The ATLAS SemiConductor Tracker Endcap

Peeters, S.J.M.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Appendix A

Details of the BPM-encoder module

For future reference and maintenance purposes, some of the details of the NIKHEF biphase-modulation unit are reproduced here.

The following figures are included:

- An overview of the NIKHEF BPM encoder (Figure A.1);
- A detailed schematic of the roboclock (Figure A.2);
- Details on several chips, shared between several units (Figure A.2);
- Details on the VCSEL-driver circuit (Figure A.2);
- A schematic of the GAL programming (Figure A.6 and A.7);
- Software definitions of the individual GAL blocks (Figure A.8)

The BPM-encoder encodes the clock and commands for one SCT module such that these can both be transferred via a single optical link. The overall functionality is described in more detail in Section 5.2.1.2.2.
Figure A.1: Overview of the NIKHEF BPM encoder

Details of the BPM-encoder module
Figure A.2: The roboclock.
Details of the BPM-encoder module

Figure A.3: LVDS drivers and buffers.
Figure A.4: TTL-NIM converters.
Details of the BPM-encoder module

Figure A.5: VCSEL driver.
Figure A.6: Block diagram of the software loaded into the Gate-Array Logic.

Toggle when Clock = '1' OR when Data was '1' during rising edge Clock.

Figure A.7: Block diagram of the BPM block.
Figure A.8: Software definitions of the individual blocks loaded into the Gate-Array Logic.

```vhdl
-- Entity declaration of 'BDG'.
entity BDG is
port(
    C1K : in std_logic;
    T : in std_logic;
    Q : out std_logic;
    Ret_n : in std_logic);
end BDG;
architecture 'A1' of BDG is
begin
    process (C1K, Ret_n)
    begin
        if Ret_n = '0' then
            Q <= '0';
        elsif Rising Edge(C1K) then
            Q <= C;
        end if;
    end process;
end BDG;
architecture 'A0' of BDG is
begin
    process (C1K, Ret_n)
    begin
        if Ret_n = '0' then
            Q <= '0';
        elsif Rising Edge(C1K) then
            Q <= C;
        end if;
    end process;
end BDG;
architecture 'A0' of BDG is
begin
    process (C1K, Ret_n)
    begin
        if Ret_n = '0' then
            Q <= '0';
        elsif Rising Edge(C1K) then
            Q <= C;
        end if;
    end process;
end BDG;
```

Details of the BPM-encoder module

```vhdl
-- Entity declaration of 'PER'.
library ieee;
use ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;
entity PER is
    port(
        C1K : in std_logic;
        T : in std_logic;
        Q : out std_logic);
end entity PER;
architecture 'A1' of PER is
begin
    process (C1K, Ret_n)
    begin
        if Ret_n = '0' then
            Q <= '0';
        elsif Rising Edge(C1K) then
            Q <= C;
        end if;
    end process;
end PER;
architecture 'A0' of PER is
begin
    process (C1K, Ret_n)
    begin
        if Ret_n = '0' then
            Q <= '0';
        elsif Rising Edge(C1K) then
            Q <= C;
        end if;
    end process;
end PER;
architecture 'A0' of PER is
begin
    process (C1K, Ret_n)
    begin
        if Ret_n = '0' then
            Q <= '0';
        elsif Rising Edge(C1K) then
            Q <= C;
        end if;
    end process;
end PER;
```

Figure A.8: Software definitions of the individual blocks loaded into the Gate-Array Logic.

```vhdl
-- Architecture 'A0' of 'BDG'.
architecture 'A0' of 'BDG' is
begin
    process (C1K, Ret_n)
    begin
        if Ret_n = '0' then
            Q <= '0';
        elsif Rising Edge(C1K) then
            Q <= C;
        end if;
    end process;
end architecture 'A0' of 'BDG';
```

Figure A.8: Software definitions of the individual blocks loaded into the Gate-Array Logic.

```vhdl
-- Architecture 'A0' of 'BDG'.
architecture 'A0' of 'BDG' is
begin
    process (C1K, Ret_n)
    begin
        if Ret_n = '0' then
            Q <= '0';
        elsif Rising Edge(C1K) then
            Q <= C;
        end if;
    end process;
end architecture 'A0' of 'BDG';
```