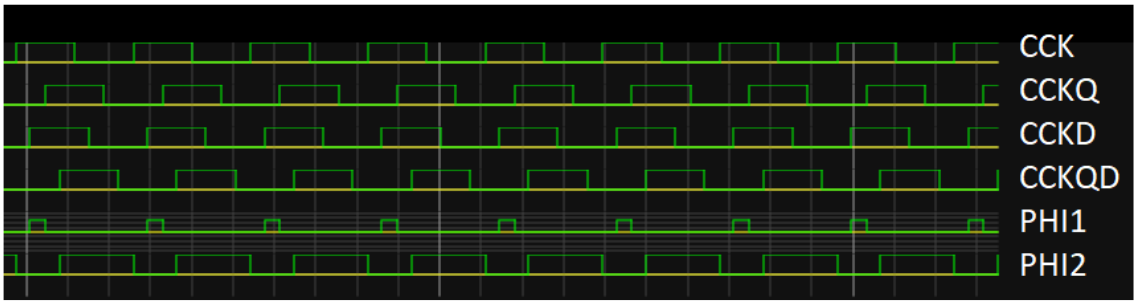


Figure 3: Output from simulation

New Amiga Registers

There shall be three new registers as follows:

Register Name	Short Name	R/W	Address	RGA [8..1] Binary	Function
Bank Control	BNKCTR	W	DFF1F0	11111000	Accepts instructions from CPU/Copper
Status	BNKSTA	R	DFF1F4	11111010	Read back current status
Config	BNKCON	R	DFF1F6	11111011	Read back configuration

Table 1: Registers added to the Amiga with the bank switching expansion

Card Input and Instruction Determination

DRD Bus Sampling

When the Amiga is not in reset, the high byte of the DRD bus (i.e. DRD[8:15]) is continuously sampled into LDRDI[8:15] on the negative edge of CCK.

When the Amiga is in reset the value of LDRD[8:15] is set to zero.

RGA Bus Sampling

If the Amiga is not in reset, the RGA bus shall be sampled onto LRGA[1:8] latches on the rising edge of PHI1.

If the Amiga is in reset, and the internal control line RPBNK is not asserted, then LRGA[1:8] is made to equal addBANKC.

If the Amiga is in reset, and the internal control line RPBNK is asserted, then LRGA[1:8] is made to equal zero.

Address decoding

If the value in LRGA[1:8] equals the constant addBANKC (i.e. the address of the Bank Control register) then the value in the bank control register BCR[8:15] is made to equal the value sampled in LDRDI[8:15]. In the prototype the value in LRDI[8:15] is also copied into BANKNUM[8:15] for display on the LEDs on those outputs.

If the value in LRGA[1:8] equals the constant addSTATUS then the internal control line STATUS is asserted. If it does not then STATUS is explicitly unasserted to

If the value in LRGA[1:8] equals the constant addCONFIG then the internal control line CONFIG is asserted. If it does not then CONFIG is explicitly unasserted.

Comment about Reset

When _RST is asserted (i.e. the Amiga is in reset) and RPBNK is not asserted, the combination of making LRGA[1:8]=addBANKC and LDRD=0 effectively issues instruction 0, which disconnects all banks.

When RPBNK is asserted the reset and RGA bus are effectively ignored.

Bank Control Register BNKCTR

Description

The Bank Control Register changes the status of the expansion; that is, it can disable the expansion, connect a RAM bank, or reset proof a RAM bank.

BNKCTR is an input from the point of view of the expansion, and write only from the point of view of the CPU, and is one byte wide. The byte is always the even byte at DFF1F0 (most significant/D8-D15). The odd byte DFF1F1 (D0-D7) is ignored.

BNKCTR is one byte wide and consists of two nibbles; an Instruction BCR[12:15] and the Bank Number BCR[8:11].

The instructions are predetermined:

Instruction	Function	Instruction	Function
0h = 0000b	Disconnect All Banks	8h = 1000b	
1h = 0001b	Reserved	9h = 1001b	Reserved
2h = 0010b	Connect Bank	Ah = 1010b	
3h = 0011b	Reserved	Bh = 1011b	Reserved
4h = 0100b		Ch = 1100b	
5h = 0101b	Reserved	Dh = 1101b	Reserved
6h = 0110b	Reset Proof Bank	Eh = 1110b	
7h = 0111b	Reserved	Fh = 1111b	Reserved

The Bank Number is simply a number up to Fh = 1111b which represents the 16 banks.

Instruction 0 disconnects all banks; this is also the initial and reset state. An unimplemented instruction also initiates instruction 0.

Instruction 2 connects the bank in the second nibble.

Instruction 6 connects and reset-proofs the bank in the second nibble.

Odd numbered instructions (i.e. those with the last bit set to 1) are reserved to allow 32 banks in a possible future expansion featuring 1MB Agnus.

Examples

To disconnect all banks write byte 00h or word 0000 DFF1F0. This also occurs automatically on reset (see *Comment about Reset* above).

To connect bank 1 write byte 21h or word 2100 to DFF1F0.

To reset proof bank 0 write 60h or word 6000 to DFF1F0.

Writing the register

The value stored in the BCR Latches changes when a byte is written to the BNKCTR register (this is covered in *Address decoding* above).

If BCR[12:15]=2 then the internal control line CNBNK is asserted, and RPBNK and DSBNK are unasserted.

If BCR[12:15]=6 then the internal control line RPBNK is asserted, and CNBNK and DSBNK are unasserted.

Instructions 2 and 6 both cause a line in BANKNUM[x] to become low, while all the other lines become high. N.B. in the prototype only lines BANKNUM[0:7] do this, however in future versions this should include all 16 lines.

If BCR[12:15]=0 or any other value then the internal control line DSBNK is asserted, and CNBNK and RPBNK are unasserted. In the prototype this causes BANKNUM[0:7]=z (i.e. all lines disconnected).

Bank Selection

Each BANKNUM line controls a BANKSEL output on the CPLD. In turn each one controls half of a 74F244, which connects the ram control lines _OE, _WE, _LCAS, and _UCAS, to the applicable ram bank. _RAS and the DRA (ram address bus) are connected all the time for refresh.

NB only BANKSEL0 and BANKSEL1 are connected in the prototype.

Status and Config Registers BNKSTA and BNKCON

The Status Register at DFF1F4 returns the current status of the expansion (the value in BCR[8:15]), and the Config register at DFF1F6 returns the number of banks installed, and the size of the banks installed.

Both are outputs from the point of view of the expansion, and read only from the point of view of the CPU, both are 8 bits wide and use the lower byte of the word (i.e. D[0:7] – the address of the bytes are actually DFF1F5 and DFF1F7). To read from either register read that memory location from the Amiga.

NB these two registers were originally intended to be one 16-bit wide register but had to be separated into two 8 bit registers due to size constraints on the CPLD.

Configuration Constant

The configuration constant is defined in the CPLD source code. There is provision for this to be set using jumpers on the card however this has not been implemented in the prototype.

Configuration constant is determined by:

Bits 7-3 are the number of banks minus one (i.e. 0 to 31), in binary. It's in reverse (i.e. least significant bit is bit 7). Bits 1&2 are not in use so should be set to 0. Bit 0 is 1 if the bank size is 1 MB and 0 if it is 512 kB; this is for a future version for 1 MB agnus. The prototype has two 1 MB banks so the configuration is 10000001

Bit number:	76543210
Constant:	10000001

Reading the register

When the PHI2 clock is changed (i.e. on both the positive and negative edge), the first instruction is to make LRDO[0:7]=z (i.e. the output register is disconnected), and WRITE is unasserted (WRITE was used during development to operate an LED). This ensures the expansion is never connected for more than one cycle of PHI2, which corresponds with the read cycle of the Amiga's CPU.

Then, if internal control line STATUS or CONFIG is asserted (this is covered in Address decoding), the applicable value is placed into LRDO[0:7] and therefore on the DRD. The internal control line WRITE is asserted.

STATUS/CONFIG are unasserted when the address on RGA[1:8] is changed, and when the PHI2 clock reaches its negative edge LRDO=z again.

RAM Interface

Background Notes

The Amiga has are six RAM control signals for the chip ram: 2x _RAS signals, 2x _CAS signals, R_W, and _OE.

When Agnus writes to RAM it asserts the R_W signal, then a _CAS signal, then a _RAS signal.

When Agnus reads from RAM it asserts the _OE signal, then a _CAS signal, then a _RAS signal.

The RAM is refreshed using only a _RAS signal.

The expansion needs to switch _OE, R_W, and the two _CAS lines for the switching bank, but _RAS needs to stay connected for refresh.

Circuit Description

The _OE, _WE, _CASL, and _CASU lines for each bank are connected to one half of a 74F244 three state buffer IC. The _G/_OE pin of each those buffers is controlled by the output pins BANKSEL[0:15] on the CPLD; when the BANKSEL line is made LOW it will enable the buffer output and therefore connect that bank of RAM. There is also an LED on each of the BANKSEL pins. NB in the prototype only RAM banks 0 and 1 are connected using a buffer.

_RAS remains connected at all times for refresh.

References

Agnus Specification Rev C; 20/7/1988

Denise Specification; 1988 (Amigawiki.de version)

Motorola MC68000 microprocessor user manual 9ed; 1993.

A500+ Service Manual; October 1991.

Amiga Hardware Reference Manual 3ed; 1991.

US Patent 4,777,621 (Amiga patent); 11/7/1988.

Paula Dissection, http://forum.6502.org/viewtopic.php?f=4&t=7681

HYB514171BJ-50 & -60 datasheet

ATF1508 and ATF1504 datasheets

Assorted TI 74 series logic datasheets

Appendix 1: Paula

So it's hard as fuck to find timing information about the registers in and out of Denise and Paula. I know the inputs and outputs are synced to clocks but I don't know which ones.

Some detail on how the registers and clocks work is available from the Paula Dissection but it's not perfect.

Edit for Rev C – the description below must be correct because the clock timing worked first try.

Paula I/O

The registers may be written or read by the CPU or Copper and are either read only or write only, not both.

Register addressing is on the RGA bus generated by Agnus. When the CPU is accessing the register it is a copy of CPU Bus A1-A8 (the rest of the decoding is done by GARY which asserts the _REGEN line).

In Paula the RGA bus is sampled into latches on the rising edge of clock Phi1, which is 45 degrees after the CCK rising edge (and 45 degrees before CCKQ) according to the phase diagram below. These are fed into an address decoder to determine which register is being used.

The DRD bus goes through some delay circuitry then is constantly sampled and saved in latches on the rising edge of #CCK. This is then put on Paula's internal data bus when Paula's _DBW is asserted. _DBW generated by the massive gate array that decodes the RGA address, and it looks like it just has a couple of inverting buffers.

If it is a read only register the internal data bus is put on a buffer, which drives the DRD bus when Paula's _DBR is asserted. It is unclear where _DBR comes from; it looks like it is an inverted _DRW NORed with a signal called DMARD that's synchronised to the rising edge of Phi2 (CCKQ rising edge plus 45 degrees) – in other words, DMARD is synchronised with the clock but it's blocked if _DBW is asserted. Tracing it back DMARD ultimately comes from STR which is also from the RGA address decoder but with some clock and inverter delays.

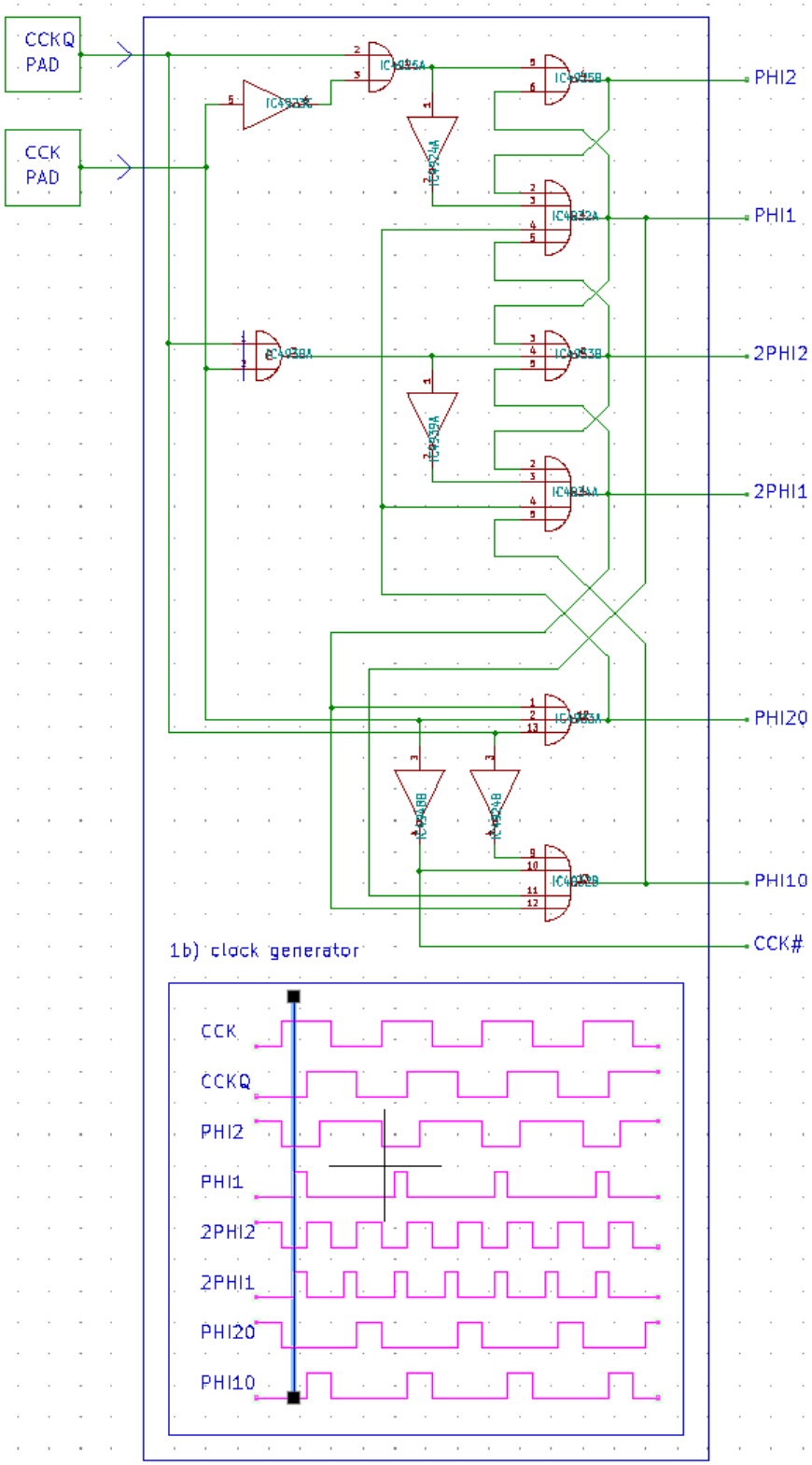
Paula Clocks

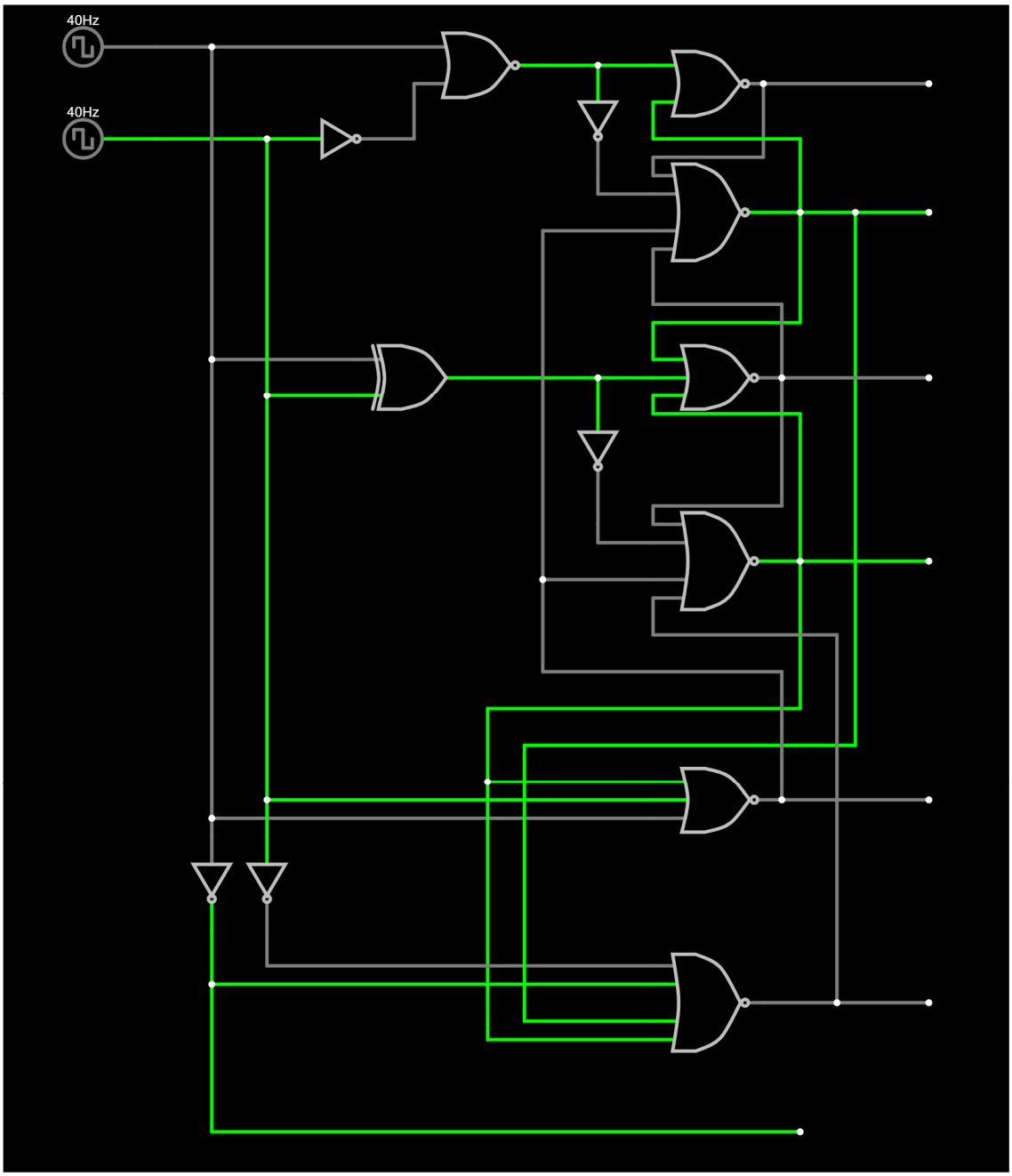
Two clock inputs from Agnus:

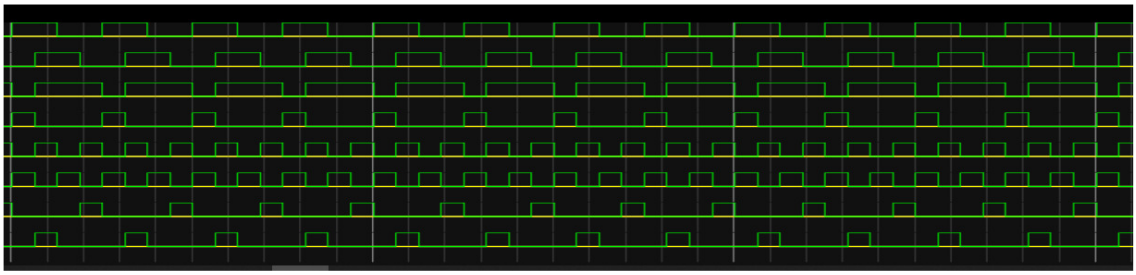
CCK (C1): 3.546895 MHz (1/8 of system clock)

CCKQ (C3): CCK phase shifted by 90 degrees

Calculates seven clocks, see phase diagram. There are propagation delays not modelled in circuitsjs simulation; some of the clocks are 45 degrees out of phase but this is not reproduced in simulation.


CCKQ
PAD
CCK
PAD
PHI2
PHI1
2PHI2
2PHI1
PHI20
PHI10
CCK#
1b) clock generator
CCK
CCKQ
PHI2
PHI1
2PHI2
2PHI1
PHI20
PHI10





Circuits.js output