DDR configuration tool for Microchip platforms

Technical Note

CONFIDENTIAL

TNxxxx
Rev. 0 (master@)

TERMS and ABBREVIATIONS

```
HIF                Host Interface
ODT                On-Die Termination
PUB                PHY utility block
SoC                System on a Chip
```

Requirements

The tools are developed using the `ruby` scripting language, which must be available.

The scripts may run under both `Linux` or `Windows`. The `Windows` environment may require installing the `WSL2` package to emulate `Linux`.

Supported platforms

This tool is supported form the following target SoC platforms:

- sparx5 (ARMv8 A53 dual-core)
- lan966x (ARMV7 A7 single-core)
- lan969x (ARMv8 A53 single-core)

Introduction

This software package contain a set of tools that can create DDR configurations specific to a target platform and board design. A board design defines the physical DDR memory system, such that the DDR controller needs a carefully crafted configuration to ensure a stable and effective operation.

The DDR controller supported is the *Synopsis uMCTL2* DesignWare component, accompanied with a *Synopsis PHY Utility Block*.

The supported platforms all use DDR3 or DDR3 + DDR4.

Workflow

The tool works by accepting a *configuration profile* as input, which can be transformed into a compact representation of the many configuration options the target, **specifically** generated for a given platform and board.

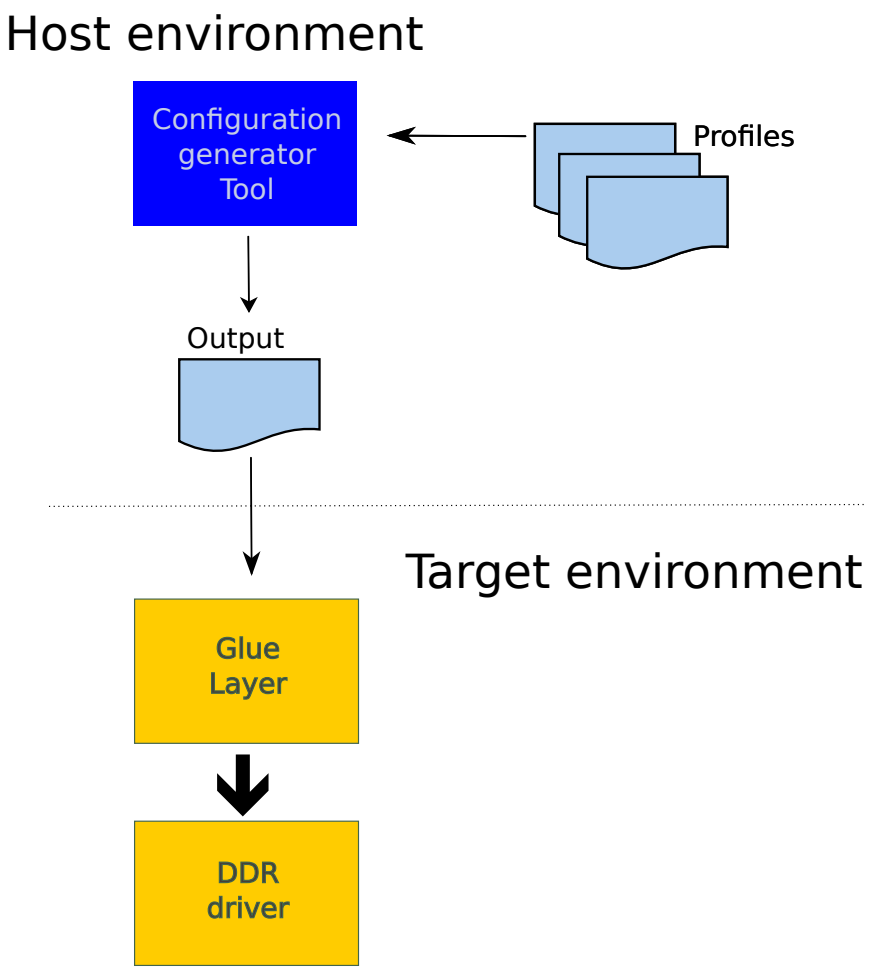
The output configuration may be represented in different formats, depending on the target platform.

The different formats are:

format	platform	comments
source	lan966x	The configuration is output to "C" code, and compiled into the target firmware.
devicetree	lan969x, sparx5	The configuration is output to device-tree format, which is added to a a target image (as a separate entity).
yaml	-none-	The configuration is stored in YAML format. The format can be used for documentation purposes as well as for comparing with other configurations.

The output representation is dictated by the target implementation of the actual target system DDR driver operating environment. Specifically, 'device tree' support may not be available on a given target (Example: `lan966x`).

The workflow of working with the DDR configuration is illustrated below:



Configuration parameters

At the top level, the following user-level parameters are available:

Keyword	Type	Description
`platform`	`sparx5 | lan966x | lan969x`	The name of the target platform SoC
`clock_speed`	`integer`	The clock speed of the DDR data bus. See later.

Keyword	Type	Description
`mem_size`	`integer`	Total Memory Size (in Mbytes)
`mem_type`	`DDR3 | DDR4`	The DDR memory type of the equipped DDR devices.
`device_bus_width`	`x8 | x16`	Bus width of the equipped DDR devices. `x8` support is experimental.
`active_ranks`	`integer`	The number of (active) ranks in the memory topology.
`ecc_mode`	`0 | 4`	The ECC mode to employ. ECC mode `0` implies ECC is disabled, mode `4` is "sec/ded over 1 beat"
`mem_profile`	`string`	The filename of a file defining DDR memory address generation. See later.
`_2T_mode`	`boolean`	Whether `2T` timing should be used
`board`	`string`	The filename of file containing board specific tuning parameters. See later.

Clock speed.

The *clock_speed* affects a lot of the calculated parameters for the memory controller.

Whether or not the clock speed can be changed or not, depends on the target system (and the actual driver implementation).

The current state of the clock speed is described below.

Platform	Supported clocks	Notes
`sparx5`	`2500 | 2000 | 1667 | 1250`	NB: DDR4 will only work at 1667 Mhz
`lan966x`	`1200`	Fixed clock
`lan969x`	`2400`	Fixed clock

Memory profile parameter *mem_profile*

In order to define the way HIF addresses are used to select ranks, groups and DDR devices, a separate YAML file is used. This configuration file may be used by several configurations (board designs) using similar physical DDR topology.

The *mem_profile* file defines the so-called "address map" registers (and sub-fields). An example is given below.

Note: The registers and sub-fields vary a little between platforms, so not all registers apply to all supported platforms. Refer to [1], section "2.11: Address Mapper".

Address map definition example

```
addrmap0:
  ADDRMAP_CS_BIT0: 31
addrmap1:
  ADDRMAP_BANK_B0: 24
  ADDRMAP_BANK_B1: 24
  ADDRMAP_BANK_B2: 24
addrmap2:
  ADDRMAP_COL_B2: 0
  ADDRMAP_COL_B3: 0
  ADDRMAP_COL_B4: 0
  ADDRMAP_COL_B5: 0
addrmap3:
  ADDRMAP_COL_B6: 0
  ADDRMAP_COL_B7: 0
  ADDRMAP_COL_B8: 0
  ADDRMAP_COL_B9: 0
addrmap4:
  ADDRMAP_COL_B10: 31
  ADDRMAP_COL_B11: 31
addrmap5:
  ADDRMAP_ROW_B0: 4
  ADDRMAP_ROW_B1: 4
  ADDRMAP_ROW_B2_10: 4
  ADDRMAP_ROW_B11: 4
addrmap6:
  ADDRMAP_ROW_B12: 4
  ADDRMAP_ROW_B13: 4
  ADDRMAP_ROW_B14: 4
  ADDRMAP_ROW_B15: 4
  LPDDR3_6GB_12GB: 0
addrmap7:
  ADDRMAP_ROW_B16: 15
  ADDRMAP_ROW_B17: 15
addrmap8:
  ADDRMAP_BG_B0: 63
  ADDRMAP_BG_B1: 63
```

Memory profile parameter *board*

In order to control memory settings relating to `ODT` and general board tuning, this file can be used to define **all** parameters where a specific is needed which is different from the default or by this tool calculated value. As such, this file can be used to override specific parameters.

Board file example (lan966x)

```
dfitmg0:
  - DFI_T_CTRL_DELAY: 4
  - DFI_RDDATA_USE_DFI_PHY_CLK: 0
  - DFI_T_RDDATA_EN: 3
  - DFI_WRDATA_USE_DFI_PHY_CLK: 0
  - DFI_TPHY_WRDATA: 1
  - DFI_TPHY_WRLAT: 2
mr1:
  - RTT_2: 1
```

All supported registers and sub-fields can be defined. See the full list of supported registers below.

Supported DDR configuration registers

The supported DDR configuration registers are a subset of the full UMCTL2 DDR controller registers. The registers supported in this tool have been identified to contain options that typically may need customization.

The configuration register set is currently different for the *lan966x* and the *lan969x*/*sparx5* driver, due to differences in the base IP version and IP configuration parameters.

bla bla bla bla

References

- [1] DesignWare Cores Enhanced Universal DDR Memory Controller (uMCTL2) Databook