MONROC
Monitor Read-Out and Control

Peter Staaf

MONROC
Monitor Read-Out and Control

1. Introduction

The ATLAS group at Uppsala University (Dept. of Radiation Sciences) is designing a control and monitoring chip (MONROC) to be used in the Silicon microstrip detector in the SCT at ATLAS. The MONROC has primarily been designed to support the digital architecture (AROW) but it can also be used with the binary or the analog architectures. This document will however focus on the digital implementation. The MONROC will perform three tasks:

1) Control the read-out of physics data and monitoring data (ROC)

2) Monitor certain parameters of the detector module (MOC)

3) Detect and execute control commands (INDEC)

Fig. 1. SemiConductor Tracker System (SCT)

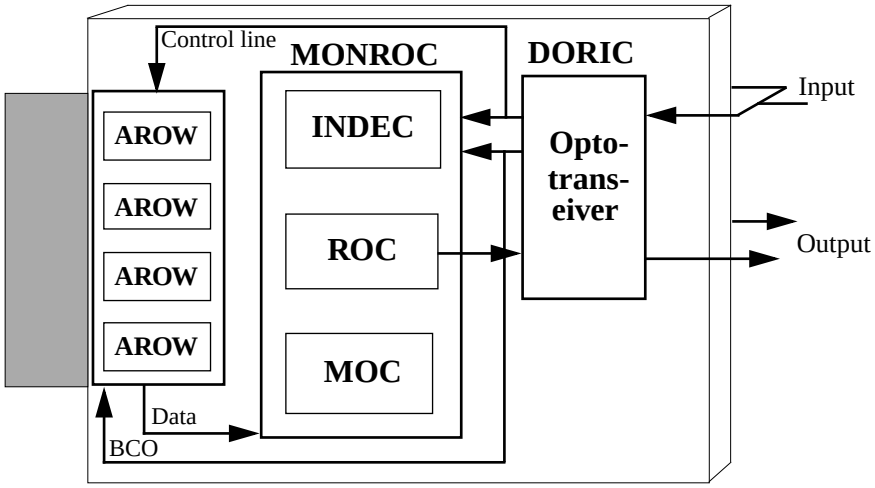


Figure 1 shows one side of a detector module. There is an equivalent circuit on the other side of the module.It is intended that if a DORIC or MONROC of one side cease to function, physics data will be redirected to the other side and read out through that MONROC / DORIC. It will also be possible to bypass a faulty AROW-chip in the read-out.Each AROW-chip plus the two MONROC-chips have a hard-wired identity number set by bond wires to **six digital** inputs on each chip. In this way it is possible to address individual chips with ”slow commands”(page 6).

In the DORIC the Bunch Cross Over signal (BCO 40 MHz) is extracted and separated from the control data and both are distributed to the entire system. Note that there is one incoming optical link but two outgoing optical links – one from each side of the module. As the control line is distributed to all AROW-chips as well as to the two MONROC-chips, there must be an input decoder in each of these chips. There is also the suggestion to bypass this control line to the adjacent module and therefore we have as many as six bits in the address identity.

Fig. 2. Overview of the MONROC

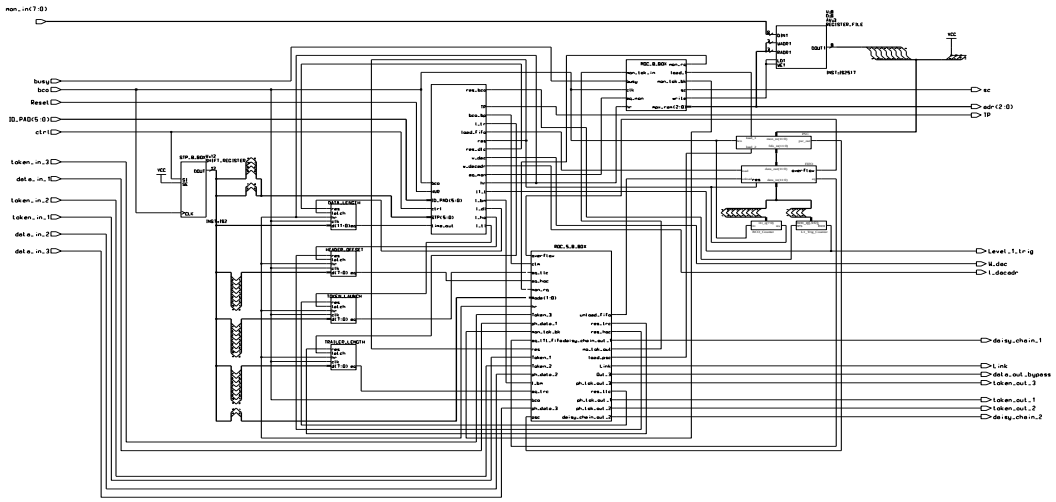


Figure 2 shows a simplified view of the MONROC system. The inputdecoder is the centered rectangle with no lable. The other two main machines are the readout controller unit (ROC_B_BOX) and the monitor controller (MOC_B_BOX). The four Word Compare Latches (WCL) DATA_LENGTH, HEADER_OFFSET, TRIGGER_LAUNCH (misspelled in diagram above) and TRAILER_LENGTH stands for the progammable timing performanse. Inside ROC_B_BOX (not indicated in diagram) is another type of WCL which latches in the data word which determines which bypass mode is in operate.All signals are shown and the main data paths should be easy to follow. Note that the PSC (Parallel to Serial Converter) has two parallel input selected by two LOAD-signals, one comming from the ROC_B_BOX and one from the MOC. These are exclusive as the ROC has a higher priority than the MOC.

2. Timing and data format

The specifications for the timing are preliminary **7ns** for both set-up and hold time with respect of the rising edge of BCO. All timings in MONROC are designed to meet this requirements on its inputs and outputs.Some latches like in the WCL:s will however be trigged on the falling edge of BCO internally. This will cause no problem as long as this clock is somewhat symmetrical.

All incoming data and tokens included the data/token in bypass mode (sending data on to opposite side AROW) will be shifted through a one bit shift register to increase safety in synchronization.

All data words in this document are written with the most significant bit (MSB) leftmost and is the first bit sent in a serial transfer. The only exception from this is the 12 bit long **data length word** which by collaboration convention is written and sent on the control wire with the least significant bit (LSB) leftmost.

3. INDEC Input Decoder

The INDEC is the main module in MONROC with an external asynchronous power-up reset input (Reset pin). In the reset state INDEC will by default send out the BCO on its output (Link). All registers in the WCL will be set to ones and the FIFO will be reset to zeros. The ROC and MOC will be synchroniosly resetted to their initiale state within three clockcycles after the input to Reset goes low

Every module will have an optical link for control and trigger data (L1-Trigger). At an L1-Trigger the INDEC will queue the event as two numbers (BCO# 8 bits and L1-Trigger# 4 bits) in a FIFO (32 bytes deep) that will subsequently be read by the ROC.As there are two MONROC chips on every module with up to four AROW chips each, a six bit Chip Address CA (Table 2 on page 6, field 4) is needed to be compared to the six static inputs (ID) bonded on each chip. The INDEC will also recognize ID ”111111” as an omnibus identity call. The only command that shouldbe recognized by both at least one AROW **and** the INDEC is the ID_GEN. We have however descided to seaparate this into two commands; one sent to the AROW and immediately after the same command is sent with the MONROC identity.. After the command ID_generation has been sent, the addressed AROW will send this data instead of the usual physics data sequence on the next received token.This means that the effect of this command on IN_DEC is that it must queue a ”Level-1 Trigger” which in turn will generate a read-out of the AROW from the ROC.

There are two groups of control commands: fast commands that can be issued during data taking and slow commands that are primarily used during start-up and initialization of the modules.

Table 1. Commands for the MONROC

Code			**Command**	
Field 1	**Field 2**	**Field 3**	**Abbrev.**	**Description**
110	—	—	L1_Trigger	Level one trigger
101	0101*	—	Soft Reset	Soft reset – resets BCO counter, L1_Ttrigger counter andFIFO. ROC and MOC returns to initalizaton state
101	0100	-	Soft Reset MONROC	Resets only the BCO counter!
101	0010	-	Send TP	INDEC will generate a 25 ns high CMOS pulse
101	0111†	< Data Length (12 bits)> <Address Field(6 bits)> <Command field (6 bits)> < DataWord (arbitrary bits)>		

* FC = Fast Command

† SC = Slow Command

Fast commands are not chip specific (no address field). Every input decoder must recognize them. There exist additional fast commands that are not listed here as they do not concern the MONROC. For slow commands there are, in addition to the address field, a data length field with the purpose of telling the IN_DEC how long a command to expect. If the command is addressed to any other address the IN_DEC will go to a wait state until an internal counter has reached the specified data length. As described before, the exception is the command ID-generation. **Note that exeptionally the data length is written with LSB first.**

In field 2 the code for the command is given.Some slow commands require a parameter given in field 6, while others do without.

Table 2. Slow commands for the MONROC

Code Field 1	Field 2	Field 3	Field 4 (CA)	Field 5	Field 6 (data)	Description
101	0111	0011 0000 0000	aaaaaa	000 100	-	Hard Reset
101	0111	0011 0000 0000	aaaaaa*	000 110	-	ID-Generation
101	0111	0011 0000 0000	aaaaaa	111 000	-	Clock Through Mode Set
101	0111	0011 0000 0000	aaaaaa	111 001	-	Clock Through Mode Reset
101	0111	0000 1000 0000	aaaaaa	100 100	dddd	Bypass Mode
101	0111	0011 0000 0000	aaaaaa	101 000		Monitor request
101	0111	0010 1000 0000	aaaaaa	100 001	dddd dddd	t_launch†
101	0111	0010 1000 0000	aaaaaa	100 010	dddd dddd	header offset‡
101	0111	0010 1000 0000	aaaaaa	100 011	dddd dddd	trailer length**..
101	0111	0010 1000 0000	aaaaaa	110 000-> 110 111	dddd dddd	DAC-data

* Must be recognized by the MONROC if MSB of CA equals MSB of ID

† Trigger LaunchTime between L1_Trigger and start of physics data readout

‡ Compensation for the number of AROW-chips cascaded in the token-ring

** Length of the n trailer indicating ”End of data”.

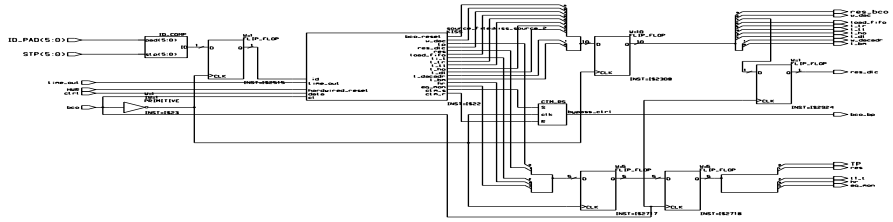
The idea is that the INDEC will latch in the parameters/word from the 12-bit long input shift register (STP=Serial To Parallel) into separate registers most of which in turn will be connected to comparators and resettable counters (WCL= Word Compare Latch). In this way these parameters will in practice be time constants and used for timing purposes. The INDEC makes use of the same principle as it latches in the 12-bit data in **field 3** (Data Length) as a parameter to an internal WCL and starts immediately the connected counter. When the comparator triggers, the INDEC will latch the data (if it is suppose to be any) in **field 6** into the correct register. If however, the control command do not concern INDEC, it will remain idle until the same trigger will indicate that the control data flow is ended.

Other parameters like the ”Bypass Mode” parameter/word register will not be connected to a WCL but rather be used as inputs to the internal multiplexors in the ROC.

4. DAC Configuration

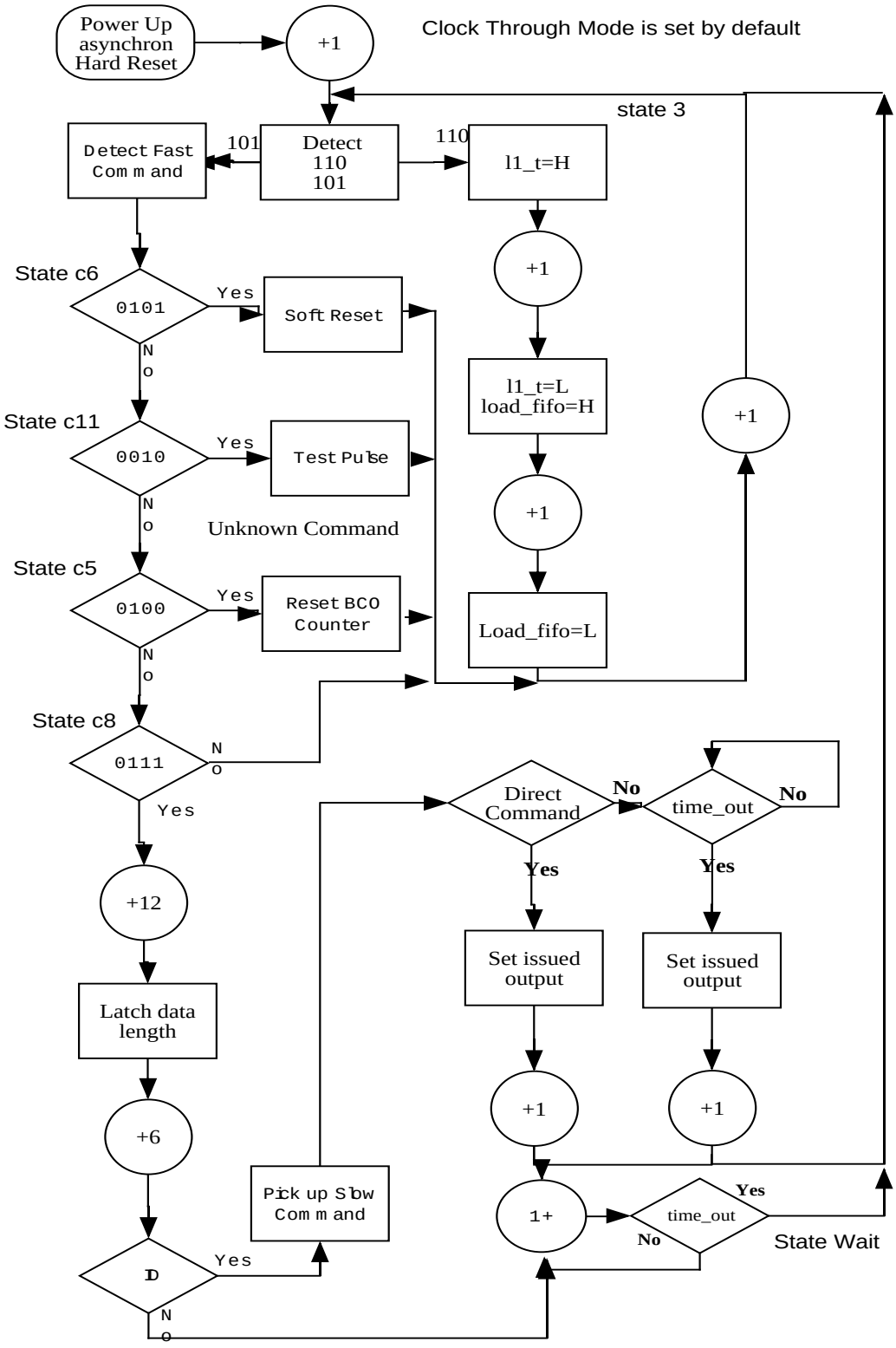
In this paper the pinout (see section 7) has an option for an external DAC with an external shift-register and a latch. The principle is that the address is set using the three LSB of **field 5** and the data/parameter is taken from **field 6**. This requires only the two control signals Write_dac and l_dacadr for the possible DAC. The signal of Write_dac is a 150 ns low CMOS and the l_dacadr (which shall latch in the three bit dac address and the eight bit data into an external latch) is a 25 ns high CMOS.

Fig. 3. IN_DEC.VHDL (KISS)



ID_PAD(5:0)	Bonding pads for chip-identity
STP(5:0)	Input bus from serial to parallell converter
time_out	Comparator input from DATA_LENGTH wcl
HWR	Hard Wired Reset (Active HIGH)
ctrl	Control line input from optical tranceiver
bco	BCO 40MHz clock input from optical tranceiver
res_bco	Reset signal to bco-counter
w_dac	Write signal for DAC data
load_fifo	Read signal to FIFO
l_tr	Latch Trailer_Length (Number of sequential zeros used for EOD indication)
l_tl	Latch Trigger_Launch (delay before data can be read out)
l_ho	Latch Header_Offset (Compensation for the number of AROW chips cascaded)
l_dl	Latch Data_Length (WLC)
w_dacadr	Write signal for DAC address and data into latch (renamed l_dacadr for 'lach in ...')
l_bm	Latch Bypass_Mode (Configures AROW cascading)
res_dlc	RESet Data Length Counter
bco_bp	Control signal to bypass BCO via data link during callibration.
TP	Test Pulse (cmos 25 ns high) for possible callibration of front-end electronics
res	Specific reset-signal for counters and registers in INDEC
l1_t	Level 1_Trigger -signal to L1_Trigger counter and level_1_trig output pin
hr	Hard Reset signal
eq_mon	Trigger signal for MOC to start a monitor (conversion) sequence

Fig. 4. Flow chart for INDEC



After a L1_Trigger, the INDEC has to be idle for two clock cycles before it starts to listen for another command. If the

5. ROC Read-Out Controller

This module will detect three conditions in the following priority:

1. Are there data in the FIFO (eq_fifo = LOW)?
2. Is the ALARM flag set due to FIFO overflow (Not a vital function if the FIFO is sufficiently deep)
3. Are there monitor data waiting (mon_req=HIGH).

A block diagram of the ROC is given in fig 5

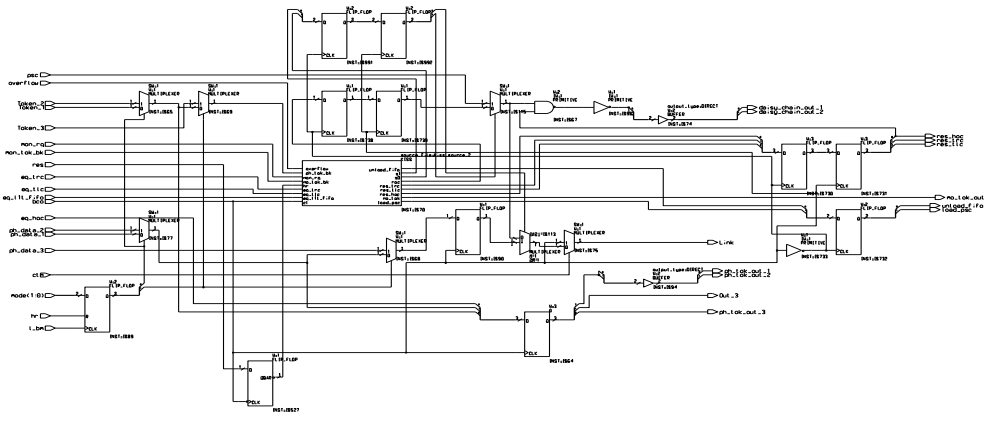
Table 3. Data format

Preamble	Data type	Header	Data	Trailer (suggested)
11101	0*	BCO# L1_T#	see Requirements for Wilkinson ADC…	1 0000 0000 0000
11101	1†	none	11 data (8bitar)11 11 data … 11 11 data 11	1 0000 0000 0000
11101	1	none	00 (ALARM FiFo overFlow)	1 0000 0000 0000

* 0= physics data

† 1= monitor data

Fig. 5. .Block diagram for the ROC Read-Out Controller



0.0.1 Inputs

psc	Serial input from Parallel to serial Converter
overflow	Overflow signal from level 1 trigger fifo
token_2	Token input; 2:nd
token_1	Token input; 1:st
Token_3	Token input from opposite side AROW
mon_rq	Monitor request signal input

mon_tok_bk	Monitor token return input
res	Soft Reset input for state machine
eq_trc	Comparator signal from TRailer_length Counter (trc) WCL
eq_tlc	Comparator signal from Trigger_Launch Counter (tlc) WCL
eq_l1t_fifo	Pointer signal from FiFo (goes low if there are data waiting)
bco	Clock signal, BCO 40 MHz
eq_hoc	Comparator signal from Header Offset Counter (hoc) WCL.
ph_data_2	Dasiy -chain return signal from AROW; 2:nd
ph_data_1	Daisy-chain return signal from AROW; 1:st
ph_data_3	Daisy-chain return signal from opposite side AROW
ctm	Clock Through Modet latch-in signal from input decoder
mode(1:0)	Input bus from STP
hr	Hard Reset (same effect as Power Up)
l_bm	Latch Bypass Mode; latch in the data word to control which inputs to listen to

0.0.2 Outputs

daisy_chain_out_1	Preamble/header output to 1:st AROW
daisy_chain_out_2	Preamble/header output to 2:nd AROW
mo_tok_out	Monitor token output
unload_fifo	Unload oldest content of level 1trigger fifo
load_psc	Load signal for psc (Parallel to Serial Converter)
Link	Data output to optical linc DORIC/ABC
ph_tok_out_1	Physics data token output to 1:st AROW
ph_tok_out_2	Physics data token output to 2:nd AROW
Out_3	Daisy chain bypass (connected to opposite side 1:st AROW)
ph_tok_out_3	Physics data token output bypass

Fig. 6. Signal paths for physics data

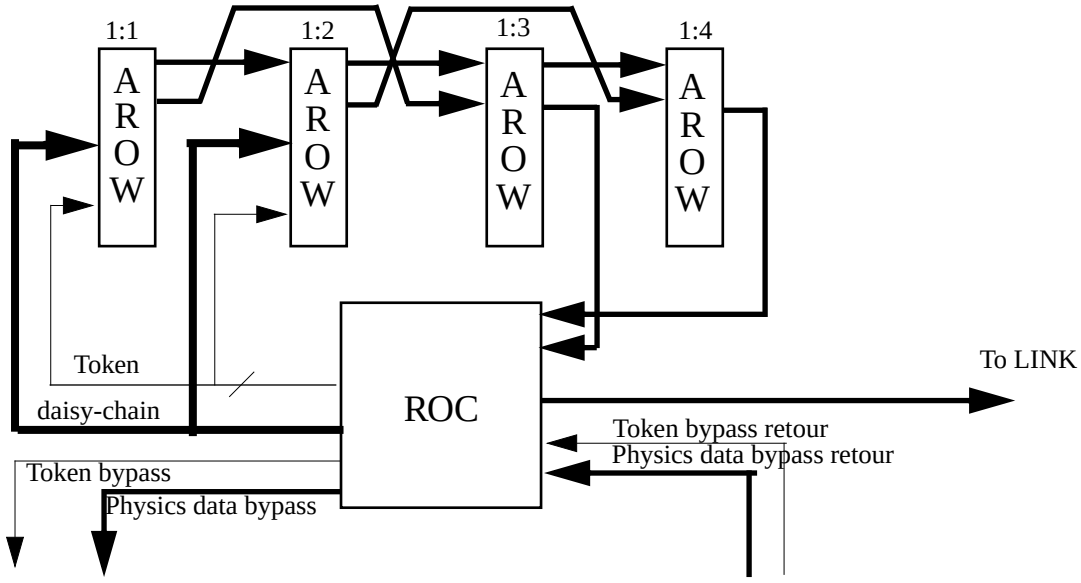
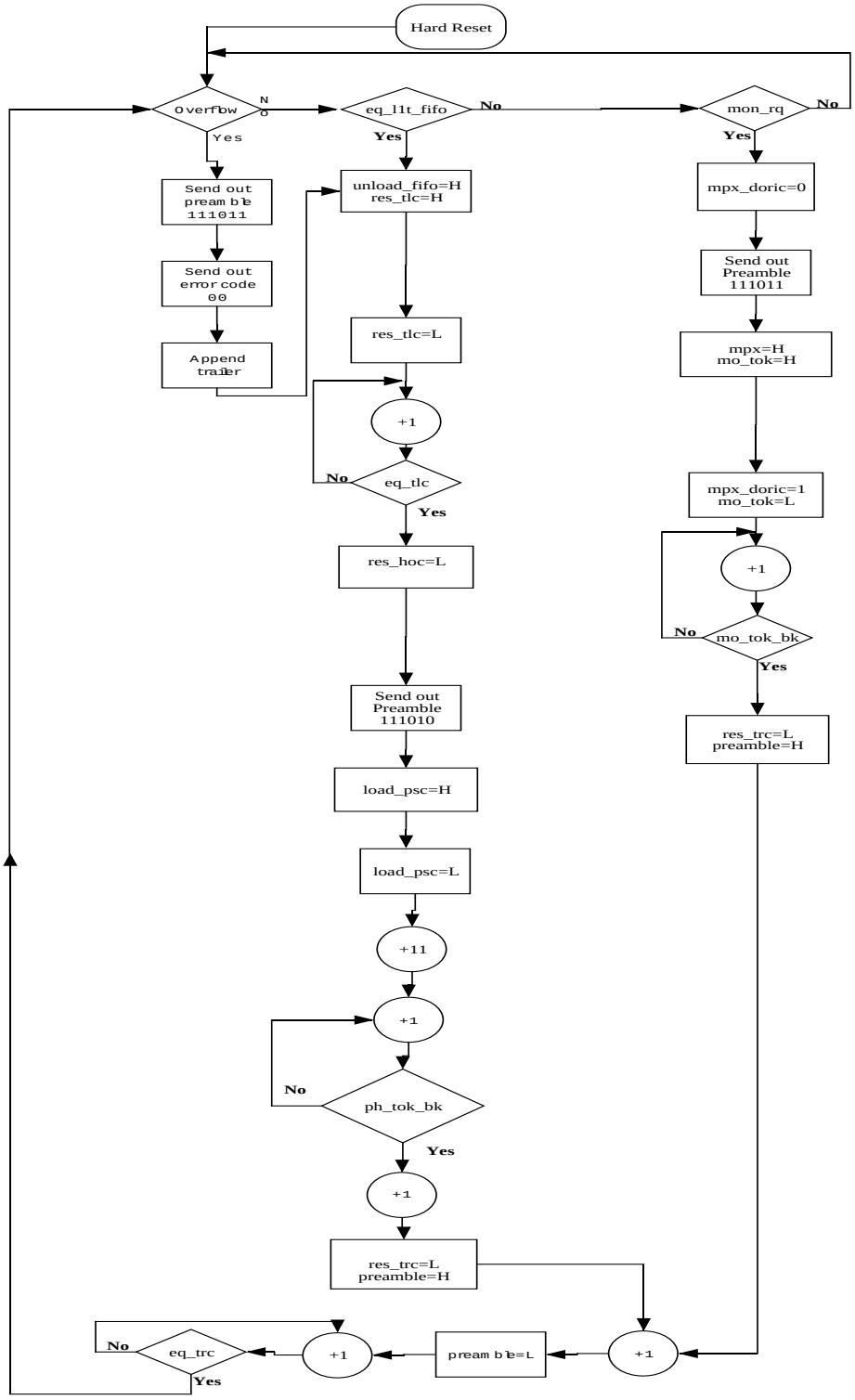


Figure 6 shows how physics data are linked through the cascade of AROW's and that each AROW chip has two inputs and two outputs for TOKEN (Shown as a bus. More clearly in Figure 8) and physics data so that it is possible to bypass any non-functional AROW chip without breaking the link. In every AROW the physics data and the TOKEN plus the pramble and header will pass through a one-bit shift register. Beacuse of set-up procedures in AROW, TOKEN has to be sent a number of clock cycles (programmable in WCL "hoc") before the end of preamble/header to make sure that physics data, if any, will be shifted out correctly.When an AROW holding physics data receives a TOKEN it will after three clock cycles connect the 1-bit shift register input to the internal data buffer. Three clock cycles before the last data bit is sent out, the token will ba passed on to the next chip. If that chip does not contain any data, the TOKEN will just be passed on the next clock cycle. Upon receiving a TOKEN, The ROC will wait appropriate time and then append a trailer consisting of an "one" and a number of zeros (programmable in WCL "trc")

What is not shown above is how the first AROW(1:1) can receive physics data and TOKEN from the opposite side-MONROC and how the last AROW(1:4) has its second physics data output and TOKEN output connected to the opposite side MONROC.

Notice that all data and token ouputs from ROC are allways active. This means that the connected AROW:s must be configurated to only listen to the correct input and ignore actions on the other ones. The ROC inputs however will be selected by the mode setting. Default setting by power-up reset is to select data_in_1 and token_in_1

Fig. 7. Flow chart for the ROC



After Power Up the ROC will enter a loop where it scans the overflow-signal,the eq_l1t_fifo-signal and the mon_rq-signal. After a L1_Trigger the FIFO is loaded with the current BCO- and L1_Trigger count by INDEC. This will make the read and write pointer non equal and the eq_fifo goes LOW. Upon detecting this the ROC will reset the tlc(Trigger Lag Counter)) and wait for comparator signal eq_toc. This time delay is necessary for the data sparsification circuit within the front-end electronics. At the comparator signal ROC will start sending out the preamble/header and at the same time it will reset the hoc (Header Offset Counter), the eq_hoc of which will send out the token in correct timing with the preamble/header .

The physics data preamble is made up of six bits (111010) after which ROC will shift out the contents of the oldest word in the FIFO as a header (12 bits).The procedure above will repeat itself until the eq_fifo is HIGH again (all values are read out) if no fifo_overflow occures.

On a monitor request (mon_rq=HIGH), ROC will send out a monitor data preamble (111011) directly to DORIC and a monitor token (mon_tok) to MOC initialize an read-out sequence of monitored data while it self goes into idle mode waiting for the rturn of token. On the last monitor data bit, the monitor controller will return a TOKEN BACK and ROC will be able to append the TRAILER.

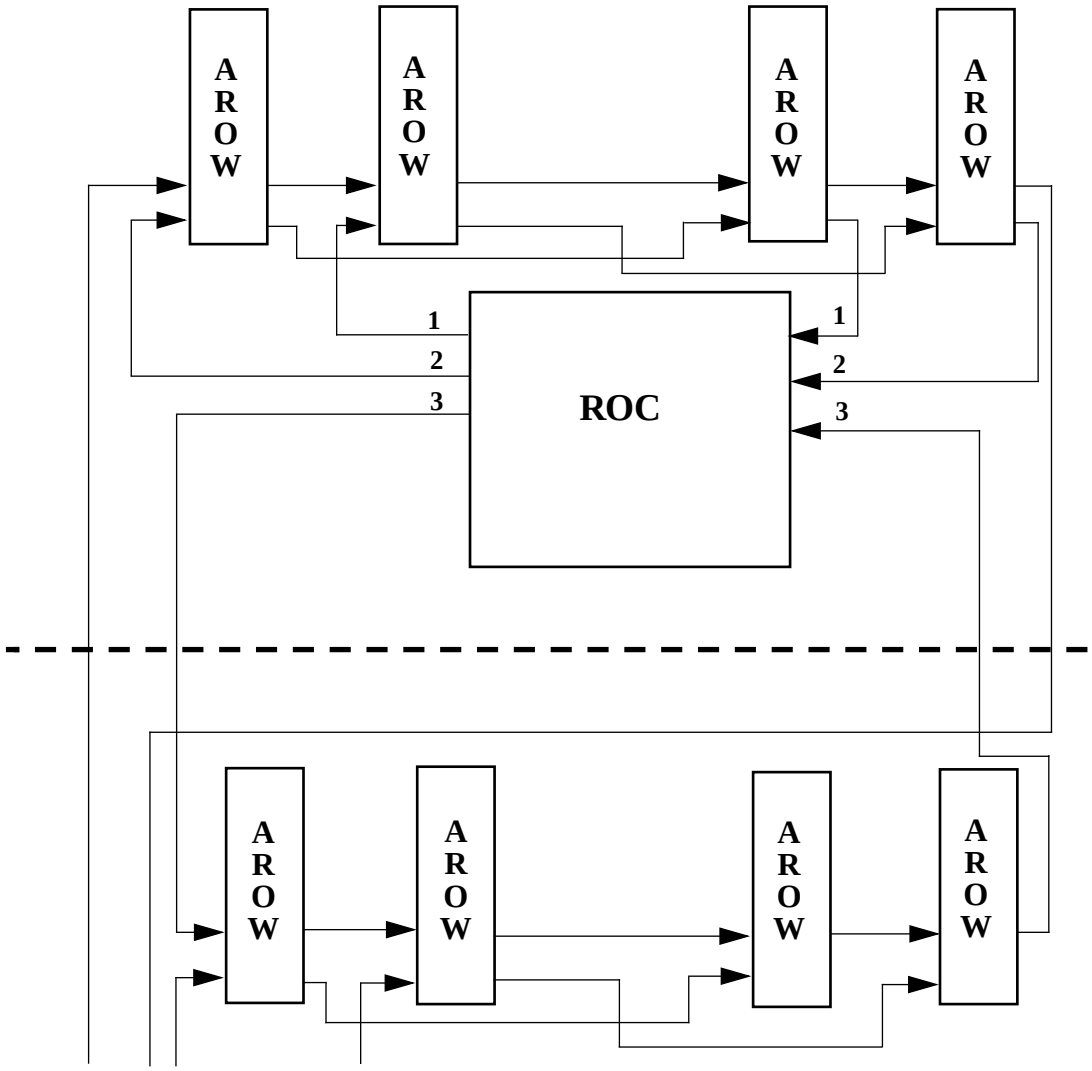
The ALARM option is implemented which means that this flag will go HIGH whenever there is a FIFO-overflow i e L1_Triggers are buffered but not read out properly. In this case ROC will be able to send out an error message as is shown in Table 3 on page 9. Directly after this the ROC will performe a physics data read-out procedure in an attemp to unload the FIFO and thereby reset the ”FIFO_overflow” flag. To make an active reset of this flag a Soft Reset(or Hard Reset) is needed which will empty the FIFO.

6. Bypass possibilities in the ROC

To ensure that no ”single point failure” will jeopardize the entire data collection from the module, a number of bypass facilities have been implemented as indicated above. In Figure 8 the TOKEN ring system is shown and it can bee seen how ROC not only sends out two tokens to its own AROW’s but also to the AROW’s of the other side to be able to read out that data. As each AROW contains its own input decoder, it is possible to make necessary settings for bypassing not depending whether or not the primary MONROC may be malfunctioning. As long as one MONROC and the DORIC connected are functioning, all or some of the data from both side of the detector module can be read out. This will however take some more time as the data then would have to be sent on one data line instead of two as in the normal case.

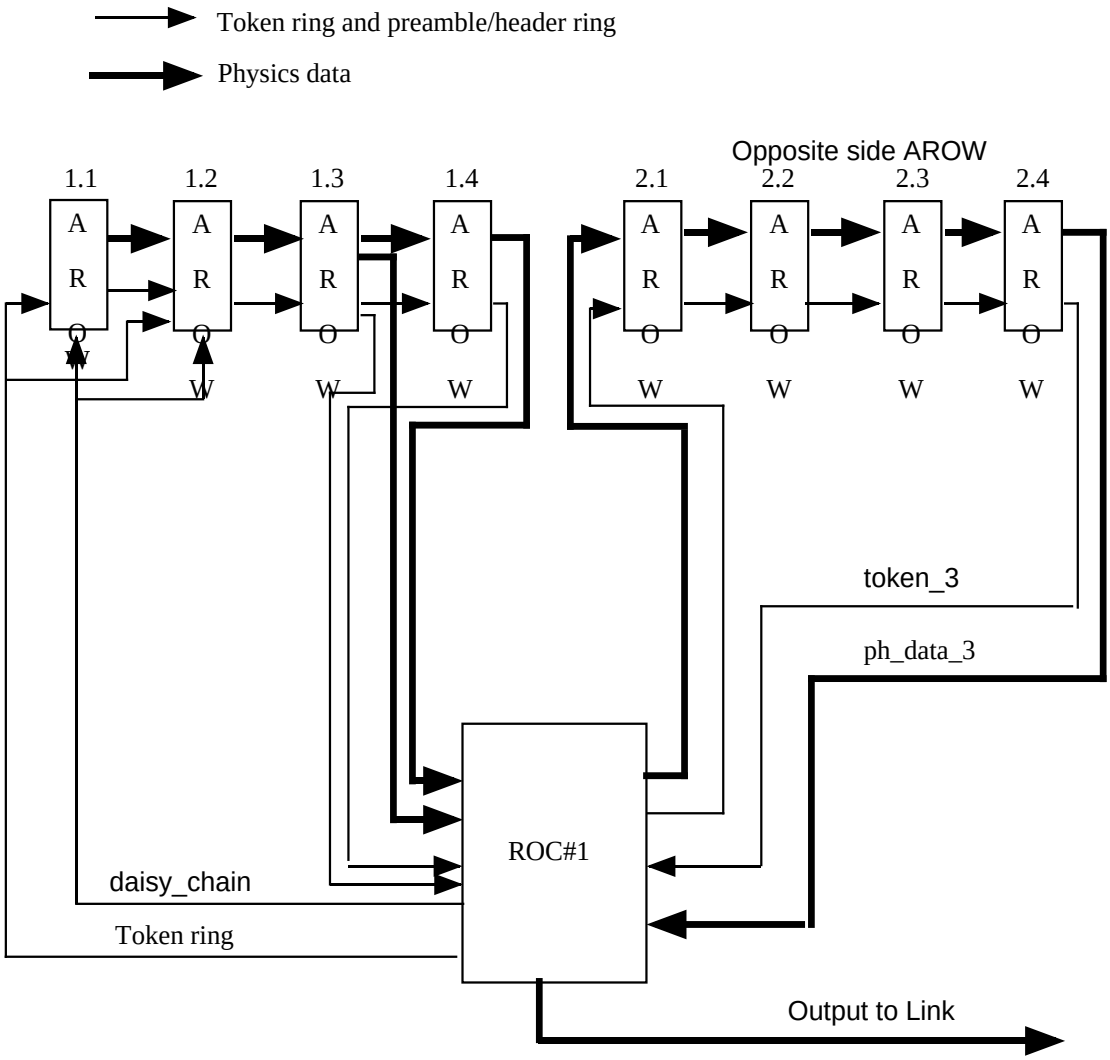
If an error occur so that data is hung up somewhere in the AROW cascade, this must be detected by the data acquisition system as it has sent down a L1-trigger but has not received any data on the link.

Fig. 8. Token ring bypass



Assume that we want to read out physics data from both the primary (same side) AROWs and the secondary (opposite side) AROWs as is illustrated in the diagram below. I this case there are eight AROW's in a cascade plus the ROC The reason we want to bypass the two links through the 1-bit shift register in MONROC (ROC) is for pure synchronization reasons; the physical length of the links could cause timing error if we did not use the shift registers as a latch-in.

Fig. 9. Read-out of physics data in mode ”bypass”(mode(1,0)=[1-])



7. MOC Monitor Controller

The MOC module starts to read in a number of channels upon receiving a monitor command (eq_mon=HIGH) from INDEC. When all channels have been read into the RAM (analogue to digital converted), MOC will send out a monitor request (mon_rq=HIGH) to the ROC to indicate that monitor data is waiting. When ROC replies with a monitor token, MOC starts to unload the RAM into the PSC and shifts out the data into the MPX_DORIC. Notice that it is the same PSC as is used for the physics data header. Two load signals select the inputs. A token back is sent on the last bit of monitor data back to the ROC. The data format with eight bits for the data resolution and the two start and stop bits are shown in Table 3 on page 9**.** No further ID codes are needed as the order of the monitored channels are predetermined by hardware.

Error detection if data ”hangs up” from MOC/PTS will be able by the DAQ as there is a preamble sent out but no trailing monitor data.

In Figure 10 a number of external units are connected to the MOC: ADC (ADC0809), MPX (4051) and a RAM . This is to illustrate the first implementation of the ASIC where no analogue components will be designed. The number of channels are determined by the internal counter to eight, but could easily be made programmable using a WCL in future implementations.

Fig. 10. Block diagram of the MOC MOnitor Controller

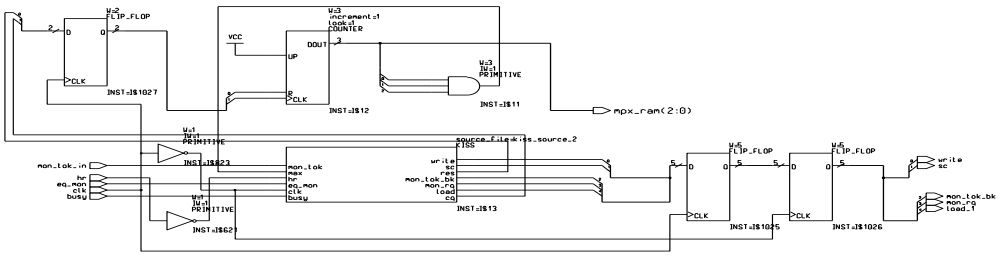
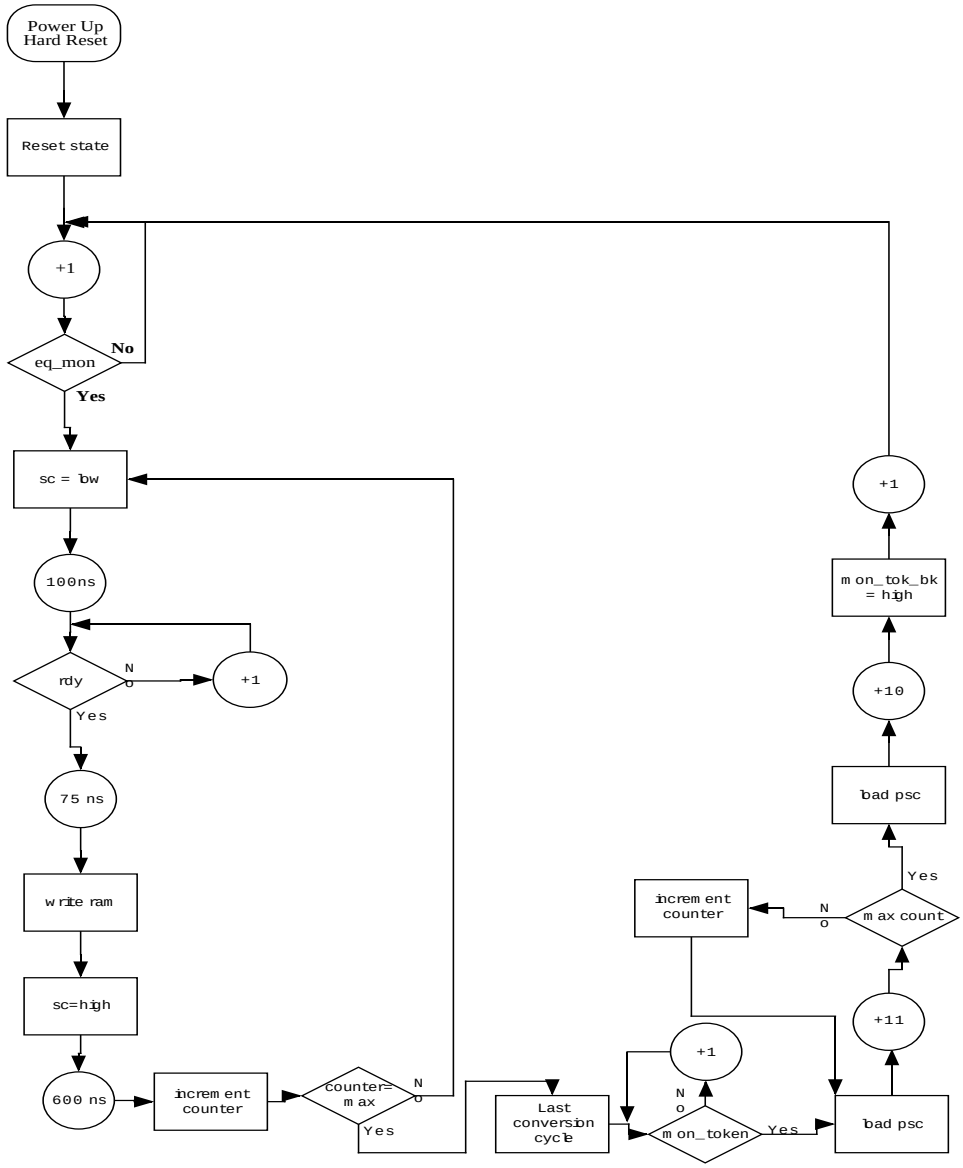


Fig. 11. Flow chart of the MOC



8. Connections the chip

A suggested pad lay-out is shown below in fig 12. What bias pins are needed for the voltage tranceiver is not known to us at this moment and has therefore been left out. It should be pointed out that there are only one pair of ctrl and bco inputs as the propose system with two pairs of these signal inputs and a semiautomatic selector according to R Nickerson could not be implemented at this moment.

Fig. 12. Proposed pad lay-out of MONROC chip

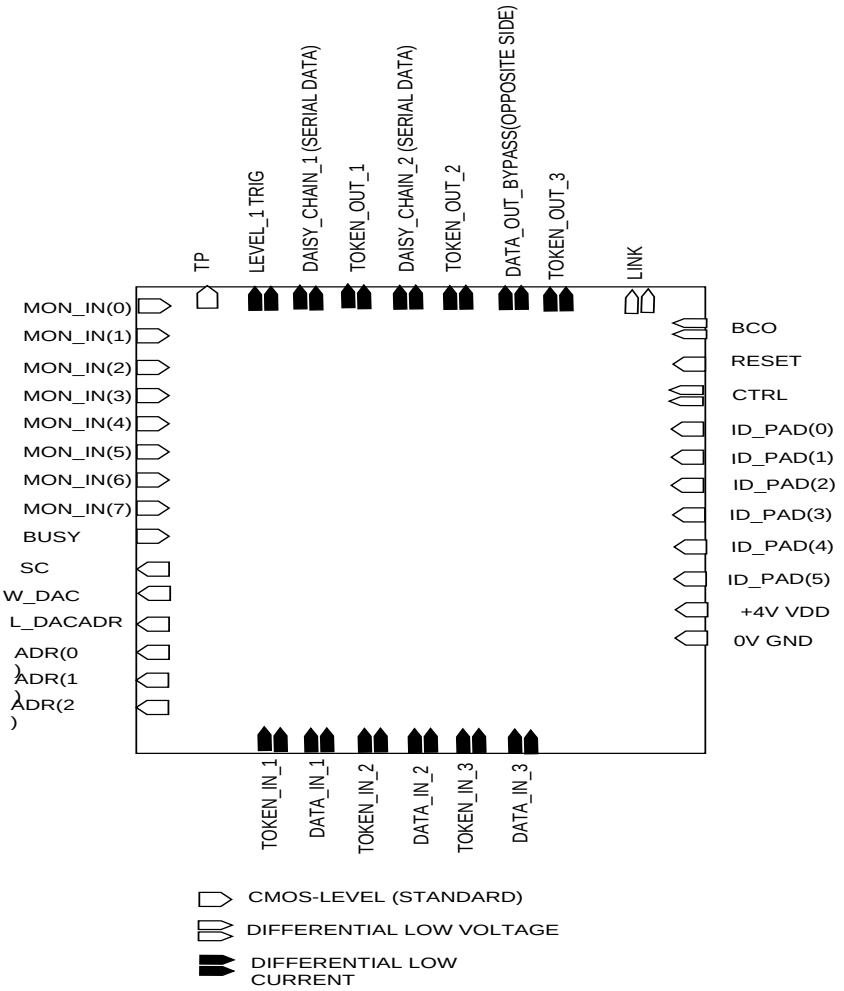


Table 4. Static inputs

Pos	Signal	Level (Volts)
1	Vdd	+4
2	Ground	0
3	ID_PAD (0)	High*
4	ID_PAD (1)	High
5	ID_PAD(2)	High
6	ID_PAD(3)	High
7	ID_PAD(4)	High
8	ID_PAD(5)	High

* High level = cmos (4V)

Table 5. Logic inputs

Pos	Signal	Level (Volts)
9	Reset	High
10	ctrl	Low Voltage*
11	ctrl(bar)	Low Voltage
12	bco	Low Voltage
13	bco(bar)	Low Voltage
14	token_in_1	Low Current
15	token_in_1(bar)	Low Current
16	token_in_2	Low Current
17	token_in_2(bar)	Low Current
18	token_in_3	Low Current
19	token_in_3(bar)	Low Current
20	data_in_1	Low Current
21	data_in_1(bar)	Low Current
22	data_in_2	Low Current
23	data_in_2(bar)	Low Current
24	data_in_3	Low Current
25	data_in_3(bar)	Low Current
26	mon_in(0)	High
27	mon_in(1)	High
28	mon_in(2)	High
29	mon_in(3)	High
30	mon_in(4)	High
31	mon_in(5)	High
32	mon_in(6)	High
33	mon_in(7)	High
34	busy	High

* Low level differential voltage and current signals

Table 6. Logic outputs

Pos	Signal	Level (Volts)
35	level 1 trig	Low Current
36	level 1 trig(bar)	Low Current
37	token_out_1	Low Current
38	token_out_1(bar)	Low Current
39	token_out_2	Low Current
40	token_out_2(bar)	Low Current
41	token_out_3	Low Current

Table 6. Logic outputs (cont'd)

Pos	Signal	Level (Volts)
42	token_out_3(bar)	Low Current
43	data_out_bypass	Low Current
44	data_out_bypass (bar)	Low Current
45	link	Low Voltage
46	linkbar)	Low Voltage
47	tp	High
48	sc	High
49	adr(0)	High
50	adr(1)	High
51	adr(2)	High
52	Write_dac	High
53	l_dacadr	High

9. Design and manufacturing

All design work was made using Mentor Graphics Idea Station v 8.4 The functional blocks in the first design made in M-language (GDT) were translated by hand into VHDL in order to synthesize in Autologic (MGC). The state machines were written using the high level description language KISS (Autologic BLOCKS). In the VHDL version of INDEC, ROC and MOC there are no Mealy inputs but an ordinary synchronious output Moore machine where each output is associated with an exclusive state. Output latches have been placed by hand on chritical signals to ensure glitch-free performance

The netlist version of this autologic design was shiped over to Sicon (Linkoping Sweden) for mask design which will be transfered to AMS (Austria) for processing. The chip will be delivered in June -96.

10. Environmental impact

No animal testing or the use of toxical chemicals has taken place during the design work in Uppsala. All paper prinouts will be recycled by the " Returpapperscentralen". No aluminium cans (Coca Cola, Sprite etc) have been used by the design staff but ordinary recycleable glass bottle.

Appendix A. QuickSim traces for MONROC

Fig. A-1. Reset Clock Through Mode

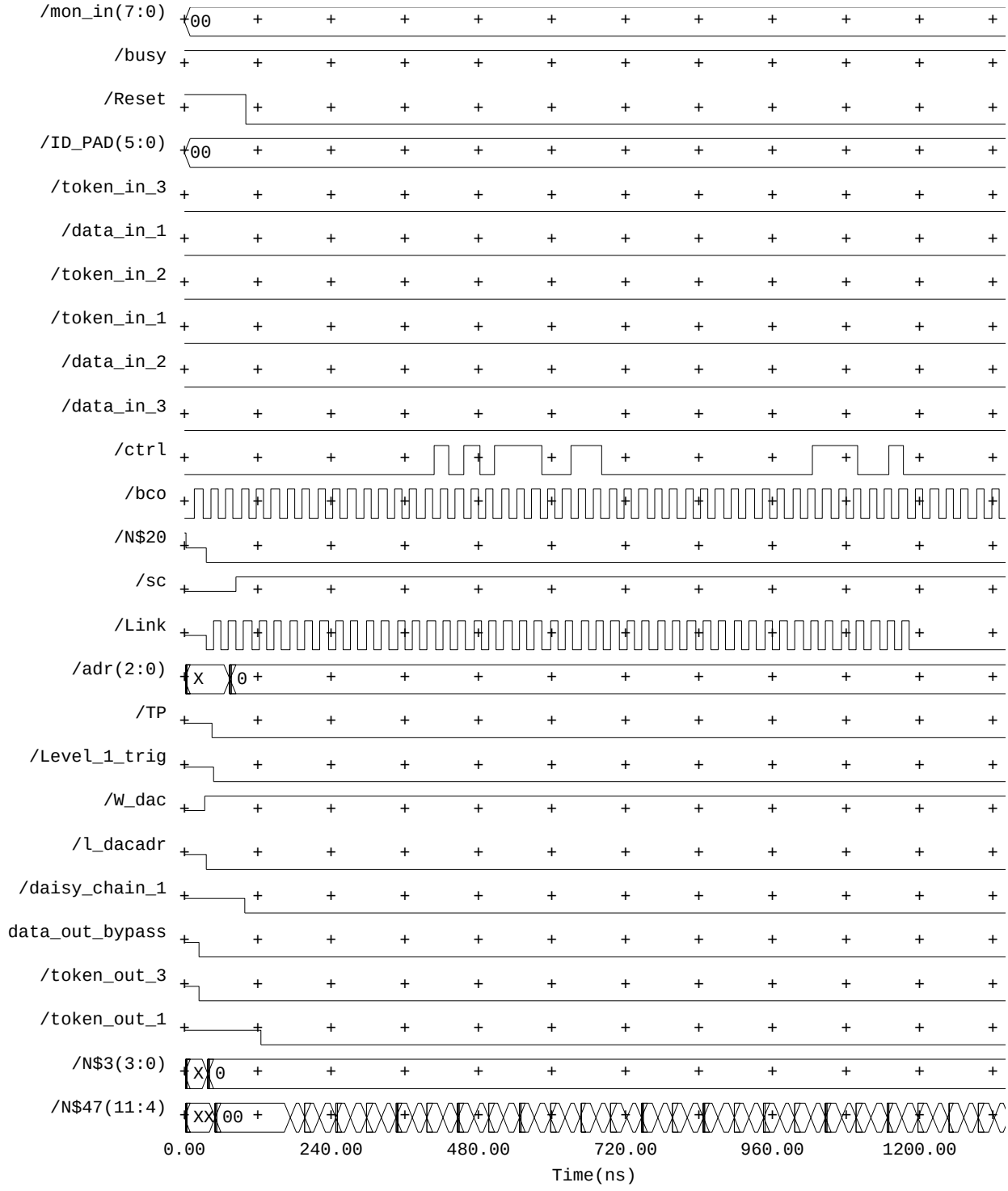


Fig. A-2. Soft Reset (Notice L1_trig and bco counter output at bottom)

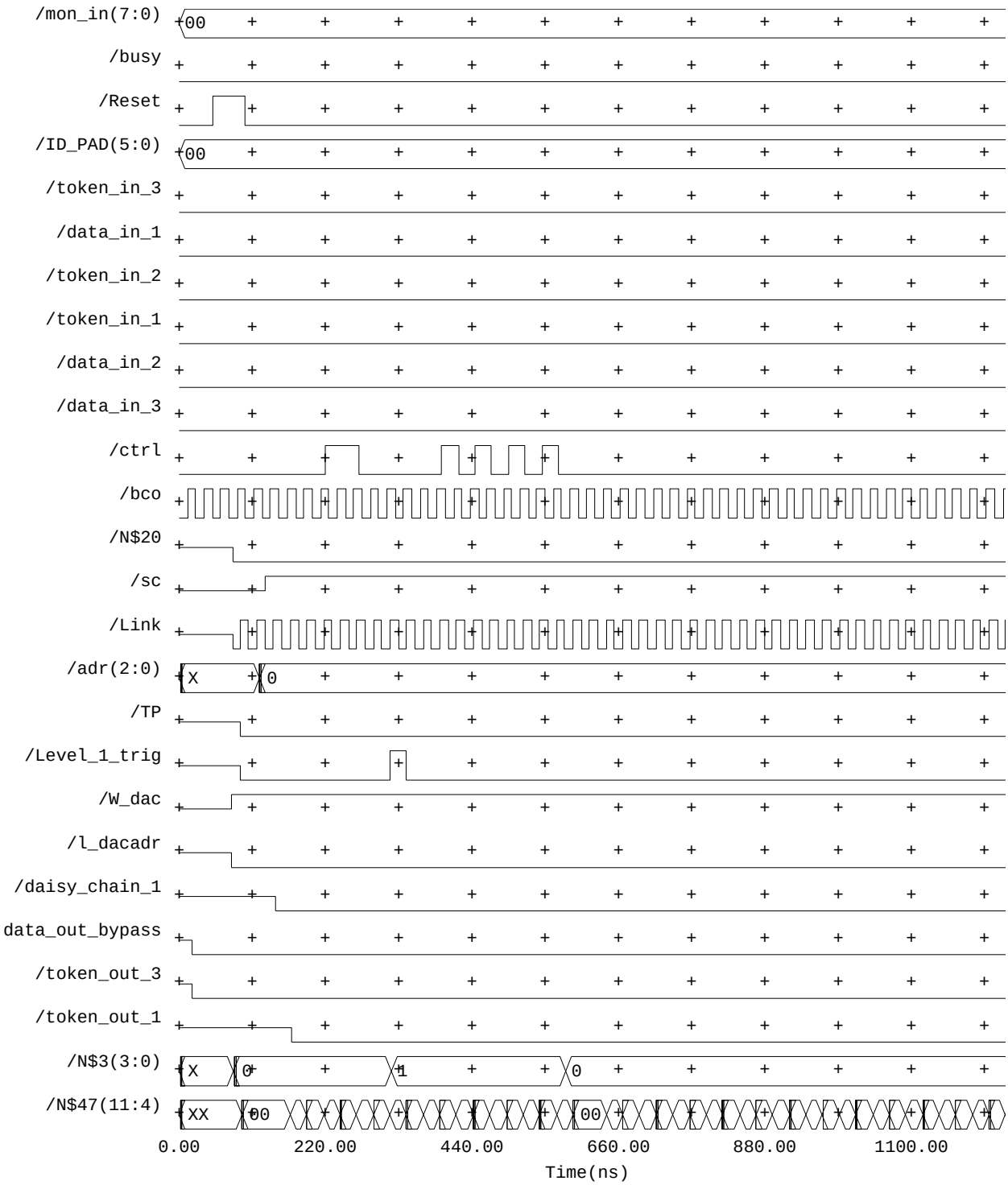


Fig. A-3. Set clock through mode

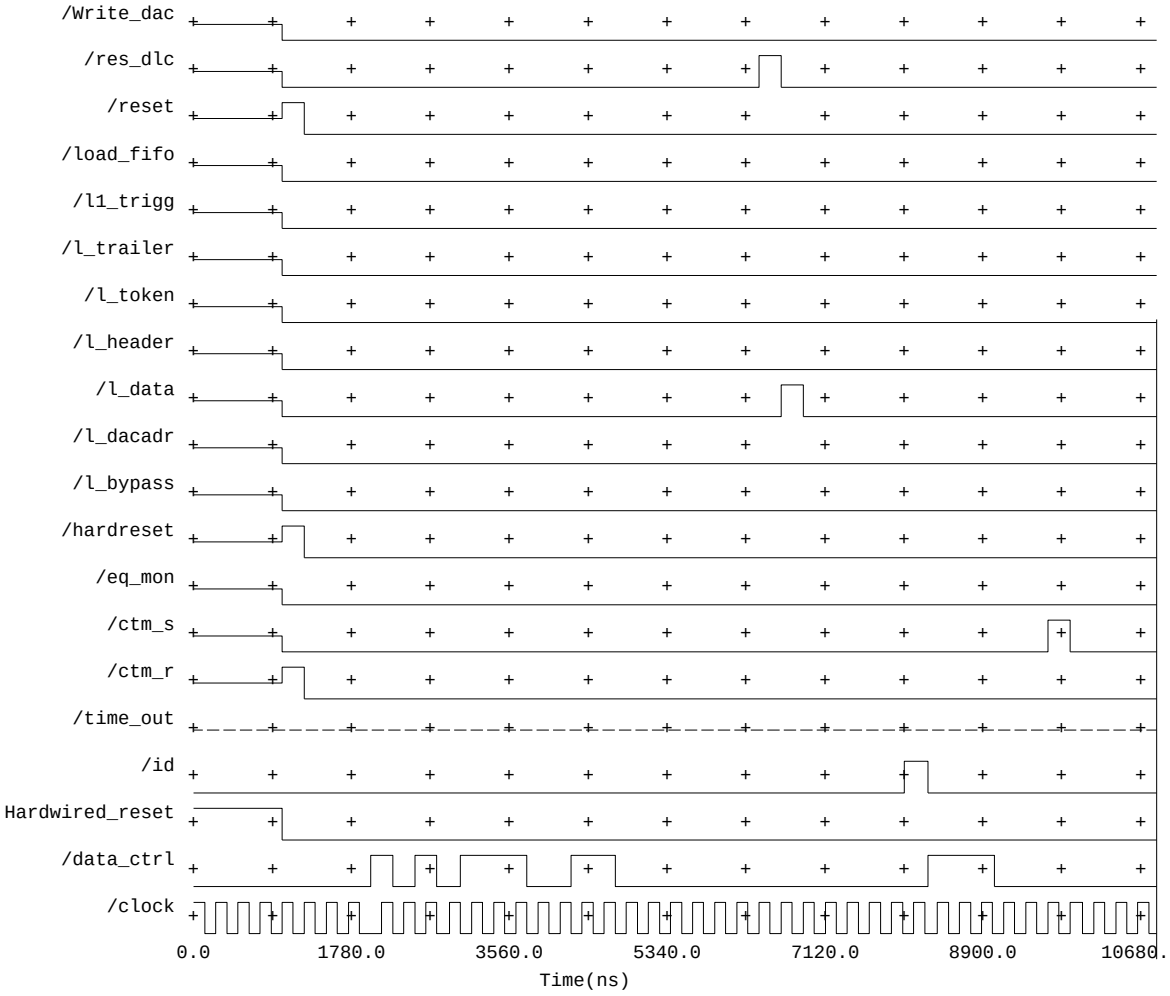


Fig. A-4. Setting of DAC data

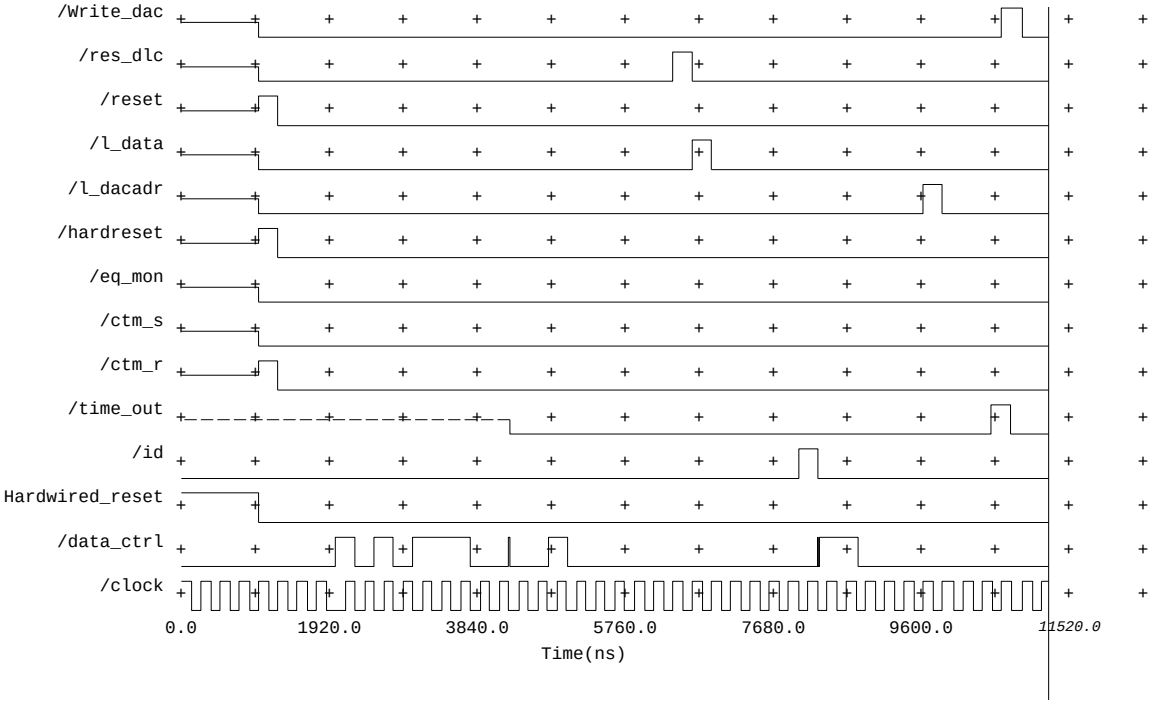


Fig. A-5. Hard reset

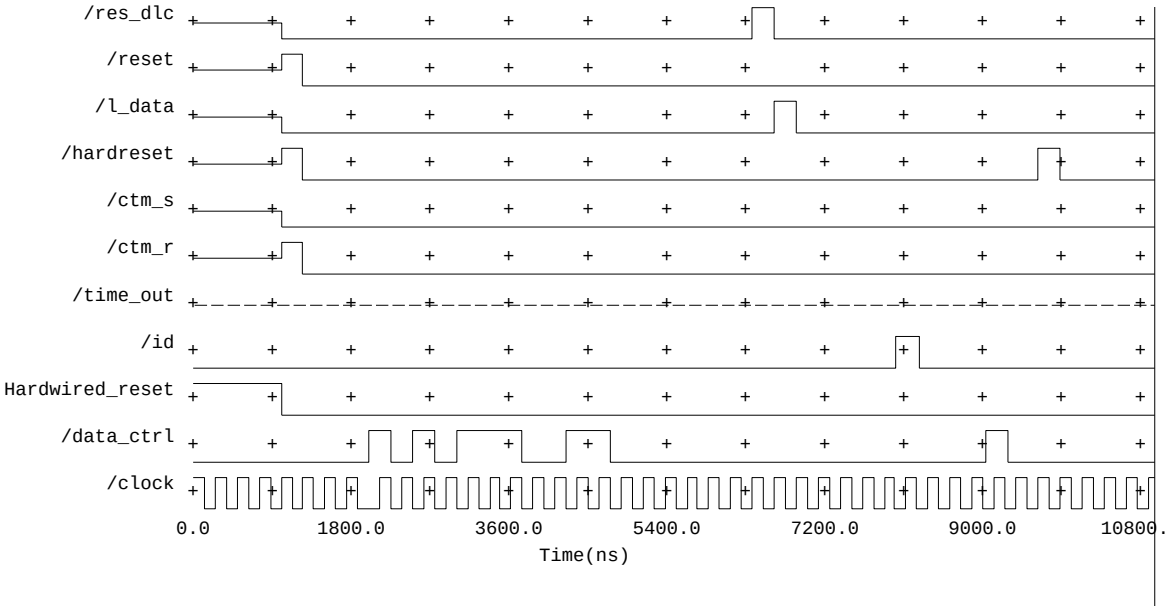


Fig. A-6. Set header offset (data =<0000 0000>)

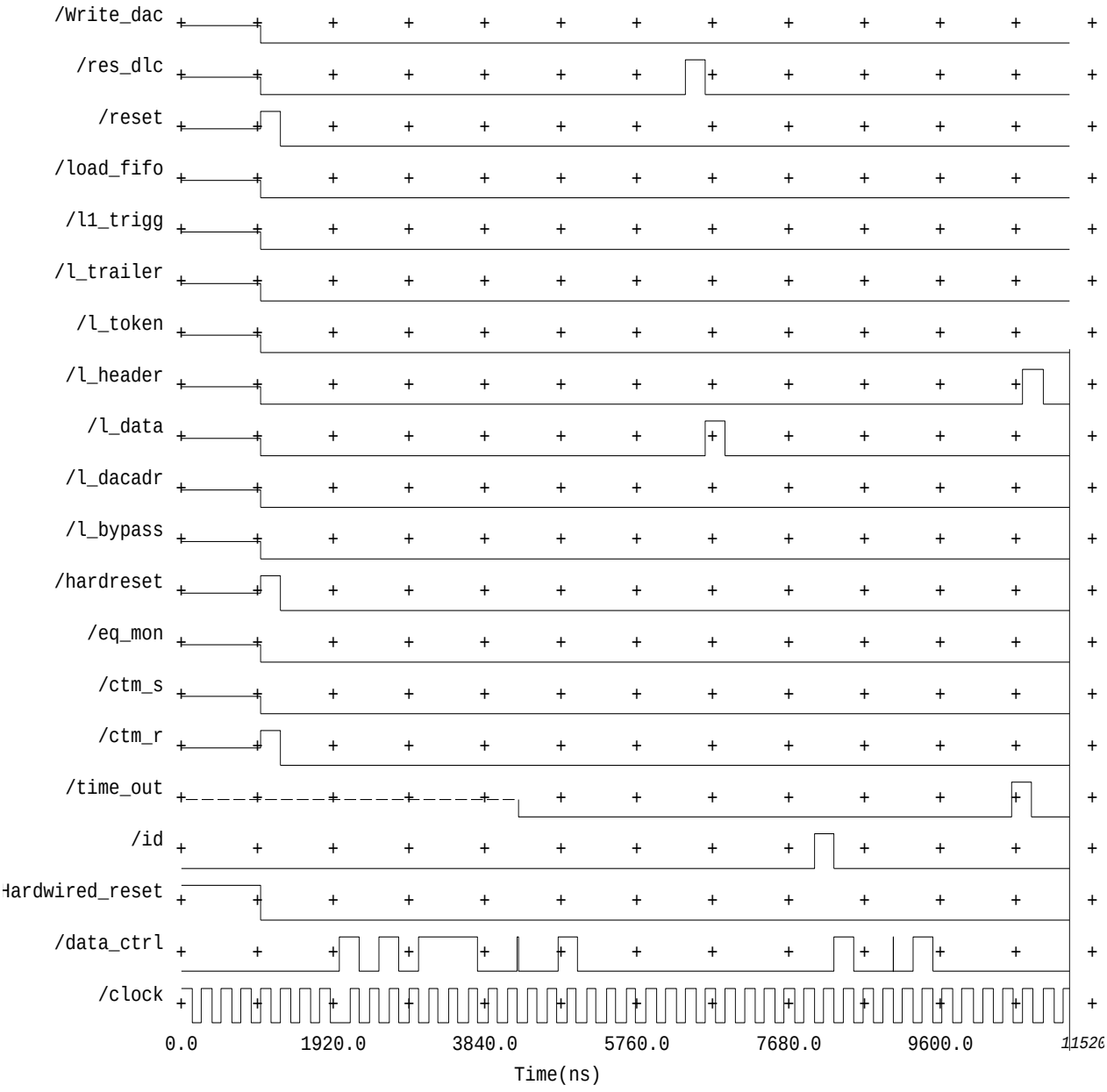


Fig. A-7. ID generation

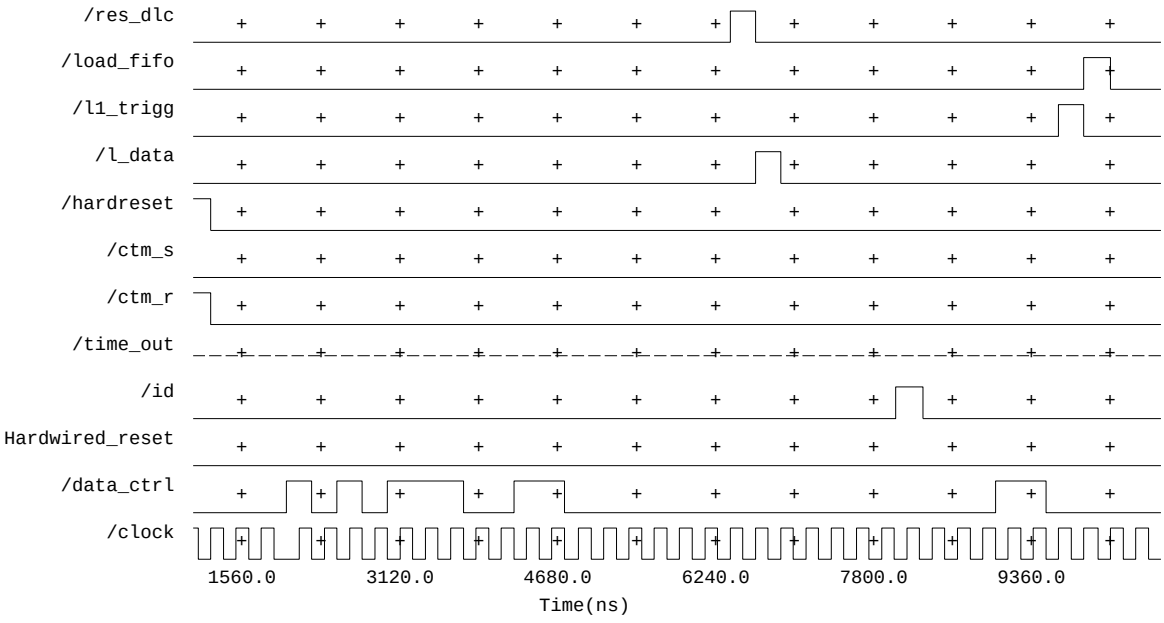


Fig. A-8. Level 1 trigger Mode =0 (default by power up)

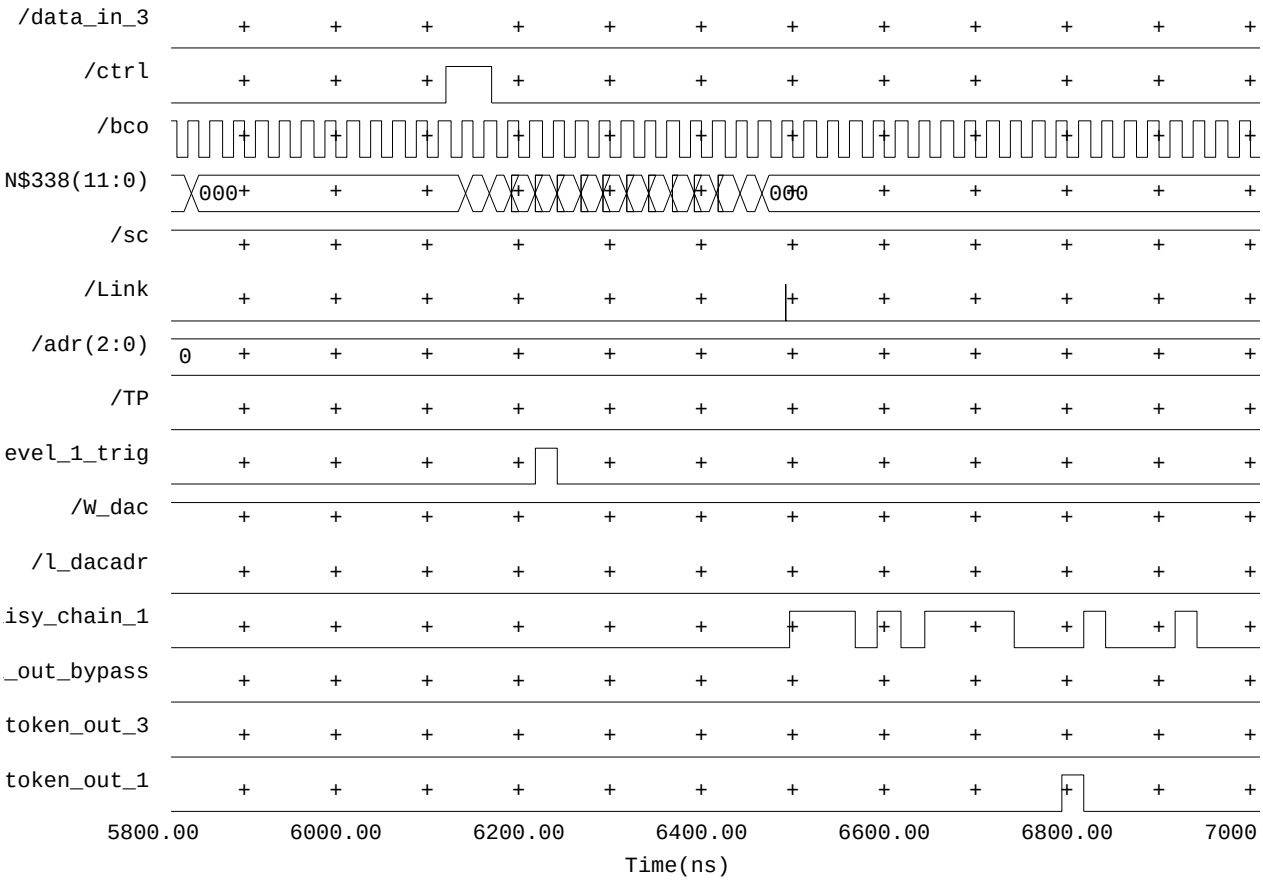


Fig. A-9. Physics data return. Mode =0

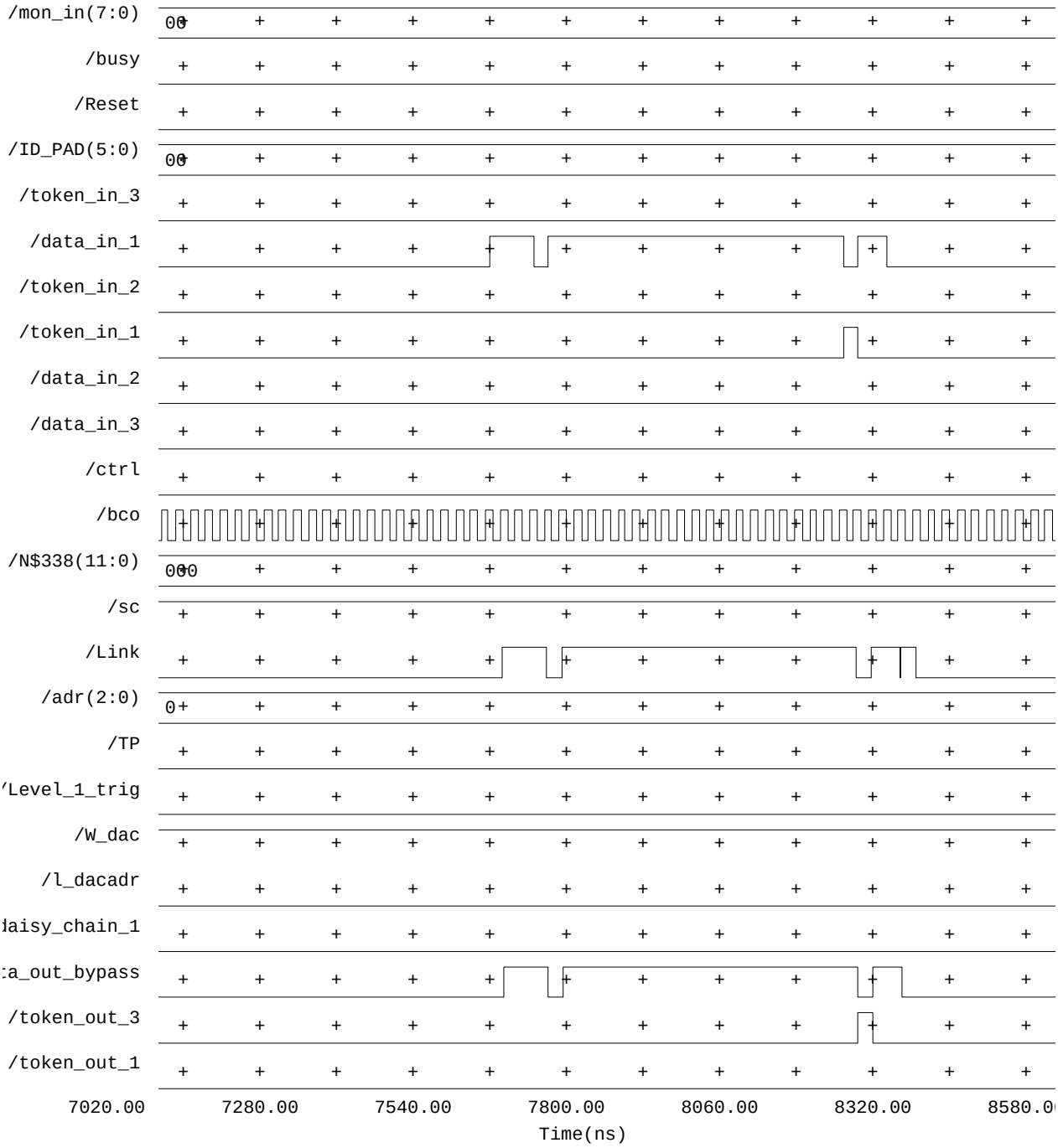


Fig. A-10. Set token launch time (data = <0000>)

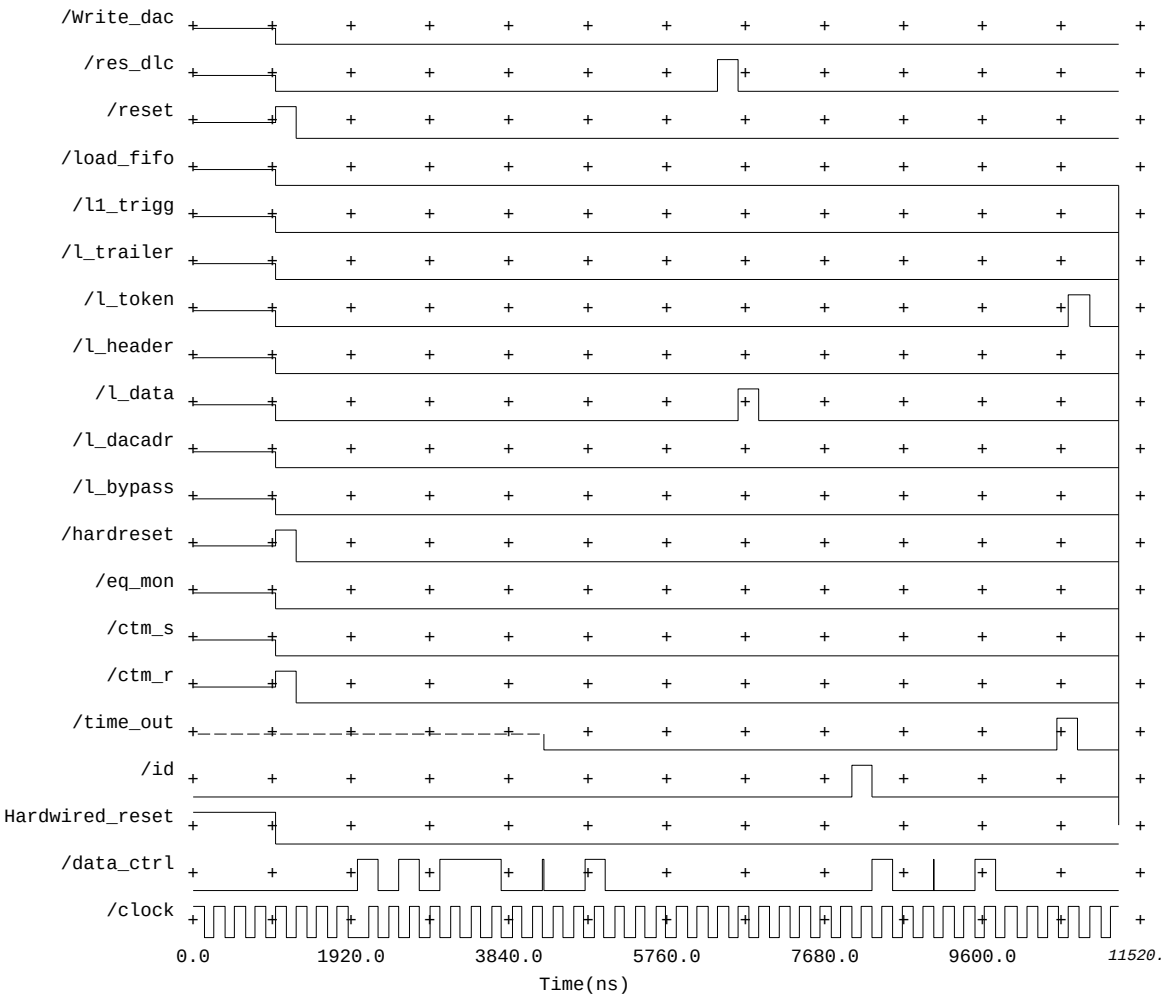


Fig. A-11. Set trailer length (data = <0000>)

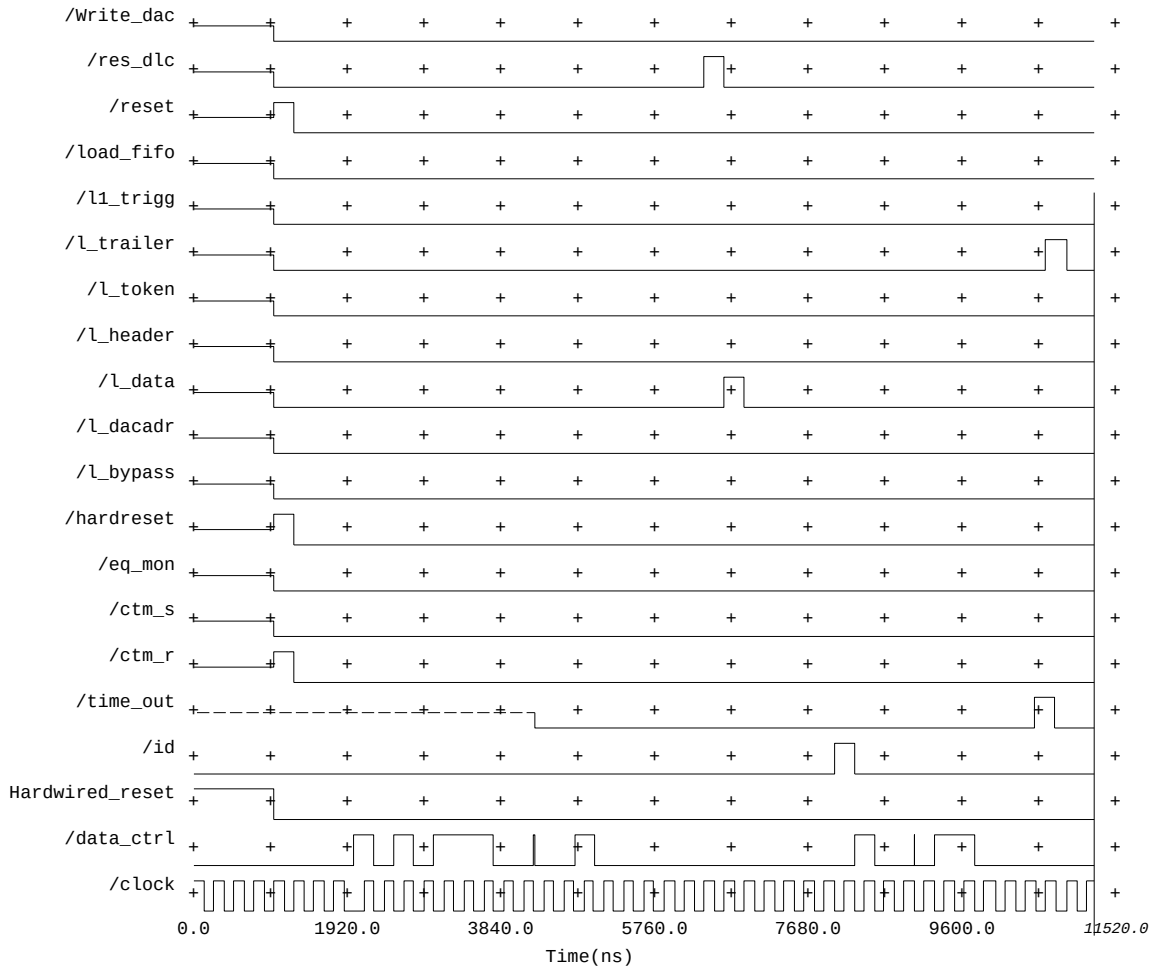


Fig. A-12. Monitoring sequence. Overwiew

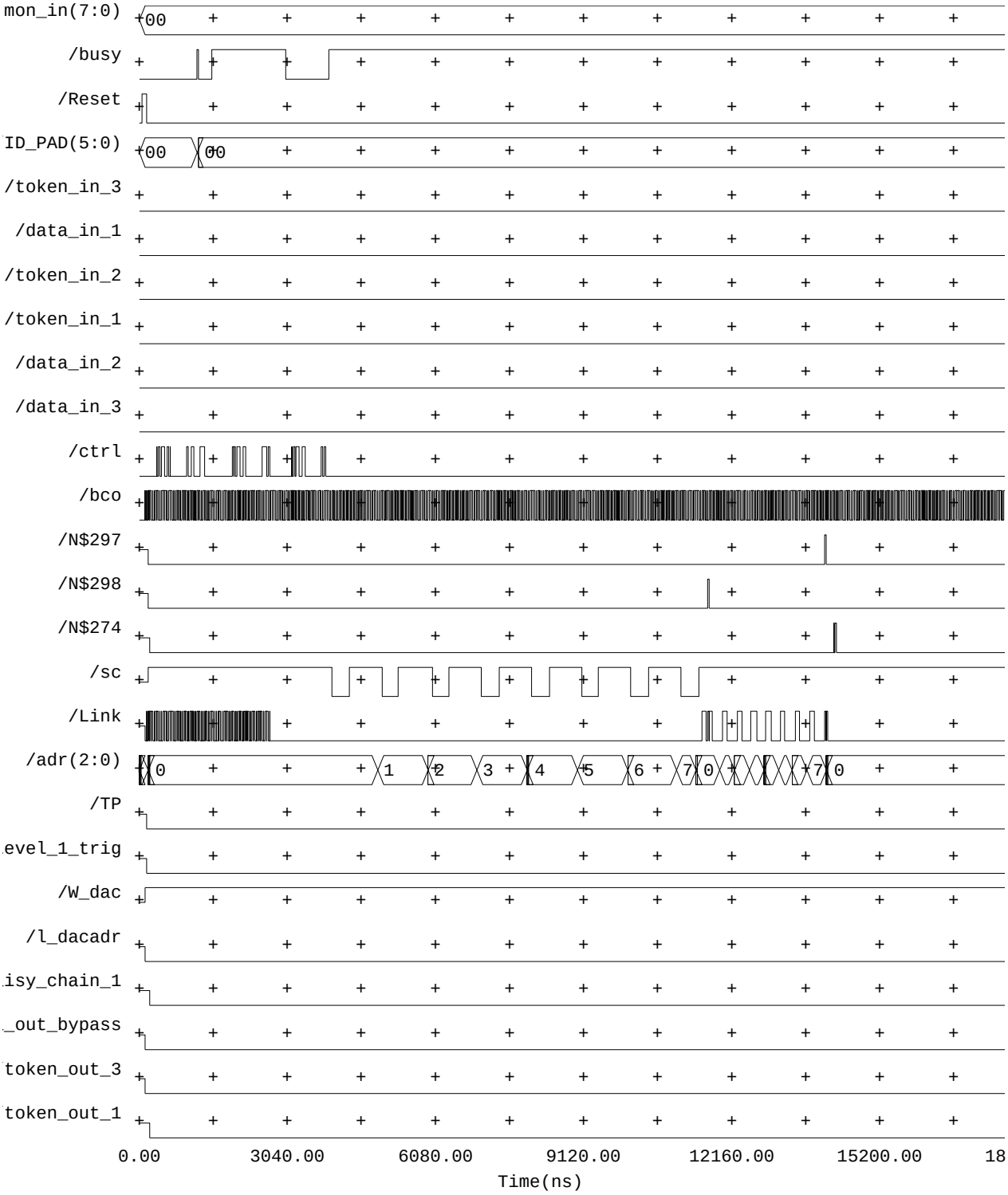
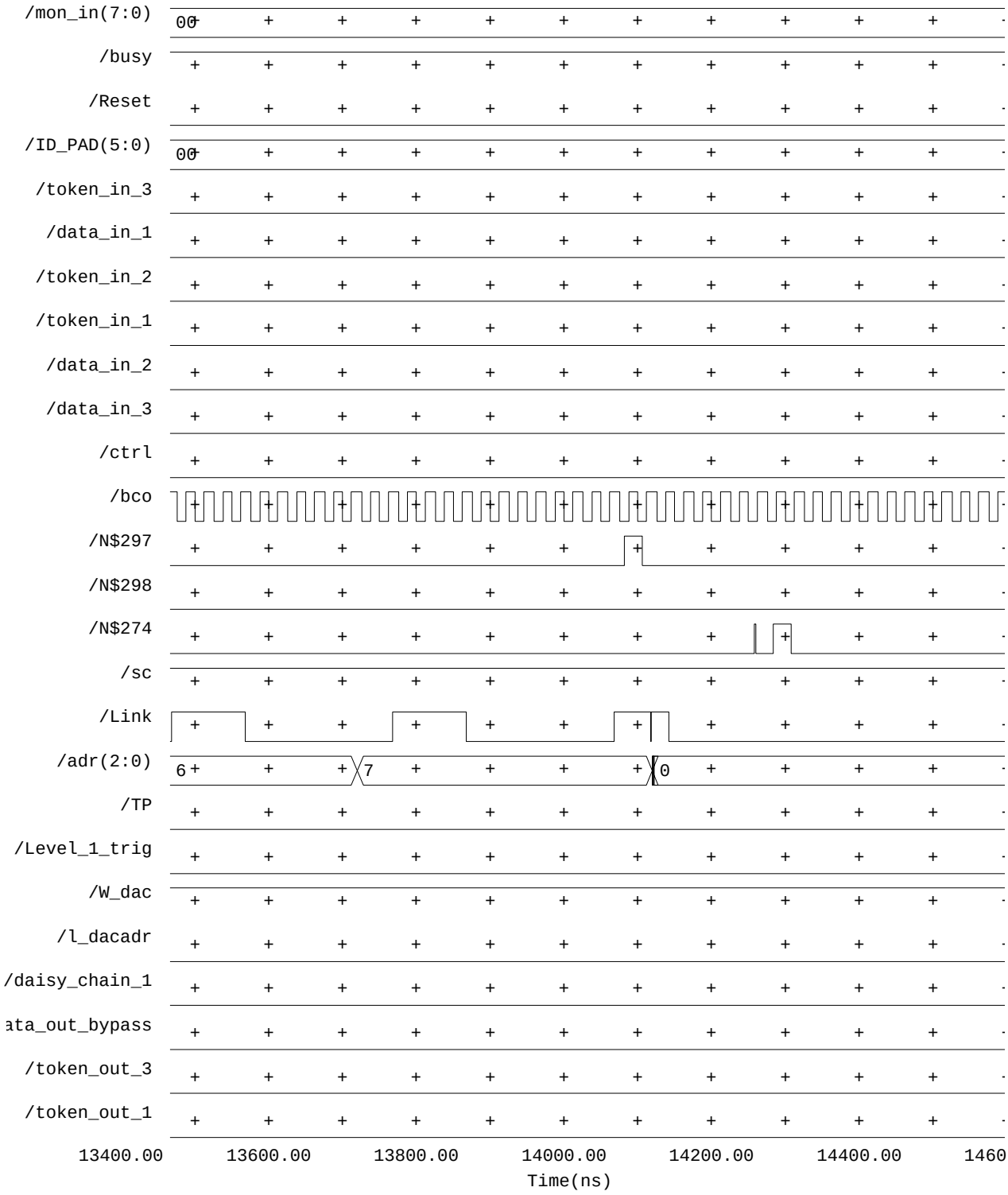


Fig. A-13. Monitor Token back. Magnification



Appendix B. KISS sources

Fig. B-1. Input decoder

```
########################## Input Decoder with Asynchronious External Reset ###########
# Modified 960403
# Note:
#
#
# Fast Codes
# 101 0101Soft Reset: resets BCO,L1_T and Fifo
# 101 0100BCO Reset
# 101 0010Send bipolar TP ( cmos high signal 25ns)
# 110 Level 1 Trigger
# Hard Reset and Power Up sets CTM!
# Recommended DAC AD7228

.i 3
.o 17
.inputsdata time_out id
.outputsli_t hr res res_dlc load_fifo l_dl l_tl l_ho l_tr l_bm eq_mon ctm_s ctm_r l_dacadr w_dac tp bco_reset
.clock cl
.async_reset hardwired_reset init_state
#.resetstate init_state
.resetoutput 00010000000100100

#
#
# State transitions
# After one L1_trigger, the machine rests for two clock cycles!
#
--- init_state p0 00010000000100100# Hard Wired Reset state ctm is set
#
########################### Looking for three bit preamble ############################
#
0-- p0   p0 01110000000000101
1-- p0   p1 01110000000000101
0-- p1   p601110000000000101
1-- p1p201110000000000101
0-- p2p3 01110000000000101
1-- p2p1 01110000000000101
--- p3p4111100000000000101# L1_Trigger
--- p4p5 01111000000000101# load_fifo
--- p5p0011100000000000101# Rest
0-- p6p0011100000000000101# Rest
1-- p6c001110000000000101
############################ Decoding command field ###################################
0-- c0c101110000000000101
1-- c0c12 01110000000000101
0-- c1c901110000000000101
1-- c1c201110000000000101
```

```
0-- c2c401110000000000101
1-- c2c301110000000000101
0-- c3c701110000000000101
1-- c3dl0 01110000000000101
0-- c4c501110000000000101
1-- c4c601110000000000101
0-- c5p0011100000000000100# Soft Reset BCO counter (101 0100)
1-- c5p101110000000000100
0-- c6p0010100000000000100# Soft Reset (101 0101)
1-- c6p101010000000000100
0-- c7p0011100000000000101# Not used (101 0110)
1-- c7p101110000000000101
#--- c8dl0 01110000000000101
0-- c9c14 01110000000000101
1-- c9 c10 01110000000000101
0-- c10 c11 01110000000000101
1-- c10 c15 01110000000000101
0-- c11 p0 0111000000000111#Send Test Pulse (101 0010)
1-- c11 p1 01110000000000111
--- c12 c1301110000000000101
--- c13 c14 01110000000000101
--- c14 c15 01110000000000101
0-- c15 p0 01110000000000101
1-- c15 p1 01110000000000101
#
############################## Pick up data_length ###################################
#
--- dl0 dl1 01110000000000101
--- dl1 dl2 01110000000000101
--- dl2 dl3 01110000000000101
--- dl3 dl4 01110000000000101
--- dl4 dl5 01110000000000101
--- dl5 dl6 01110000000000101
--- dl6 dl7 01110000000000101
--- dl7 dl8 01110000000000101
--- dl8 dl9 01110000000000101
--- dl9 dl10 01110000000000101
--- dl10 dl11 01100000000000101 # Reset data_length counter(goes via dff)
--- dl11 dl12 01110000000000101
--- dl12 a0 0111010000000101# Latch data_length
#
############################### Pick up chip address( six bits!) ######################
#
--- a0a1011100000000000101
--- a1a2011100000000000101
--- a2a3011100000000000101
--- a3 a401110000000000101
--- a4a5 01110000000000101
--0 a5wait 01110000000000101
1-1 a5sc1 01110000000000101# Jump to sc1 if ID AND ctrl.
0-1 a5sc23 01110000000000101# Jump to sc23 if ID AND /ctrl

#
############################### Check if ID_gen is issued ############################
```

```
#
#0--- id0 id1 01110000000000101
#1--- id0 wait 01110000000000101
#0--- id1 id2 01110000000000101
#1--- id1 wait 01110000000000101
#0--- id2 id3 01110000000000101
#1--- id2 wait 01110000000000101
#0--- id3 wait 01110000000000101
#1--- id3 id4 01110000000000101
#0--- id4 wait 01110000000000101
#1--- id4 id5 01110000000000101
#0--- id5 p3 01110000000000101
#1--- id5 wait 01110000000000101

#
############################### Pick up sub command #######################################
# One rest clock cycle (p0) after each command
#0-- sc0 sc23 01110000000000101
#1-- sc0 sc1 01110000000000101
0-- sc1 sc3 01110000000000101
1-- sc1 sc2 01110000000000101
0-- sc2 sc13 01110000000000101
1-- sc2 sc18 01110000000000101
0-- sc3 sc4 01110000000000101
1-- sc3 sc5 01110000000000101
0-- sc4 sc9 01110000000000101
1-- sc4 sc10 01110000000000101
0-- sc5 sc6 01110000000000101
1-- sc5 wait 01110000000000101
0-- sc6 sc7 01110000000000101
1-- sc6 wait 01110000000000101
0-- sc7 sc8 01110000000000101
1-- sc7 wait 01110000000000101
--- sc8 p0 01110000001000101# Initiate monitor data readout
0-- sc9 sc11 01110000000000101
1-- sc9 sc12 01110000000000101
0-- sc10 sc17 01110000000000101
1-- sc10 wait 01110000000000101
0-- sc11 wait 01110000000000101
1-- sc11 token_l 01110000000000101
0-- sc12 header_o 01110000000000101
1-- sc12 trailer_l 01110000000000101
--- sc13 sc14 01110000000000101
--- sc14 sc15 01110000000000101
--- sc15 sc16 01110000000000101
--- sc16 write_d 01110000000000101# DAC address present
0-- sc17 bypass_m 01110000000000101
1-- sc17 wait 01110000000000101
0-- sc18 sc19 01110000000000101
1-- sc18 wait 01110000000000101
0-- sc19 sc20 01110000000000101
1-- sc19 wait 01110000000000101
0-- sc20 sc21 01110000000000101
1-- sc20 sc22 01110000000000101
```

```
--- sc21 p0 01110000000100101# Set clock through mode
--- sc22 p0 01110000000010101# Reset clock through mode
0-- sc23 sc24 01110000000000101
1-- sc23 wait 01110000000000101
0-- sc24 sc25 01110000000000101
1-- sc24 wait 01110000000000101
0-- sc25 wait 01110000000000101
1-- sc25 sc26 01110000000000101
0-- sc26 sc27 01110000000000101
1-- sc26 sc28 01110000000000101
0-- sc27 hardreset 01110000000000101# hardreset
1-- sc27 wait 01110000000000101
1-- sc28 wait 01110000000000101
0-- sc28 p3 01110000000000101# ID_gen (000110)
#
############################### Command destination ###################################
#
--- hardreset init_state 00010000000100100
-0- wait wait 01110000000000101# wait for time_out
-1- wait p0 01110000000000101
-0- token_l token_l 01110000000000101
-1- token_l to_latch 01110000000000101
--- to_latch p0 01110010000000101# Latch token launch delay parameter
-0- header_o header_o 01110000000000101
-1- header_o ho_latch 01110000000000101
--- ho_latch p0 01110001000000101# Latch header offset
-0- trailer_l trailer_l 01110000000000101
-1- trailer_l t_latch 01110000000000101
--- t_latch p0 01110000100000101# Latch trailer length parameter
-0- bypass_m bypass_m 01110000000000101
-1- bypass_m bm_latch 01110000000000101
--- bm_latch p0 01110000010000101# Latch bypass mode for ROC
-0- write_d write_d 01110000000000101
-1- write_d w_dac 01110000000001101# Write data into DAC latch
--- w_dac w_dac1 01110000000000001# Write signal 150 ns to presumptive DAC
--- w_dac1 w_dac201110000000000001
--- w_dac2 w_dac3 01110000000000001
--- w_dac3 w_dac4 01110000000000001
--- w_dac4 w_dac5 01110000000000001
--- w_dac5 w_dac6 01110000000000001
--- w_dac6 p0 01110000000000001



.e
```

Fig. B-2. Read-Out Controller

```
######################## Readout Controller ############################
######################## Modified 960326  ############################
```

```
# Programmable token launch (hoc)
#
# This ROC is clocked by rising edge of BCO!!!!
# Token back Three clock cycles in advance of end of data!!
# Outputs unload_fifo and load_psc via falling edge dff
# outputs "roc",res_hoc, res_trc and res_tlc are dealyed 1 clock cycle via dff
# Daisy-chain is quenched by "res_hoc".
# s0 selects roc (0) or psc (1) (via dff)
# s1 selects roc/psc (0) or data back (1) (via dff)
.i 7
.o 9
.inputs  eq_l1t_fifo eq_tlc eq_trc   mo_tok_bk  ph_tok_bk  mon_rq   overflow
#    #    #
.outputs res_tlc res_hoc res_trc    load_psc unload_fifo roc    mo_tok s0 s1
#########    #     #
.clock cl
.extreset hr
.resetstate init_state
.resetoutput 000000000
# Moore machine
# State transitions on rising clock edge!
#
######################### Scan physics data and monitor data #########
------- init_state s0 000000000
------0 s0 s1 000000000
------1 s0 of0 000000000# Overflow
1------ s1 s2 000000000# No trigger pending.
0------ s1 l0 000000000 # Triggers!
-----0- s2 s0 000000000
#
######################### Read out monitor data ######################
-----1- s2 m0 000000000# Monitor request!
------- m0 m1 000001000# Send out preamble (s1=0)
------- m1 m2 000001000
------- m2 m3 000001000
------- m3 m4 000000000
------- m4 m5 000001000
------- m5 m6 000001100# Monitor token out
------- m6 m7 000000010# Select PSC(s0=1
---0--- m7 m7 000000010# Wait for monitor token back
---1--- m7 m8 000000010
------- m8 l25 001001000# Append trailer (PSC connected one clock cycle more)
#
######################### Send out error message "overflow" ###########
------- of0 of1 000001000# Send out preamble (s1=0)
------- of1 of2 000001000
------- of2 of3 000001000
------- of3 of4 000000000
------- of4 of5 000001000
------- of5 of6 000001000
------- of6 of7 000000000
------- of7 of8 000000000
------- of8 of9 001001000# Append trailer start trc
--0---- of9 of9 001000000# Wait for trc
--1---- of9 l0 001000000# Return to read_out data to decrement fifo.

#
######################### Read out physics data ######################
#
```

```
------- l0 l1 100010000# Unload FIFO, start tlc
-0----- l1 l1 100000000# Wait for tlc
-1----- l1 l2 100000001
------- l2 l3 110001001# Send preamble 111010 via daisy-chain. DORIC connected to data(s1=1) start hoc
------- l3 l4 010001001
------- l4 l5 010001001
------- l5 l6 010000001
------- l6 l7 010001001
------- l7 l8 010000001
------- l8 l9 010100011 # Load PSC(s0=1 select PSC)
------- l9 l10 010000011
------- l10 l11 010000011
------- l11 l12 010000011
------- l12 l13 010000011
------- l13 l14 010000011
------- l14 l15 010000011
------- l15 l16 010000011
------- l16 l17 010000011
------- l17 l18 010000011
------- l18 l19 010000011
------- l19 l20 010000011# Quench daisy_chain
------- l20 l21 000000001# Reset hoc
----0-- l21 l21 000000001 # Wait for physics token back
----1-- l21 l22 000000001
------- l22 l23 000000001
------- l23 l24 000000001
------- l24 l25 001001000# Append trailer, start trc
--0---- l25 l25 001000000# Wait for trc DORIC connected to ROC
--1---- l25 s0 001000000# Return to start
.e
```

Fig. B-3. Monitor Controller

```
################################ Monitor Controller ################################
################################ Modified 960403###################################
# Connected to  DAC with neg rdy (MAX158/AD7828 in mode 0 )
# The conversion starts and the adress latch at the falling edge of sc (connected to CS and RD on ADC)
# A 600 ns wait loop is implemented between sc.
# Token back on 3:nd last data (ROC is delayed by dff)
# Internal RAM with positive write
# All outputs via dff
.i 4
.o 7
.inputs eq_mon mon_tok busy max
.outputs mon_rq mon_tok_bk sc write load res cq
.clock clk
.extreset hr
.resetstate init_state
.resetoutput 0010010
############################### State transitions ##################################
---- init_state wmi0 0010010
0--- wmi0 wmi0 0010010# Wait for monitor initialize
1--- wmi0 la1 0010010
---- la1 la2 0000000# Start conversion
---- la2 la3 0000000# Wait for rdy to go low
---- la3 la4 0000000
```

```
---- la4 la5 0000000
---- la5 la6 0000000
---- la6 m1 0000000
--0- m1 m1 0000000# Wait for rdy to go high
--1- m1 m2_1 0000000
---- m2_1 m2_2 0000000# Wait for data set-up 75 ns
---- m2_2 m2_3 0000000
---- m2_3 m2_4 0000000
---- m2_4 m2_5 0001000# Write RAM
---- m2_5 m2_6 0001000
---- m2_6 w1 0000000# Settle down
---- w1 w2 0001000
---- w2 w3 0010000# Reset sc
---- w3 w4 0010000
---- w4 w5 0010000
---- w5 w6 0010000
---- w6 w7 0010000
---- w7 w8 0010000
---- w8 w9 0010000
---- w9 w10 0010000
---- w10 w11 0010000
---- w11 w12 0010000
---- w12 w13 0010000
---- w13 w14 0010000
---- w14 w15 0010000
---- w15 w16 0010000
---- w16 w17 0010000
---- w17 w18 0010000
---- w18 w19 0010000
---- w19 w20 0010000
---- w20 w21 0010000
---- w21 w22 0010000
---- w22 w23 0010000
---- w23 w24 0010000
---- w24 w25 0010000
---- w25 m4 0010000
---- m3b m3c 0010000
---- m4 m4a 0010001# Increment mpx-counter
---- m4a m4b 0010000# Settle down.
---0 m4b la1 0010000# Check max count
---1 m4b m4c 0010000
---- m4c m4d 0000000# Start last Conversion
---- m4d m4e 0000000# Wait for rdy to go low.
---- m4e m4f 0000000
---- m4f m4g 0000000
---- m4g m4h 0000000
---- m4h m5 0000000
--0- m5 m5 0000000# Wait for rdy to go high
--1- m5 m6_1 0000000
---- m6_1 m6_2 0000000# Wait for data set-up
---- m6_2 m6_3 0000000
---- m6_3 m6_4 0000000
---- m6_4 m6b 0001000# Write last data
---- m6b m7 0001000
---- m7 m7b 0000000# Settle down
---- m7b m8 0000010# Reset mpx
-0-- m8 m8 1010000# wait for mon_token, set mon_rq
-1-- m8 se0 1010000
######################### Send out monitor data  ################################
```

```
---- se0 se1 0010100# Load PSC
---- se1 se2 0010000
---- se2 se3 0010000
---- se3 se4 0010000
---- se4 se5 0010000
---- se5 se6 0010000
---- se6 se7 0010000
---- se7 se8 0010000
---- se8 se9 0010000
---- se9 se10 0010001# Increment mpx-counter
---- se10 se11 0010000# Settle down
---0 se11 se0 0010000# Check max count
---1 se11 se12 0010000
---- se12 se13 0010100# Load
---- se13 se14 0010000
---- se14 se15 0010000
---- se15 se16 0010000
---- se16 se17 0010000
---- se17 se18 0010000
---- se18 se19 0010000
---- se19 se20 0010000
---- se20 se21 0010000
---- se21 se22 0010000
---- se22 se23 0010000
---- se23 se24 0110000# Mon_tok_bk on 3:nd last data (2:nd last data on output)
---- se24 wmi0 0010000
```

Appendix C. . Block diagram of sub modules in MONROC

Fig. C-1. Parallel to Serial Converter

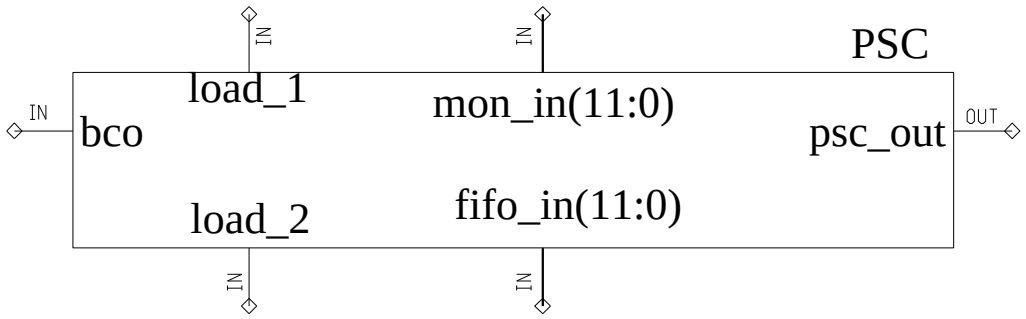


Fig. C-2. Serial To Parallel converter

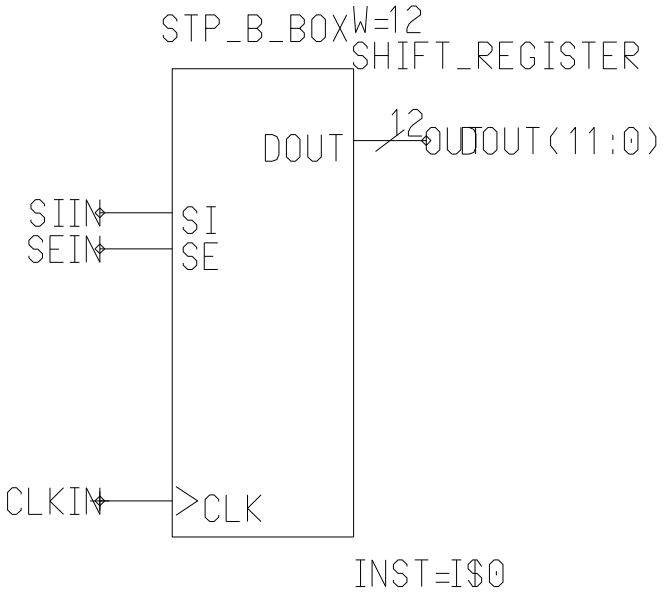


Fig. C-3. First In First Out memory for L1_trigger

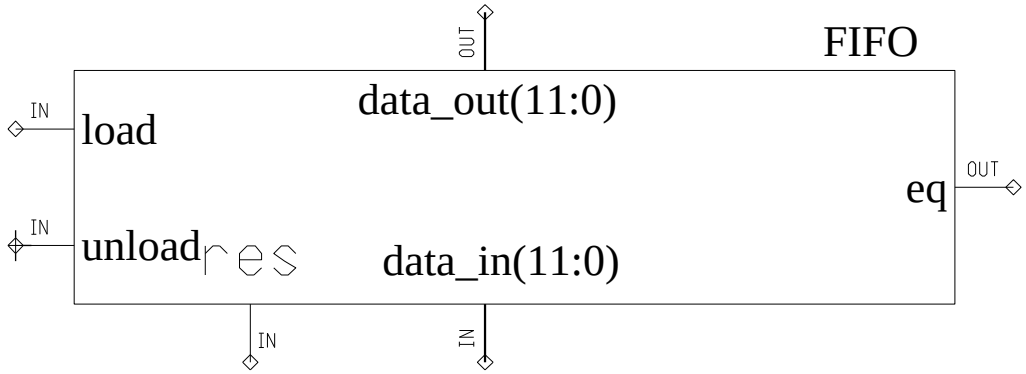
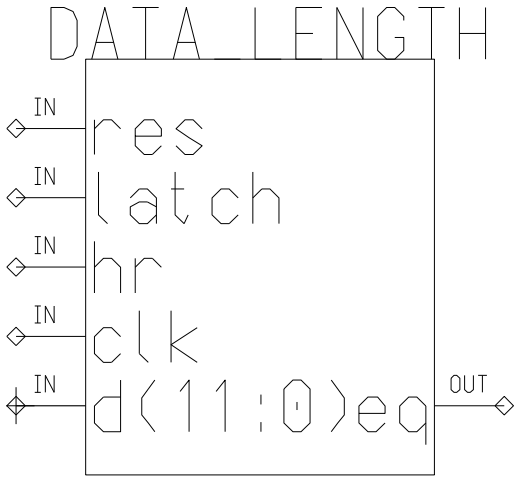


Fig. C-4. Data Length WCL (Word Compare Latch)



. Block diagram of sub modules in MONROC

. Block diagram of sub modules in MONROC

. Block diagram of sub modules in MONROC