External PHY Failover Proposal

Title	**External PHY Abstraction Interface**
Authors	**Broadcom**
Status	**Draft**
Type	
Created	**8/31/2020**
SAI-Version	**0.1**

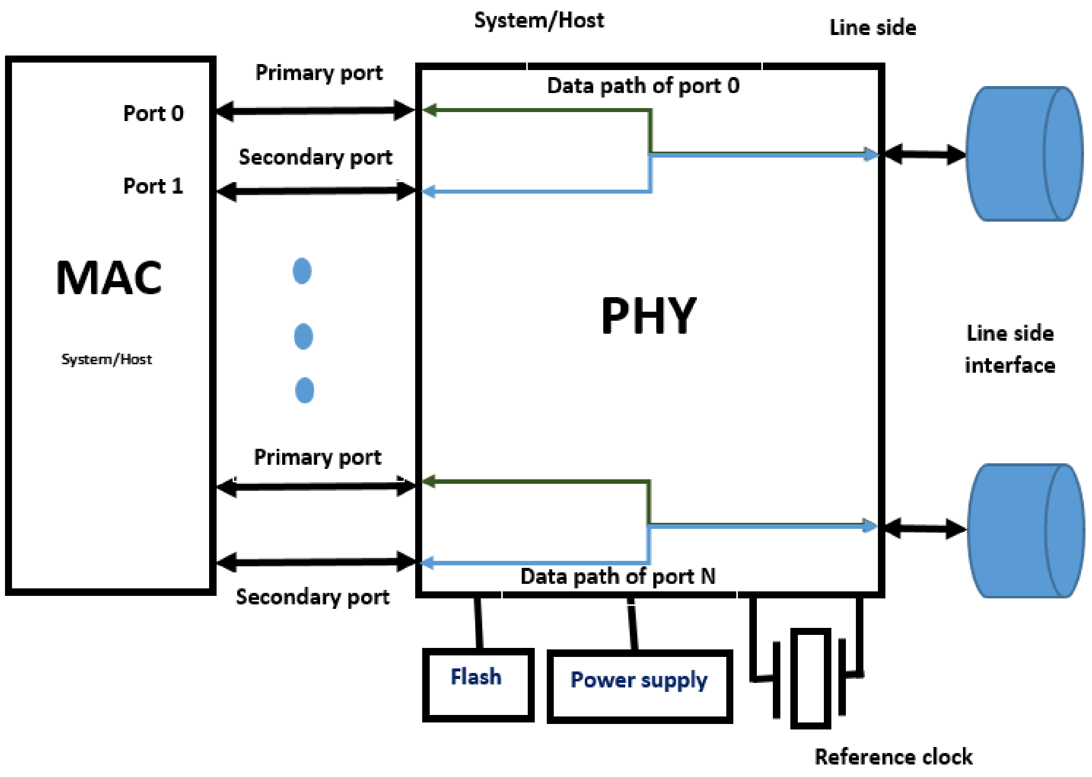
Contents

List of Changes

Version	Changes	Name	Date
0.1	Proposal for PAI Failover		8/31/2020

1 Overview

The purpose of this document is to describe PHY failover (FO) functionality and interface to manage failover functionality of the PHY. PHY is the connector between MAC (SerDes) and physical medium such as optical fiber or copper transceivers. PHY failover feature uses two physical port configurations: primary and secondary. This can be enabled on system/host or line/client side of a port. It provides the ability to switch from primary port to secondary port or vice versa without losing data/link.



In the above multi-port configuration, failover mode is configured on the system side of external PHY. Primary and secondary ports are physical connections between MAC and PHY. The FO configuration uses primary and secondary port connections on selected side (system side or line side).

The PHY failover mode enables a port to broadcast data on both primary and secondary ports. The MAC would be configured to receive the data on either primary or secondary ports. PHY failover active ingress port can be configured as either primary or secondary port. When the link between MAC and PHY goes down on primary port, user can switch to secondary port.

1.1 Failover Configuration on PHY

Failover feature can be configured on all ports of the PHY. Failover ports can operate in either hitless (the ability to switch over port without losing link) or non-hitless mode.

Additions to saiport.h

```
    /**
     * @brief Attribute data for #SAI_PORT_CONNECTOR_ATTR_FAILOVER_MODE
     * Used for Failover mode configuration on port
     */
     typedef enum _sai_port_connector_failover_mode_e
     {
        /** Failover mode disable */
        SAI_PORT_CONNECTOR_FAILOVER_MODE_DISABLE,

        /** Configure Failover mode on primary port */
        SAI_PORT_CONNECTOR_FAILOVER_MODE_PRIMARY,

        /** Configure Failover mode on secondary port */
        SAI_PORT_CONNECTOR_FAILOVER_MODE_SECONDARY
    } sai_port_connector_failover_mode_t;

    /**
     * @brief Configure the failover mode on port
     *
     * @type sai_port_connector_failover_mode_t
     * @flags CREATE_AND_SET
     * @default SAI_PORT_CONNECTOR_FAILOVER_MODE_DISABLE
     */
    SAI_PORT_CONNECTOR_ATTR_FAILOVER_MODE,

    /**
     * @brief System Side Port ID
     *
     * @type sai_object_id_t
     * @flags CREATE_ONLY
     * @objects SAI_OBJECT_TYPE_PORT
     */
    SAI_PORT_CONNECTOR_ATTR_SYSTEM_SIDE_FAILOVER_PORT_ID,

    /**
     * @brief Line Side Port ID
     *
     * @type sai_object_id_t
     * @flags CREATE_ONLY
     * @objects SAI_OBJECT_TYPE_PORT
    */
    SAI_PORT_CONNECTOR_ATTR_LINE_SIDE_FAILOVER_PORT_ID,
```

1.2 Additions to saiswitch.h

```
/**
 * @brief Attribute data for #SAI_SWITCH_ATTR_FAILOVER_CONFIG_MODE
 * Used for Failover configuration mode
 * In case of primary port failure, hitless enables the switch over to secondary
 * port without losing link. It allows uninterrupted data transmission
 */
typedef enum _sai_switch_failover_config_mode_t
{

    /** Ports are configured but do not operate in hitless */
    SAI_SWITCH_FAILOVER_CONFIG_MODE_NO_HITLESS,

    /** Ports are configured and operate in hitless */
    SAI_SWITCH_FAILOVER_CONFIG_MODE_HITLESS
} sai_switch_failover_config_mode_t;

    /**
     * @brief Failover configuration mode
     *
     * @type sai_switch_failover_config_mode_t
     * @flags CREATE_AND_SET
     * @default SAI_SWITCH_FAILOVER_CONFIG_MODE_NO_HITLESS
     */
    SAI_SWITCH_ATTR_FAILOVER_CONFIG_MODE,

    /**
     * @brief Query for Failover mode support
     *
     * @type bool
     * @flags READ_ONLY
     */
    SAI_SWITCH_ATTR_SUPPORTED_FAILOVER_MODE,
```

2 Configuration Example

```
Following example shows how to setup failover configuration on a system side port. In this
example a 16 lane PHY is used.

    /* Create System side, Line side, and System Side Failover ports */
    sys_attr[0].value.u32list.list = sys_lane_list;
    line_attr[0].value.u32list.list = line_lane_list;
    failover_attr[0].value.u32list.list = failover_lane_list;
    line_attr[1].id = sys_attr[1].id = SAI_PORT_ATTR_SPEED;
    line_attr[1].value.u32= sys_attr[1].value.u32 = 100000;
    line_attr[2].id = sys_attr[2].id = SAI_PORT_ATTR_INTERFACE_TYPE;
    line_attr[2].value.u32 = sys_attr[2].value.u32 = SAI_PORT_INTERFACE_TYPE_KR;
    line_attr[3].id = sys_attr[3].id = SAI_PORT_ATTR_FEC_MODE;
    line_attr[3].value.u32 = sys_attr[3].value.u32 = SAI_PORT_FEC_MODE_RS;
    line_attr[4].id = sys_attr[4].id = SAI_PORT_ATTR_LINK_TRAINING_ENABLE;
    line_attr[4].value.booldata = sys_attr[4].value.booldata = 1;
    line_attr[5].id = sys_attr[5].id = SAI_PORT_ATTR_ADMIN_STATE;
    line_attr[5].value.booldata = sys_attr[5].value.booldata = 1;


    for (phy_index = 0; phy_index < PAI_MAX_PHY; phy_index ++) {
        for (index = 0; index < 2; index ++) {
             line_lane_list[index] = (phy_index*16)+ port_index + index;
             sys_lane_list[index] = (phy_index*16)+16 + port_index + index;
             failover_lane_list[index] = (phy_index*16)+20 + port_index + index;
        }

        rv = pai_port_apis_ptr->create_port(&sys_port_id[phy_index][port_index],
                switch_id[phy_index], COUNTOF(sys_attr), sys_attr);
        if (SAI_STATUS_SUCCESS != rv)  {
            printf("System side port creation failed with error:%d\n", rv);
            return rv;
        }

        rv = pai_port_apis_ptr->create_port(&fail_over_port_id[phy_index][port_index],
                switch_id[phy_index], COUNTOF(fail_over_attr), fail_over_attr);
        if (SAI_STATUS_SUCCESS != rv)  {
            printf("System side failover port creation failed with error:%d\n", rv);
            return rv;
        }

        rv = pai_port_apis_ptr->create_port(&line_port_id[phy_index][port_index],
                 switch_id[phy_index], COUNTOF(line_attr), line_attr);
        if (SAI_STATUS_SUCCESS != rv)  {
            printf("Line side port creation failed with error:%d\n", rv);
            return rv;
        }

        /* Populate failover port IDs */
        port_conn[0].id = SAI_PORT_CONNECTOR_ATTR_SYSTEM_SIDE_PORT_ID;
        port_conn[0].value.oid = sys_port_id[phy_index][port_index];
        port_conn[1].id = SAI_PORT_CONNECTOR_ATTR_LINE_SIDE_PORT_ID;
        port_conn[1].value.oid = line_port_id[phy_index][port_index];
        port_conn[2].id = SAI_PORT_CONNECTOR_ATTR_SYSTEM_SIDE_FAILOVER_PORT_ID;
        port_conn[2].value.oid = fail_over_port_id[phy_index][port_index];
        port_conn[3].id = SAI_PORT_CONNECTOR_ATTR_FAILOVER_MODE;
        port_conn[3].value.u32 = SAI_PORT_CONNECTOR_FAILOVER_MODE_PRIMARY;

        rv = pai_port_apis_ptr->create_port_connector(&port_conn_id[phy_index][port_index],
                switch_id[phy_index], COUNTOF(port_conn), port_conn);
        if (SAI_STATUS_SUCCESS != rv)  {
            printf("Port connector creation failed with error: %d\n", rv);
            return rv;
        }
        printf("Port connector: 0x%lx.\n", port_conn_id[phy_index][port_index]);
    }

    /* Get current state of failover mode */
    for (port_index = 0; port_index < 1; port_index ++) {
        for (phy_index = 0; phy_index < PAI_MAX_PHY; phy_index ++) {
            attr_count = 1;
            memset(&port_conn_attr_get, 0, sizeof(port_conn_attr_get));
            port_conn_attr_get[0].id = SAI_PORT_CONNECTOR_ATTR_FAILOVER_MODE;

            rv = pai_port_apis_ptr->get_port_connector_attribute(
                 sys_port_id[phy_index][port_index], attr_count, port_conn_attr_get);
            if (SAI_STATUS_SUCCESS != rv)  {
                printf("Get port connector attribute failed with error: %d\n", rv);
                return rv;
            }
            printf("PAI Port Connector failover mode get attribute values :%d\n",
                    port_conn_attr_get[0].value.u32);
        }
    }


    /* Configure primary failover port */
    for (port_index = 0; port_index < 1; port_index ++) {
        for (phy_index = 0; phy_index < PAI_MAX_PHY; phy_index ++) {
            port_conn_attr_set.id = SAI_PORT_CONNECTOR_ATTR_FAILOVER_MODE;
            port_conn_attr_set.value.u32 = SAI_PORT_CONNECTOR_FAILOVER_MODE_PRIMARY;

            rv = pai_port_apis_ptr->set_port_connector_attribute(
                 sys_port_id[phy_index][port_index], &port_conn_attr_set);
            if (SAI_STATUS_SUCCESS != rv)  {
                printf("Set Port Connector Attribute failed with error: %d\n", rv);
                return rv;
            }
        }
    }
```

```
     /* Failover switch attribute configuration applicable for all ports on PHY with no
hitless) */

    if (pai_switch_apis_ptr->set_switch_attribute) {
        sai_attribute_t sai_set_attr;
        sai_set_attr.id = SAI_SWITCH_ATTR_FAILOVER_CONFIG_MODE;
        sai_set_attr.value.u32 = SAI_SWITCH_FAILOVER_CONFIG_MODE_NO_HITLESS;
        rv = pai_switch_apis_ptr->set_switch_attribute(switch_id[0], &sai_set_attr);
        if (SAI_STATUS_SUCCESS != rv) {
            printf("Set switch attribute failed: Error: %d\n", rv);
            return rv;
        }
    }
    if (pai_switch_apis_ptr->get_switch_attribute) {
        for (switch_index = 0;switch_index < PAI_MAX_PHY; switch_index ++) {
            sai_get_attr.id = SAI_SWITCH_ATTR_FAILOVER_CONFIG_MODE;
            rv = pai_switch_apis_ptr->get_switch_attribute(switch_id[switch_index],
                     attr_count, &sai_get_attr);
            if (SAI_STATUS_SUCCESS != rv) {
                printf("Get switch attribute failed: Error: %d\n", rv);
                return rv;
            }
            printf("failover config mode: %d\n", sai_get_attr.value.u32);
        }
    }
```