OI14: Crossing-Aware Channel Routing for Integrated Optics

Christopher Condrat, Priyank Kalla and Steve Blair

***Abstract*—Increasing scope and applications of integrated optics necessitates the development of automated techniques for physical design of optical systems. A key area of integrated optic design is waveguide routing, which currently lacks the level of automation found in VLSI design. Unlike VLSI, where signal nets are routed with metal layers and vias, integrated optics is a planar technology and lacks the inherent signal restoration capabilities of static-CMOS. Waveguides suffer signal loss due to planar (perpendicular) waveguide crossings and also from sharp bends. Therefore, in contrast to area or wire length, *signal loss minimization* — as a function of waveguide crossings and bends — is a primary objective of any routing solution. For many applications, our studies show that waveguide routing problems can be suitably formulated as *planar channel routing*.**

This paper investigates channel routing for integrated optical waveguides fabricated in a planar substrate. We present routing techniques where crossings, bends, and area are accounted for in an integrated solution. Two distinct channel routing techniques are presented: 1) a new channel router based on *net sorting* and utilizing non-Manhattan routing grids, and 2) a router based on crossing-aware graph-constrained track assignment that also exploits waveguide curves to reduce track utilization. Both techniques are crossing-minimal, and also are constrained suitably to reduce bend loss and area. We compare and evaluate the performance of our channel routers on a number of optical design benchmarks.

***Index Terms*—Integrated optics, channel routing, crossing minimization, photonic waveguides, signal loss**

1. Introduction

Recent breakthroughs in silicon-based integrated optics — *Silicon Photonics* — are establishing the viability of silicon for integrated optics. The use of silicon enables fabs to leverage already existing and mature silicon processes and infrastructure for optical device fabrication as well as integration for electro-optical systems. Investment in Si-photonics integration is significant [1], [2]; also significant are the open foundary initiatives and developmental programs such as the OPSIS framework [3]. These developments are enabling applications far beyond traditional roles of optics in communications — such as *optical routing and photonic networks-on-chips* [4], *signal processing* [5], and also *optical digital logic* [6], [7], *quantum and reversible computation* [8]–[10].

As the availability and applications of integrated optics expand, the need for automated design space exploration, optimization, and physical synthesis of integrated electro-optical systems is also beginning to appear. For this reason, the Electronic Design Automation (EDA) community is investigating how automatic design space exploration techniques can be adapted to the photonics domain [6], [11]–[15]. This paper also takes a step in this direction and presents a methodology and solutions for detailed routing of integrated optical waveguides. In particular, we show that the detailed routing problem manifests itself as a *channel routing* problem, where (Silicon) optical waveguides are fabricated on a planar substrate and are connected to devices at the ends of the channel[1]. Planar routes require waveguides to bend (curve) and cross each other — causing loss of signal power. Channel routing techniques are therefore needed that minimize waveguide crossings and bends. This paper presents two techniques: 1) a channel router based on sorting, and 2) a channel router based on crossing-aware, graph-constraint track-assignment. Both routers minimize signal loss as a function of waveguide crossings and bends within the channel, while also reducing area.

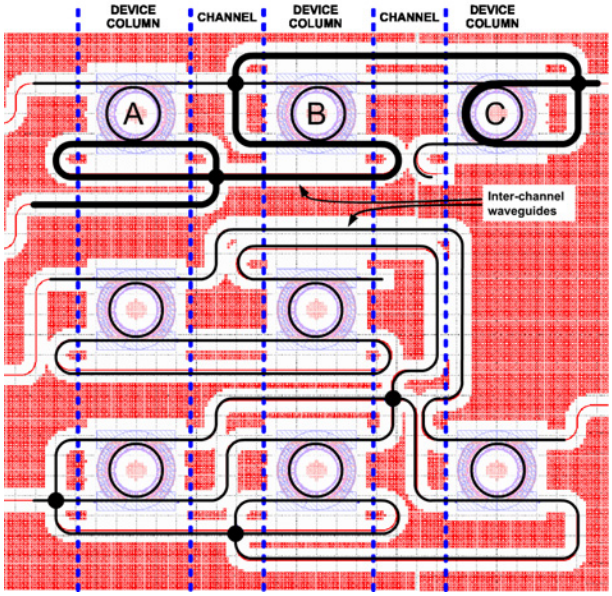


Fig. 1: Routing channels of optical GDS layout

A. Optical Routing Problem Formulation and Objectives

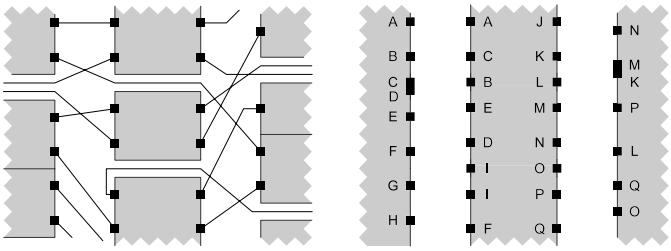
The main motivation for solving this problem stems from physical design of integrated optical logic circuits [6], [7],

This research was funded in part by a sub-contract to AFOSR grant FA9550-09-1-0661.

Christopher Condrat (chris@g6net.com), Priyank Kalla (kalla@ece.utah.edu) and Steve Blair (blair@ece.utah.edu) are with Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT, USA

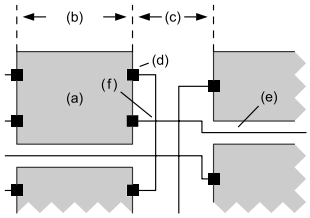
[1]In the VLSI domain, channel routing is no more a topic of extensive research investigations due to the availability of a large number of metal layers and over-the-cell routing. This paper revisits channel routing specifically for optical technology, which introduces new optimization criteria not addressed by VLSI channel routers.

[10], [16]. Such circuits comprise a set of pre-designed optical devices — such as modulators, switches, splitters and detectors — placed on a planar substrate, and connected together with waveguides. Consider the optical network depicted in Fig.1. Eight (8) ring resonators are arranged into columns by a device placer such as to minimize area as well as routing complexity. The column arrangement induces the presence of vertical routing regions between device columns denoted as **channels**, with device connection-points, denoted as **ports**, facing the channels. Inter-channel waveguides are used to allow routes between devices in other channels.



(a) Placement of devices with global routes

(b) Resulting channel for detailed routing



(c) Detailed routing formulation

Fig. 2: Channel routing for an optical device network

In general, for a layout such as Fig.2(c), pre-designed optical devices are represented as rectangular blocks (a) that are arranged (placed) in fixed-width columns (b). Such a placement gives rise to vertical routing channels (c), which are routing regions that separate the placed devices. Waveguides are routed between devices at ports (d) that face the channels. For routing between ports in different columns, connections are made to inter-channel waveguide, as depicted in (e). Due to the planar nature of routing, waveguides may only cross each other perpendicularly (f) — at the cost of signal loss.

The waveguide connecting two ports is denoted as a **net** and comprises a single route with no signal sharing (fanout). Signal sharing is explicitly provided by pre-designed *waveguide splitter devices*; these devices are treated as placed, 3-port, pre-designed optical devices, with ports for routing. Therefore, our methodology renders every net a **two-terminal net** within the channel.

As a relatively generic routing formulation, waveguide channel routing is applicable to many optical network architectures, especially as most assume a single, planar optical routing layer. This includes optical interconnect networks incorporating wavelength division multiplexing (WDM), as the routing does not affect the multi-wavelength, channel-carrying capabilities of the waveguides. Many topologies lend themselves to channel routing. For example, in [17], various WDM network topologies are explored, many of which required column-based layouts of modulators, filters, and endpoints that could produce channels. Integration-related aspects such as access to laser sources and the proximity of devices to heat sources are also a function of device placement, and may still give rise to channels, even if not specifically designed for them. Another aspect of on-chip photonic integration is timing. The routing itself does not have a great effect on the timing of the overall system. This is especially true, given that the speed of modulation and conversion will dominate the overall timing budget as compared to the transmission delay of the routed interconnects.

Such aspects of optical network design are important; however, this paper is concerned with channel routing and not with device placement or data conversion. It is assumed that a (column-based) placement of optical devices is already given, along with the general routing path/topology of optical signals. This, subsequently, gives rise to a channel routing problem — such as the one depicted in Fig.2 — which we solve while minimizing signal loss.

Paper Organization: We begin with an overview of signal loss mechanisms on which optimizations are based. Following is a description of the contributions of this work, along with previous work in integrated optic design. We then describe our investigations into existing a non-Manhattan grid sorting routers, the limitations of which lead to the development of our own sorting-based router. After describing our sorting-based router, we investigate a second channel router: a crossing-aware, left-edge style router. We then compare and evaluate the performance of our channel routers on a number of optical design benchmarks. The paper concludes with a final analysis of the results.

B. Signal Loss

We identify signal loss as the primary guiding metric in integrated optics routing. All devices, including bulk waveguides, have insertion losses measured in decibels (dB). These losses are pre-characterized through device analysis (e.g. FDTD modeling and simulation) and technology parameters.

In terms of planar routing, we identify the following loss mechanisms:

- **Waveguide crossings [0.1–0.2 dB / crossing]** Per-crossing losses are on the order of 0.1–0.2dB per crossing [18], [19], affecting both crossing waveguides.
- **Waveguide bends [0.001–0.3 dB / bend]** Losses depend on inherent waveguide properties (materials, geometry, etc.), radius of curvature of the bend, and surface roughness due to fabrication [20]–[22].
- **Bulk waveguides [0.01–2 dB / cm]** As these losses are extremely low (dB per *centimeter*, e.g. 0.03dB/cm [23]), we consider bulk waveguides essentially *lossless*.

Optimization Objective: The primary optimization objective in our routing formulation is *signal loss* minimization. Within the channel, this is achieved by: 1) minimization of the total number of waveguide crossings; and 2) minimization of the number of waveguide bends. Minimization of the number of tracks (channel height) is the subsequent secondary objective.

We optimize for the *total signal loss within the channel* due to optical feedback within the system. For example,

consider the 1-bit full-adder circuit implemented using ring-resonator-based switches in silicon photonics, depicted in Fig.1. We designed this circuit and fabricated it through OpSIS [3]; the design is currently under characterization. A signal may be routed such that it enters a given channel multiple times and may cross multiple other nets; this is depicted in the highlighted signal path. Therefore, instead of minimizing losses on a per-net basis, we minimize for *total* losses within a channel.

C. Contributions of this work

This paper presents methods for channel routing of integrated optical waveguides fabricated on a planar substrate. Two distinct crossing-aware channel routing techniques are presented: 1) a sorting-based router based on a non-Manhattan routing grid, and 2) a left-edge style router utilizing crossing-aware graph-constraints. Both techniques are crossing-minimal, and are constrained in a technology-suitable fashion to reduce bend-loss and area (number of tracks).

Our sorting-based channel routing technique utilizes a non-Manhattan routing grid and positional net sorting. We draw inspirations from sorting-based routers [24], [25]. These routers have useful properties of being *minimal in terms of crossings*. In our investigations, however, the original sort-router formulation suffers from a number of unaddressed limitations and detrimental side-effects that make it impractical. We show that there are fundamental flaws in the way the original swap/sort-routing channel problems are encoded, requiring excessive area, introducing more waveguide bends, and even affecting the original problem specification.

This leads us to develop our own crossing-minimal, bend-reducing, sorting-based router. We overcome the problems of the original formulation by: 1) performing routing separately on both sides of the channel; 2) constraining the formulation to avoid unnecessary bends and enable routes to better utilize the routing grid. As a result, our router not only retains minimal crossings, but further minimizes the number of waveguide bends. Track utilization is also greatly improved over the original technique.

We also present a channel router based on traditional, *left-edge style* constraint-graph track assignments [26]–[28]. For such channel routing, we show how 1) crossing constraints are incorporated into the underlying constraint model enabling routing solutions to be both crossing-minimal; and 2) exploit the physical realization of waveguide curves for improved track utilization. For the former, the concept of *pin-rotation* is introduced as a means to determine whether nets require crossings. For the latter, we utilize the geometric properties of waveguide curves to enable knock-knees (KK) to facilitate track sharing, reducing overall track height.

Our channel routing techniques are then applied to a number of optical waveguide routing problems derived from photonic designs. We evaluate and compare the techniques with respect to each other in terms of crossings, bend-loss, and channel-height.

2. Previous Work

State-of-the-Art in Photonics Design Automation: One of the main focus of current investigations is toward *architectural explorations* for photonic interconnection networks in multi-core processor systems [4], [12], [29]–[31]. At the functional/logic-design level, there have been investigations into use of optical components as building-blocks, connected by waveguides, to design optical computing systems [6], [8], [10], [16], [32], [33]. High-level synthesis, using technology-mapping with a library of optical device building-blocks, has also been presented [11]. The focus of these works is on architectural and functional analysis and optimization; physical design and fabrication details are beyond the scope of such works.

At the much lower (physical) level, [13] demonstrates a *full-custom layout* of photonic structures using a commercial CMOS-based layout editor (Cadence Design Systems Virtuoso). Waveguide curves are discretized at a fine level into rectangular geometry, enabling waveguides to be represented in a format that traditional foundries accept. This methodology is significant in that it provides a building-block pathway for producing foundry-ready layouts and masks for non-Manhattan device geometries (rings, arcs, waveguide curvature). However, for such methodologies, design automation is essentially absent, and designs must be optimized manually. Similarly, commercially available photonics CAD suites [34], [35] provide frameworks for physical device design, analysis, mode solving, and simulation (FDTD) for performance analysis of optical design components. However, automated techniques for design space exploration during physical synthesis — *automated floorplanning, placement, waveguide routing while optimizing for physical parameters such as insertion-loss, bend-loss, phase coherence issues, etc.* — are not available.

Recently, [12], [14] present techniques for global optical interconnect synthesis. Such techniques analyze the routing problem at different levels of abstraction than the techniques presented in this work. Once global routing is performed, local routing is necessary to complete routing. At this routing level, a channel router may be utilized to ensure crossing-minimality, as well heuristically minimal bends. The work of [15] presents high-level, run-time calibration and reconfiguration techniques to reduce power usage in on-chip interconnect networks. The technique employs multiple redundant optical devices to which sub-channels can be remapped in cases where it would reduce overall tuning power.

In VLSI physical design, channel routing algorithms [26]–[28] are textbook knowledge [36], [37]. Crossing minimization in routing has been studied in the context of the crossing distribution problem (CDP) [38], [39]. The CDP is concerned with the distribution of a minimal set of crossings within a routing topology; this is performed through permutations of net orderings. In contrast, while our work also utilizes net orderings to ensure crossing minimality, the final routing is performed with the goal of reducing *signal loss* with respect to crossings and bends — not the *distribution* of crossings across a layout.

Channel routing with crossing minimization has also been studied in the context of QCA routing [40]. Track assignments

for multi-terminal nets induce varying numbers of crossings; therefore, [40] formulates crossing minimization, heuristically, as a weighted-minimum-feedback-edge-set problem. However, in the context of our problem — utilizing exclusively 2-terminal nets — we exactly minimize waveguide crossings, obviating the need for such an approach.

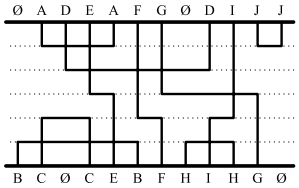


Fig. 3: Track-optimized channel solution

3. Non-Manhattan Grid, Sorting-based Routing

A channel routing problem is represented by net "pins" fixed to the top and bottom of a channel. The purpose of the router is to route all nets in the channel's routing region, while minimizing parameters such as area (channel height, number of tracks), signal delay (net-length), or signal loss. Fig.3 depicts a minimum track channel routing obtained by a left-edge router. In traditional VLSI channel routing, area and net-length are primary optimization goals. Channel routing also seeks to minimize other objectives, such as vias [41], and in the case of this paper, signal loss.

Manhattan-based (rectilinear) grids are traditionally employed in VLSI routing, dedicating layers specifically to horizontal or vertical spans for routing flexibility; non-Manhattan-based grids (e.g. octilinear) are rarely utilized. Integrated optics, however, is well suited to non-Manhattan-based routing grids. Such routing grids can suitably represent waveguide curves, and provide greater routing flexibility in the absence of multiple routing layers.

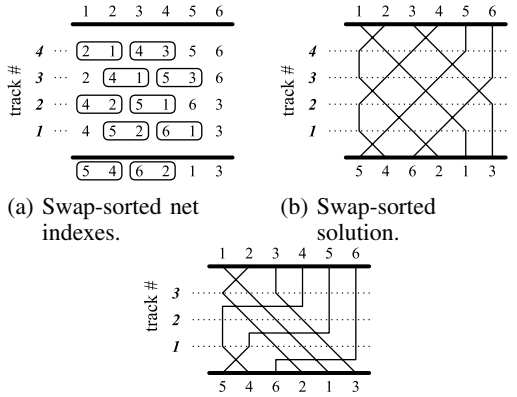


Fig. 4: Channel routing performed by sorting indexes. Circled indexes denote a pair that is reordered (sorted).

The work of [24], [25], also described in textbook [37], investigates a non-Manhattan grid channel router based on *sorting*. The nets of a channel are assigned numerical indexes, and routing is performed by sorting the nets in a finite number of permutations. The number of permutations performed represents the number of tracks utilized. Examples of this sorting-based routing are depicted in Fig.4.

Crossing minimality: In addition to utilizing non-Manhattan grids, sort-router's channel solutions are minimal in terms of the number of crossings. Crossing minimality results from the fact that [24]: 1) crossings only occur if nets are positioned out-of-order during sorting, and 2) once sorted, pairs of nets *never cross each other again during the sorting process.*

The flexibility of a non-Manhattan-based grid and crossing minimality makes sorting-based routing attractive for integrated optics. However, the original sorting-based channel routing solutions presented in [24], [25] have drawbacks that make them impractical. Below, we describe the limitations of the sort routers of [24], [25]; these limitations motivate the design of our own sorting-based channel router, specifically designed for integrated optics.

A. Sorting-based Channel Routing

Two sorting techniques are presented in [24], [25]: swap-sorting and bubble-sorting. The swap-based router swaps positions of pairs of adjacent nets if they are out-of-order. For example, in the bottom track of Fig.4(a), nets 5 and 4 are out-of-order and they swap columns in the transition to track 1; this is reflected in the swap depicted in Fig.4(b). A bubble-sort based technique can also be used, as depicted in Fig.4(c), allowing indexes to sort across multiple column positions. Bubble-sorting, however, causes nets to cross at non-perpendicular angles, and therefore is *unusable* for optical waveguide routing. *Our channel routing technique utilizes swap-based sorting for routing.*

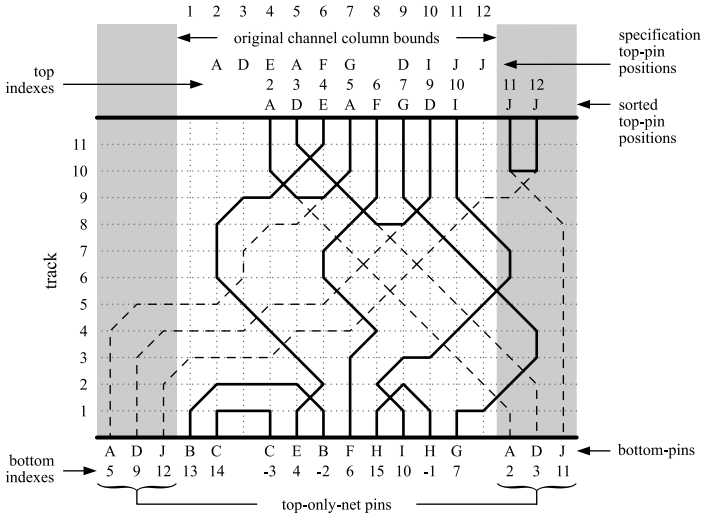


Fig. 5: A swap-router channel solution. Shaded region denotes columns outside the initial channel bounds.

B. Encoding Side-only Nets

The described channel problem setup assumes that all nets appearing on the bottom of a channel also appear on the top. However, most channel problem instances also incorporate nets with pins exclusively on one side of the

channel. Also, empty spaces *between* pins (i.e. "gaps"), or within the routing tracks are also not accounted for by the basic routing algorithm. We first formally define net types with respect to their pin locations, as well as the concept of empty spaces in the routing region:

Definition 3.1. [X-net, T-net, B-net, side-only nets] *A net with pins on both the top and bottom of a channel is an **X-net** (cross-over/shared net). Nets with pins exclusively on one side of a channel are denoted **side-only nets**: a **T-net** (top-only net) has pins on only the top side of the channel; a **B-net** (bottom-only net) is net with pins on only the bottom of the channel.*

Definition 3.2. [Gap] *A **gap** is an empty location in the routing grid, or an empty pin location. Gaps are not assigned sorting indexes, but may be routed over if unoccupied.*

Consider the channel depicted in Fig.5. We must assign net pins initial sorting indexes in order to route the channel nets. Observe nets E, B, and D. Net E is an X-net, and therefore the bottom pin of E is assigned the index corresponding to its top pin's index-position, in this case 4. Nets B and D are B-net and T-net types respectively; they have their pins exclusively on one side of the channel — *side-only nets*. The work of [24] provides a means for encoding T-nets and B-nets into the channel sorting problem:

B-net encoding: For a given B-net, the left-side pin is assigned a high-valued index, and the right-side pin is assigned a negative index. For example, in Fig.5, the left-pin of B is assigned index 13 — *a value greater than the number of channel columns* — and the right-pin is assigned -2. The result of these index assignments is that the high-valued index causes the route of the left-pin to sort to the right; the negative index causes the route of the right-pin to sort to the left. When the routes meet, the net is considered routed and the indexes are removed from the problem in subsequent tracks.

T-net encoding: T-nets do not exist at all on the bottom track, and therefore are added to the initial bottom sorting track as *additional columns ("virtual columns")* to the left and right of the original channel columns. In Fig.5, these are indicated by the shaded areas. The bottom pins of the T-nets are assigned indexes corresponding to the positions of the top-pins of the net; this causes the respective routes to sort towards these positions during the sorting. The routes of the T-nets must, however, meet at some point within the channel. To facilitate this, T-net pins are assigned to the sides *opposite* of their relative positions in the top track. This causes the routes to cross each other at some point on their way to their final positions.

In Fig.5, net D — a T-net— has pins on top of the channel at columns 3 and 9. These column positions are used as the indexes assigned to the pins at the bottom of the channel — index 9 on the left, 3 on the right. As the sorting proceeds, the routes for the T-nets converge towards each other, cross at some point within the channel, and continue to their final sorted position. After the channel routing completes, only the routing above the crossing point of a T-net is retained as the actual T-net route (e.g. the solid-line route of D in Fig.5).

C. Limitations of Swap Router

Consider the swap-router solution depicted in Fig.5. Immediately apparent are the following problems:

1) wasted tracks (gaps) with sub-optimal routes;
2) the final (top) positions of routed net terminals are shifted to the right, as compared to the original specification;
3) routes detour away from destination column position, creating more "bends", which can cause optical signal loss;
4) routes exist outside the channel's column bounds.

These problems arise from the following sources:

Gaps in the sorting problem: Empty space (i.e. "gaps") in the channel problem affect the relative positioning of nets in a track, while not providing any sorting information themselves. The end result is that the routing solution only respects the *relative* positioning of nets, not the *absolute* position. This is problematic, as the routed nets of the top track may not reflect the positions of the pins in the original specification; this is demonstrated in the top track of Fig.5. Both B-nets and T-nets introduce gaps into the sorting problem.

T-net encoding: As demonstrated in Fig.5, the encoding for T-nets *negatively impacts virtually all aspects of the channel solution.* T-nets introduce additional channel columns that affect positioning of the net pins, as well as enabling routes to extend outside the channel's original column bounds. The T-net temporary/virtual routes (dashed lines in Fig.5) also cause unnecessary detours for any other route they interact with, increasing track count — a track for each crossing — during the sorting. For example, in Fig.5, the T-net route originating from the left-side for net J must cross seven (7) other routes to form a solution. None of the temporary routing below the cross-over point serves any real purpose in the final channel solution. This also causes other routes to move outwards, such as the route for E detouring left as it crosses the left terminal route for A, D, and J.

Post-processing: Accepting the solution from the swap router as-is would require an additional post-processing step to: 1) position T-net terminals correctly with respect to the original specification, and 2) post-process routes within the solution to improve track usage and prevent routes from extending outside the bounds of the channel. While some post-processing work is performed in [25] to compact the track space, gap handling remains unaddressed. Larger problems, especially with many T-nets, can produce extremely large solutions, most of which is wasted space. This effectively defeats the purpose of utilizing the swap router in the first place.

4. 2-sided Swap Routing (2Swap)

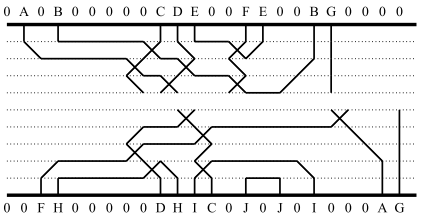
To overcome the limitations of the original sorting routing, we introduce the 2-sided Swap Router (2Swap): a sorting-based router that performs routing from both sides of the channel simultaneously. Sorting still remains a key component of routing, ensuring crossing-minimality; however, the routing from both sides overcomes two key limitations of the original router:

- The elimination of T-nets from the routing solution. No additional columns are added to the channel problem and no temporary routes are needed.

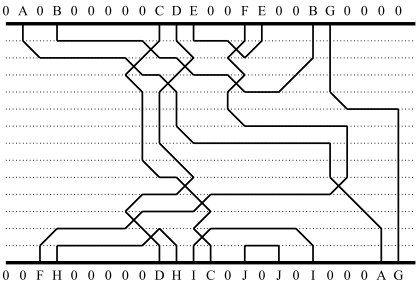
- Sorting in the presence of gaps is addressed, and pins on each side of the channel are fixed as per the original specification.

Crossing minimality: As in the original swap router, the 2Swap router produces a crossing-minimal solution. This is ensured by updating the position of route pins at each iteration of the routing. A swap that takes place on one side of the channel is reflected when the other side is routed, retaining the sorting-based assurance of crossing-minimality.

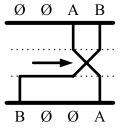
Gap crossing: Gaps are an intrinsic part of virtually every channel routing instance. The presence of gaps, however, is not even mentioned by [24], and other examples in literature [25] only depict and describe dense (gapless) routing problems such as Fig.4. We address the existence of gaps within the sorting problem by allowing routes to span horizontally across gaps to the point of crossing. During sorting, pairs of routes are analyzed across gaps. Should a swap be possible, a horizontal span is created up to the point of crossing, and the swap then occurs. This is depicted in Fig.6(c), where a route for net *B* traverses multiple gaps (∅) to cross over route *A* on the right.



(a) Sorted, but not fully-routed solution



(b) After post-sort routing



(c) Horizontal gap spanning

Fig. 6: Post-sort routing for 2Swap solution.

Two-sided swap-routing: The 2Swap router alternates between both sides of the channel for swap-routing, while performing bookkeeping to ensure that crossings are not performed twice and sorting-indexes are updated. We now define two different net-concepts:

Definition 4.1. [Source/destination sides, S-net, D-net] *Given a channel side being routed, routing is performed from the **source** (src) to **destination** (dest) sides of the channel. The src/dest sides of the channel are analogous to the bottom/top sides when routing is performed from the bottom-to-top. Given a channel side being routed, an **S-net**is an **source-side net** — a net with both pins exclusively on the source side of the channel. Likewise, a **D-net**is an **destination-side net** — a net with both pins exclusively on the destination side of the channel.*

The technique works in the following manner:

1) For each iteration of the 2Swap router, swap-routing is performed for a single track on both sides of the channel, subject to the following conditions:
   - For each iteration of the sorting, relative ordering of nets is recomputed for swap-routing, utilizing the current set of sorted tracks. This helps ensure that the routing performed for a given side is aware of net-swaps that took place on the opposite side of the channel.
   - For a given side of a channel being routed, only S-nets and X-nets are encoded as indexes and routed. D-nets are treated as gaps on the destination track for purposes of routing. This ensures that D-nets are only routed on their respective side as S-nets, avoiding the problems previously associated with T-nets in the original sorting-based router.
   - Any S-nets that have completed routing are removed in subsequent tracks (replaced by gaps).
2) Routing completes when routes on both the top and bottom are completely sorted with respect to each other.
3) A post-sort routing is performed to provide a routable channel.

The 2Swap technique is given in Algorithm 1.[2]

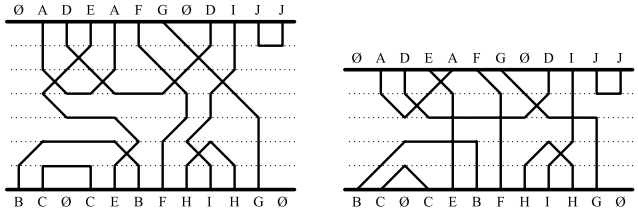
Post-sort routing: The sorting phase of routing completes when all S-nets are routed, and all X-net-indexes are sorted with respect to each other. The latter condition, however, does not guarantee that the channel is fully routed. Consider the 2Swap routing solution depicted in Fig.6(a). While the X-nets in the middle of the channel are *sorted* with respect to each other, they are not fully *routed* as their column positions are misaligned. In such cases, *post-sort routing* must be applied, as demonstrated in Fig.6(b).

A. Solution Quality

We route the channel depicted in Fig.5 using the new 2Swap router and observe that it produces a far more usable solution (Fig.7(a)). Fewer tracks are utilized (6 vs 11) and no additional columns are added to the routing solution. In addition, the top and bottom pins locations are the same as in the specification. Overall, the solution produced by the 2Swap router is improved with respect to the original swap-router. However, further improvements are still possible, especially in terms of waveguide bends.

1) Excess bends and their cause: The 2Swap route produces a large number of excess bends in its solutions. This is especially apparent in larger channel instances such as Fig.6(b). The excess bends in 2Swap routes are due to the fact that while the position of net *terminals* are fixed to *absolute* positions, the intermediate sorting is still a *relative* sorting-position operation. Pairs of routes for a given net are therefore

[2]All algorithms are found at the end of the manuscript



(a) 2-sided swap (2Swap) (b) Constrained 2-sided swap (2SwapC)

Fig. 7: Routing solutions for the same channel instance

not actively converging towards each other during the sorting. For example, in Fig.7(a), net E swaps with net B, shifting E to the right — in the opposite direction of its destination pin. As a result, route E must detour across a large expanse of space to connect both sides — despite being only one column away in the original problem. The same problem afflicts net I, which detours left in its swap with H.

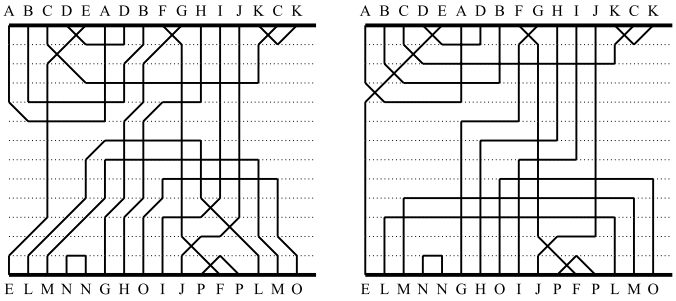
As a signal loss mechanism, bends must also be accounted for, and we address these with a *constrained* 2-sided Swap Router.

5. Constrained 2-Sided Swap Routing

The Constrained 2-Sided Swap Router (2SwapC) introduces the following key features:

- **Convergence and swapping constraints:** Routes cannot change columns unless swapping or utilizing a horizontal span. Swapping only occurs between adjacent routes, under the condition that the swap results in each respective route converging towards its respective opposite pin.
- **Horizontal spanning:** Routes may now cross each other using horizontal spans in addition to diagonal (swap) crossings. For a given route, the horizontal spans are only permitted as a direct connection to the paired pin's route, completing the routing.

The 2SwapC technique is given in Algorithm 3.

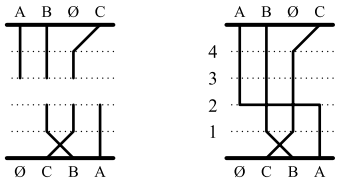


(a) 2SwapC allowing convergence towards other pins (12-tracks) (b) 2SwapC fully-constrained (12-tracks)

Fig. 8: Effect of constraining route movement

Convergence constraint and swapping restrictions: Routes are restricted from shifting columns, except during a swap or utilizing a horizontal span (explained later). Swapping is still allowed, but restricted: adjacent net-routes must both converge towards their opposing pin. The convergence swapping constraint work together to reduce the number of bends in the routed solution.

To reduce unnecessary bends, we strictly constrain routes to only shift horizontally during a swap. In the absence of such a swap, this restriction results in straight vertical connections, as depicted in Fig.8(b). While the routes cover more horizontal area, that area is already empty. In addition, the overall track height generally remains the same or is often reduced, as will be seen later.



(a) Net spanning: A cannot cross B or C diagonally (b) Horizontal connection across nets

Fig. 9: Horizontal connections to complete routing

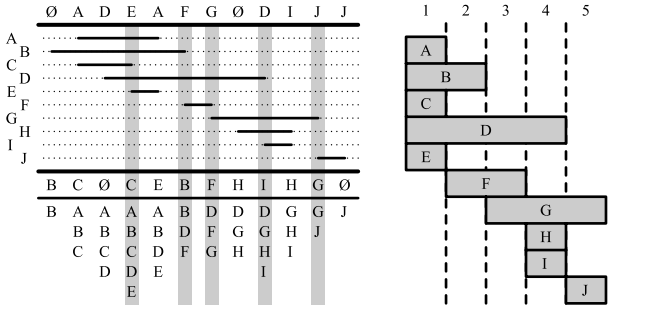
Vertical spans form naturally due to the convergence constraint, as depicted in Fig.9(a). As vertical spans form, a *horizontal span* is allowed in cases where crossing other routes does not result in a sorting violation. Horizontal spans are also only allowed to cross vertical spans that traverse two (2) or more tracks, to ensure that bends will not occur at the junction. This is depicted in Fig.9(b), where the horizontal crossing may not cross at track 1, but must instead cross at track 2. Furthermore, to reduce the number of bends, routes may span horizontally only if they can make a *direct* connection to the column of their respective paired route. The effect of this is that any given net will only make one horizontal span within the channel. This can be observed in the channel solution depicted in Fig.8(a).

Comparison with 2Swap: While the number of crossings is the same, channel solutions produced by 2SwapC have fewer bends than 2Swap. This is evident in Fig.7 where the 2Swap router produces many routes that "zig-zag" throughout the routing region, whereas 2SwapC utilizes straight routes with few or no bends. The net result is fewer bends, more vertical spans to permit horizontal cross-overs, and retaining the ability for neighboring nets to cross diagonally — reducing tracks. Moreover, the restrictions still retain the sorting mechanism of the other swap routers, ensuring crossing minimality.

The 2SwapC router is applied to a number of benchmarks later in this paper to evaluate its performance. We also investigate incorporating crossing-aware constraints onto traditional left-edge style routers, which we describe in the following section.

6. Left-edge-style Channel Routing

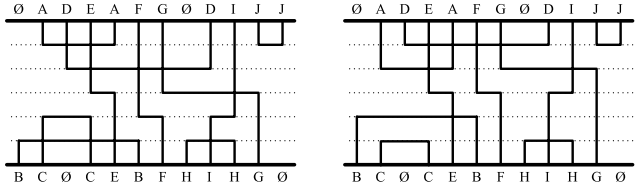
Traditional left-edge-style channel routers [26]–[28] represent the channel routing problem using horizontal and vertical constraint graphs (HCG, VCG). An alternate representation of the HCG is the zone representation, which is derived from the HCG, where every zone is defined by a maximal clique. The number of signals in the largest zone is the lower bound on the number of tracks needed for routing. These graphs encode constraints on how tracks may be assigned to nets in the channel. Consider the channel routing problem depicted



(a) Five maximal subsets of signals. (b) Resulting five zones.

Fig. 10: Horizontal constraints and zone representation.

in Fig.10(a). The resulting zone representation is depicted in Fig.10(b). Likewise, the VCG for the problem is represented in Fig.12(a).



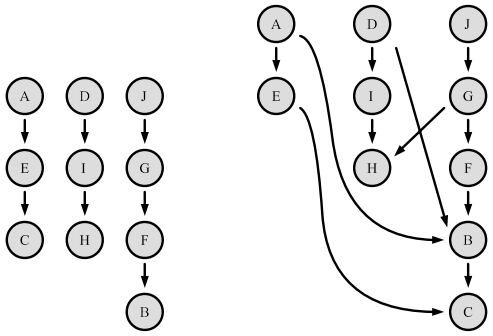
(a) Track-optimized (5 tracks, 10 crossings). (b) Crossing-constrained (5 tracks, 8 crossings).

Fig. 11: Channel routing solutions under differing constraints.

A net may be assigned to a track should it have no descendants on the VCG, and have no overlapping zone conflicts with previously assigned nets on a given track. Nets are removed from the VCG as they are assigned to tracks. When a track cannot contain more nets, a new track is created and the process is repeated until no more nets are left for assignment.

Multiple nets can be candidates for assignment to a given track, each with different horizontal overlaps. Therefore heuristics are used to choose which nets are assigned first. One of the simplest is a greedy heuristic used in *constrained left-edge channel routing* [26], where the left-most available nets in channel are assigned first to tracks. This can lead to sub-optimal track-utilization; more sophisticated heuristics analyze the graph structure for better results, such as [28], which attempts to reduce the longest path in the VCG for better track utilization. We refer to the class of track assignment algorithms above generically as "left-edge-style" channel routing. The approach we describe below can be incorporated into any such techniques.

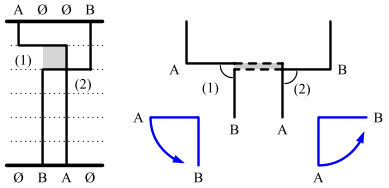
Crossing-constrained track assignment: Fig.11(a) depicts the output of a (VLSI) left-edge 2-layer channel router, and Fig.11(b), a channel routing constrained for crossing-minimization. Both solutions are minimal in terms of tracks; however, the total number of crossings in Fig.11(a) is 10, compared to 8 in Fig.11(b). The discrepancy in the number of crossings is attributed to the two crossing points caused by nets B and C. By forcing C to appear below B, two crossings are avoided. However, transforming from Fig.11(a) to Fig.11(b) is not as simple as moving net C below B, not if track height is



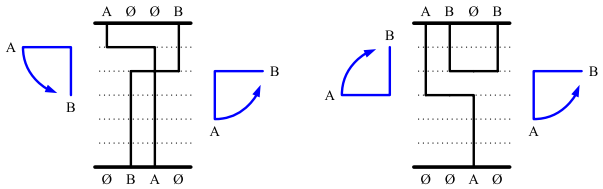
(a) Original VCG. (b) With crossing constraints.

Fig. 12: Crossing-constraints modifications to the VCG

to be kept minimal. Crossing minimization must therefore be encoded into the routing process itself as constraints.



(a) Rotation direction with respect to pin locations.



(b) Same rotation direction ⇒ Unavoidable crossing. (c) Opposite rotation directions ⇒ Avoidable crossing.

Fig. 13: Crossing detection via rotation from A to B.

We constrain the channel routing problem to favor crossing minimization. The VCG is modified such that avoidable crossings impose vertical constraints on the net ordering. Only nets that share zones have the possibility of crossing, and pairwise analysis takes place after the zones are derived.

A crossing constraint is only encoded into the VCG if a crossing can be avoided. For example, the pair of nets in Fig.13(c) would not normally be constrained in the VCG; however, a net crossing can be avoided if B is assigned a track above A. Therefore, an edge connecting B to A is added to the VCG. Conversely, the two nets in Fig.13(b) cannot avoid crossing, and therefore no constraint is added.

We introduce the concept of *pin-rotation* to detect avoidable crossings. If we were to map the pins of nets on a unit circle, a crossing is unavoidable if rotating from one pin to the next is not possible without first passing through the pin of another net. Consider the nets depicted in Fig.13(a). Collapsing the shared horizontal region, and considering the areas Fig.13(a)(1) and Fig.13(a)(2) shows how pins of a given side rotate with respect to each other (clockwise/counter-clockwise) around an axis fixed at the center. In the case of Fig.13(a)(1), the rotation of the left pin of A to the left pin of

B is *counterclockwise,* and likewise the pins on the right-side also rotate in the same *counterclockwise* direction. If the pins on both left and right terminals rotate in the same direction a crossing is unavoidable. More formally:

$$X^{left}_{A,B,CW} = \begin{cases} X^{left}_{B,top} & \text{if } C^{left}_{A} < C^{left}_{B} \\ \neg X^{left}_{A,top} & \text{otherwise} \end{cases} \quad (1)$$

$$X^{right}_{A,B,CW} = \begin{cases} X^{right}_{A,top} & \text{if } C^{right}_{A} < C^{right}_{B} \\ \neg X^{right}_{B,top} & \text{otherwise} \end{cases} \quad (2)$$

$$X_{\text{avoidable}}(A,B) = \left(X^{left}_{A,B,CW} \neq X^{right}_{A,B,CW} \right) \quad (3)$$

where $C^{left/right}_{N}$ is the integer-valued column-position of a pin of net N on a given side (left, right); the Boolean variable $X^{left/right}_{N,top}$, using the same notation, denotes whether that pin resides on the top side of the channel. Eqn.1 and Eqn.2 utilize the horizontal relationships of pins and their channel-sides (top/bottom) to determine the *clockwise rotation* (CW) of a given pair of $left$ or $right$ pins for nets A and B, rotating from A to B. A crossing is avoidable only if left and right rotations are *not* the same, the result of Eqn.3.

For example, in Fig.13(a), consider the left side of the shared span Fig.13(a)(1):

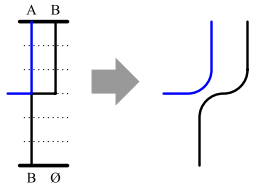
- The variables C^{left}_{A} and C^{left}_{B} are the column positions of the respective left-terminals of nets A and B. In the example, $C^{left}_{A} = 1$, $C^{left}_{B} = 2$.
- $C^{left}_{A} < C^{left}_{B}$ implies $X^{left}_{A,B,CW} = X^{left}_{B,top}$ from Eqn.1.
- The left pin of net B is *not* on the top side of the channel ($X^{left}_{B,top} =$ **false**). Therefore, the left side of the pair of nets is *not* rotating clockwise from A to B, i.e. $X^{left}_{A,B,CW} = X^{left}_{B,top} =$ **false** .
- On the right side of the shared span Fig.13(a)(2), $C^{right}_{A} < C^{right}_{B}$. This condition implies that $X^{right}_{A,B,CW} = X^{right}_{A,top} =$ **false** . The right side is therefore also *not* rotating clockwise from A to B.

Having the same direction of rotation ($X^{left}_{A,B,CW} = X^{right}_{A,B,CW} =$ **false**) implies that a crossing is *unavoidable,* as determined by Eqn.3; this is reflected in the figure.

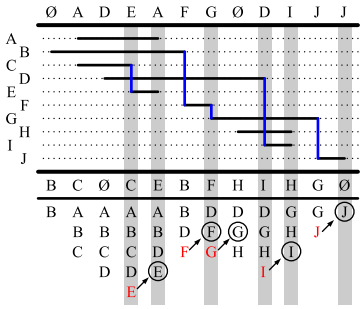
Applying crossing constraints to the problem depicted in Fig.10(a) results in the VCG depicted Fig.12(b). As compared to the original VCG Fig.12(a), the crossing-constrained VCG is more heavily constrained, ensuring that unnecessary crossings do not occur, such as the double-crossing of nets B and C in Fig.11(a).

Knock-knee track sharing: Though the modified VCG is effective in preventing waveguide crossings, the additional constraints can affect overall track height, and may produce a worse solution in terms of number of tracks. However, we observe that the bend geometry of optical waveguides can be exploited to further reduce channel height. This is discussed below.

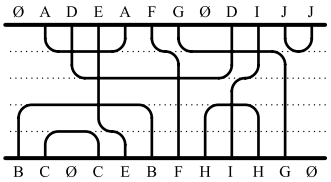
Consider the two nets in Fig.14(a). The endpoints of the two nets occupy the same column and therefore net A should be placed above B in the VCG. However, given the same track, the two nets would intersect at a corner of each horizontal span–a *knock-knee*. In VLSI, this situation is untenable, and



(a) Knock-knee implementation.



(b) Knock-knee-constrained zone representation.



(c) 4-track routing solution utilizing knock-knees.

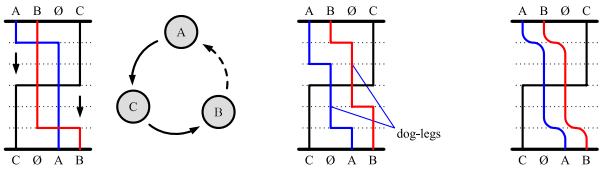
Fig. 14: VCGs for Fig.10(a) and knock-knee extension.

different tracks would need to be assigned to each net. However, for waveguides, the minimum grid spacing for a channel can permit knock-knees in the routing grid. This is depicted in Fig.14(a), where a track is shared between the two nets without overlap.

A knock-knee occurs where one net ends and another begins, e.g. nets C and E in Fig.14(c). During zone construction, at columns where knock-knees appear, the net that is beginning its horizontal span is only added to the *subsequent* column set, rather than the current column set under consideration. For example, in Fig.14(c) knock-knee signals E, F, G, I, and J are removed from the marked columns and only appear in the subsequent columns. The effect of this column change on the resulting zones is demonstrated in Fig.14(b), where there are six (6) zones rather than the five (5) from the previous zone analysis Fig.10. Despite containing an additional zone, the largest column set now contains *one fewer* net than the original, resulting in the 4-track solution depicted in Fig.14(c).

Overall, the effect incorporating knock-knees into a routing solution is that two knock-knee nets can now occupy separate zones, and therefore can be placed on the same track. Additional zones may be created; however, those zones are equal in size or *smaller* in terms of nets — *potentially reducing the lower bound on the number of tracks required for routing.*

Cycles induced by crossing constraints: Crossing constraints can induce cycles in the VCG. Consider the three nets depicted in Fig.15(a). Without crossing constraints, nets A and B would



(a) Vertical cyclic constraints. (b) Dog-legging avoids crossings. (c) Knock-knees avoid additional tracks.

Fig. 15: Cycles induced by crossing constraints.

be unconstrained, and no cycle would occur; however, due to the constraint edge between B and A such a cycle occurs. Cyclic constraints cannot be routed without additional tracks and require "doglegging" to complete routing [27]. In order to avoid crossings, the routes for A and B are converted into doglegging routes as depicted in Fig.15(b), utilizing the same columns as the original. Unfortunately, this results in an additional two (2) tracks being added to the routing solution should spare tracks not be available adjacent to the cycle. However, in the presence of knock-knees, both the crossings, and the additional tracks can be avoided, as depicted in Fig.15(c). The experimental results show that knock-knees can have a marked difference in track utilization especially in the presence of cyclic constraints induced by crossings.

7. Experimental Results

We evaluate our sorting-based router (2SwapC) and our crossing-aware left-edge router (LE+KK) on a number of channel problem instances. We also compare against the Yoshimura-Kuh (YK) router [28] as a baseline. The original sort-based router [24], [25] is not compared, as it produces-routing solutions that violate the original channel specifications.

Channel problem instances: In the VLSI community, it is customary to evaluate routing techniques against benchmark suites such as those described in [42]. However, such benchmarks are inapplicable to this work. The VLSI routing benchmarks utilize multiple routing layers, with nets incorporating large amounts of fan-out. This is due to the aggressive factorization-based logic decomposition/synthesis applied in the VLSI domain.

Our investigations found these techniques to be inapplicable to photonic logic. Subsequently, we proposed technology-specific netlist decomposition techniques specific to silicon photonics [6]. The channel routing instances are derived from relevant integrated optical designs.

Our channel problem instances are derived from the ACM/SIGDA (i.e. MCNC) logic synthesis benchmark suites [43]. These designs are synthesized into waveguide-connected, optical switching networks, utilizing our optical logic synthesis technique [6]. The optical netlist is then placed into rows using a VLSI row-placer [44]. Crossing-aware global routing is performed subsequently, utilizing an MILP-based approach on a set of candidate routes. After global routing completes, the regions between rows are extracted as channel problem instances. Multiple channel routing problems were derived using the above design flow.

A. Metrics

Channels are routed to evaluate their performance in terms of key metrics: 1) crossings, 2) bend-loss, and 3) track utilization (area). In all problem instances, the number of crossings produced by the 2SwapC and LE+KK routers is the same.

(a) 90° curve with $r = 1$ loss/bend = 0.368 (b) 135° curve with $r = 2$ loss/bend = 0.135

Fig. 16: Radius of curvature for grid bends.

Bend-loss: In our octilinear grid, bends can be either 90° or 135°, having different loss characteristics. Bend-loss is computed for each using Eqn.4 [45] as a function of radius of curvature (Fig.16):

$$\alpha_{bend}(r) = C_1 \cdot \exp(-C_2 \cdot r) \tag{4}$$

The constants C_1 and C_2 are dependent on the physical parameters of the waveguides. For simplicity and convenience, we use a unit grid for calculating bend-radius, and select $C_1 = C_2 = 1$. This results in bends-loss for $\alpha_{bend}(\{135°, 90°\}) = \{0.135, 0.368\}$.

In addition to counting bends within the channel, we also include bends at the interface of the side of the channel to the first track of each respective side.

B. Analysis of results

The results of the routings are depicted in Tbl.I (appended at the end of the manuscript). Total time for both routers to complete all channel problem instances was less than 10 seconds. Analysis of the results reveals that:

- **Crossings:** Both 2SwapC and LE+KK perform equally well in terms of number of crossings. Moreover, on average, both routers have 43% fewer crossings than the YK router.
- **Bend loss:** The LE+KK router produces nearly the same bend loss as YK router due to the similiar track-assignment routing formulation. However, 2SwapC consistently produces better results compared to LE+KK (5.6% less average bend loss).
- **Tracks:** LE+KK often utilizes fewer tracks than 2SwapC (8% less, on average). As expected, both 2SwapC and LE+KK produce more tracks than the strictly track-optimizing YK router, except in the two instances noted in the table. On average, the increase in tracks is 24% and 14% for 2SwapC and LE+KK, respectively.

In comparing 2SwapC and LE+KK, the 2SwapC router ultimately produces less bend loss. We attribute this to 2SwapC's use of a non-Manhattan grid, enabling waveguide bends with larger curvature (e.g. 135° curves). This contrasts with nets routed by LE+KK, which always require 90° bends. In terms of tracks, LE+KK offers slightly better track utilization due to enforced 90° curves and the single-track utilization model.

8. Conclusion

This paper presents channel routing techniques for integrated optical waveguides fabricated on a planar substrate. We identify our primary optimization objective as signal loss minimization — in terms of waveguide crossings and bends-loss — with area as a secondary objective. We present two distinct crossing-aware channel routing techniques for integrated optics: 2SwapC — a sorting-based router based on a non-Manhattan routing grid, and LE+KK — a left-edge style router utilizing crossing-aware graph-constraints. Both techniques are crossing-minimal, and are constrained in a technology-suitable fashion to reduce bend-loss.

Our new 2SwapC router addresses and overcomes the shortcomings of previous sort-based routers through the use of two-sided swap routing. We have further improved the router through additional routing constraints, as well as innovative extensions that enable routes to utilize horizontal spans, while reducing bends. Bend-losses are markedly improved, as demonstrated in tests on many channel routing instances, and 2SwapC demonstrates that it is superior in terms of bend-loss than the LE+KK router.

The LE+KK router we present builds upon traditional constraint-graph based ("left-edge style") routers to produce a crossing-minimal channel routing solution. This router incorporates additional constraints to ensure crossing minimality by analyzing pairs of nets using the concept of *pin rotation*. We also exploit waveguide curves to enable knock-knees to improve track utilization as well as to enable crossing-constrained doglegging, improving the solution quality.

Our channel routers provide effective means for automating optical waveguide routing with signal loss as a primary metric. When applied to channels derived from optical designs, 2SwapC and LE+KK both produce comparable results. In terms of our primary signal loss metric, however, we choose the 2SwapC router, as its ability to utilize non-Manhattan grids gives it greater potential in reducing overall signal loss. The LE+KK router is still a good choice in cases where area is most important.

References

[1] M. Peach, "Silicon photonics forges ahead", *Optics.org*, Feb 2012.

[2] "IBM Research: Silicon Integrated Nanophotonics", www.research.ibm.com/photonics.

[3] "OpSIS: Optoelectronic System Integration in Silicon", http://www.opsisfoundry.org.

[4] J. Chan, G. Hendry, K. Bergman, and L. Carloni, "Physical layer modeling and system-level design of chip-scale photonic interconnection networks", *IEEE Trans. on Computer-Aided Design of Electronic Systems*, 2011, In Press.

[5] C. Madsen and J. Zhao, *Optical Filter Design and Analysis: A Signal Processing Approach*, Wiley, 1999.

[6] C. Condrat, P. Kalla, and S. Blair, "Logic Synthesis for Integrated Optics", *Proc. ACM Great Lakes Symp. on VLSI*, pp. 13 – 18, 2011.

[7] H. J. Caulfield, C. S. Vikram, and A. Zavalin, "Optical logic redux", *Optik*, vol. 117, pp. 199–209, 2006.

[8] A. Politi, J. Matthews, and J. O'Brien, "Shor's Quantum Factoring Algorithm on a Photonic Chip", *Science*, vol. 325, pp. 1221, 2009.

[9] A. J. Poustie and K. J. Blow, "Demonstration of an all-optical Fredkin gate", *Optics Communications*, vol. 174, pp. 317 – 320, 2000.

[10] S. Kotiyal, H. Thapliyal, and N. Ranganathan, "Mach-Zehnder Interferometer based Design of all Optical Reversible Binary Adder", *in Proc. Design Automation & Test in Europe (DATE) Conf.*, 2012.

[11] D. Ding and D. Pan, "Oil: A nano-photonics optical interconnect library for a new photonic network architecture", *in System-Level Interconnect Prediction Workshop (SLIP)*, 2009.

[12] D. Ding, Y. Zhang, H. Huang, R. T. Chen, and D. Z. Pan, "O-Router: an optical routing framework for low power on-chip silicon nano-photonic integration", *in Design Auto. Conf.*, pp. 264–269, 2009.

[13] J. Orcutt and R. Ram, "Photonic Device Layout within the Foundry CMOS Design Environment", *IEEE Photonics Tech. Lett.*, 2010.

[14] D. Ding, B. Yu, and D. Pan, "GLOW: A global router for low-power thermal-reliable interconnect synthesis using photonic wavelength multiplexing", *in Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pp. 621–626, 30 2012-Feb. 2.

[15] Y. Zheng, P. Lisherness, M. Gao, J. Bovington, K. Cheng, H. Wang, and S. Yang, "Power-Efficient Calibration and Reconfiguration for Optical Network-on-Chip", *J. Opt. Commun. Netw.*, vol. 4, pp. 955–966, Dec 2012.

[16] H. J. Caulfield et al., "Generalized optical logic elements GOLEs", *Optics Communications*, vol. 271, pp. 365–376, mar 2007.

[17] C. Batten, A. Joshi, V. Stojanovic, and K. Asanovic, "Designing Chip-Level Nanophotonic Interconnection Networks", *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 2, pp. 137 –153, june 2012.

[18] W. Bogaerts, P. Dumon, D. V. Thourhout, and R. Baets, "Low-loss, low-cross-talk crossings for silicon-on-insulator nanophotonic waveguides", *Opt. Lett.*, vol. 32, pp. 2801–2803, Oct 2007.

[19] F. Xu and A. W. Poon, "Silicon cross-connect filters using microring resonator coupled multimode-interference-based waveguide crossings", *Opt. Express*, vol. 16, pp. 8649–8657, Jun 2008.

[20] J. Cardenas, C. B. Poitras, J. T. Robinson, K. Preston, L. Chen, and M. Lipson, "Low loss etchless silicon photonic waveguides", *Opt. Express*, vol. 17, pp. 4752–4757, Mar 2009.

[21] Y. Vlasov and S. McNab, "Losses in single-mode silicon-on-insulator strip waveguides and bends", *Opt. Express*, vol. 12, pp. 1622–1631, Apr 2004.

[22] Y. Qian, S. Kim, J. Song, G. P. Nordin, and J. Jiang, "Compact and low loss silicon-on-insulator rib waveguide 90° bend", *Opt. Express*, vol. 14, pp. 6020–6028, Jun 2006.

[23] G. Li, J. Yao, H. Thacker, A. Mekis, X. Zheng, I. Shubin, Y. Luo, J. Lee, K. Raj, J. E. Cunningham, and A. V. Krishnamoorthy, "Ultralow-loss, high-density SOI optical waveguide routing for macrochip interconnects", *Opt. Express*, vol. 20, pp. 12035–12039, May 2012.

[24] K. Chaudhary and P. Robinson, "Channel Routing by Sorting", *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 10, pp. 754 –760, jun 1991.

[25] J. Yan, "An improved optimal algorithm for bubble-sorting-based non-Manhattan channel routing", *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 18, pp. 163 –171, feb 1999.

[26] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures", *in Proceedings of the 8th Design Automation Workshop*, DAC '71, pp. 155–169, New York, NY, USA, 1971. ACM.

[27] D. Deutsch, "A Dogleg channel router", *in Proceedings of the 13th Design Automation Conference*, DAC '76, pp. 425–433, New York, NY, USA, 1976. ACM.

[28] T. Yoshimura and E.S. Kuh, "Efficient Algorithms for Channel Routing", *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 1, pp. 25 – 35, january 1982.

[29] M. Petracca, B. G. Lee, K. Bergman, and L. Carloni, "Design exploration of optical interconnection networks for chip multiprocessors", *in IEEE Symp. High Perf. Interconnects*.

[30] M. Cianchetti, J. Kerekes, and D. Albonesi, "Phastlane: A rapid transit optical routing network", *in Proceedings of the 36th annual International Symposium on Computer Architecture*, ISCA '09, pp. 441–450, New York, NY, USA, 2009. ACM.

[31] R. Beausoleil et al., "A Nanophotonic Interconnect for High-Performance Many-Core Computation", *Symposium on High-Performance Interconnects*, pp. 182–189, 2008.

[32] J. Hardy and J. Shamir, "Optics Inspired Logic Architecture", *Optics Express*, vol. 15, pp. 150–165, 2007.

[33] C. Condrat, P. Kalla, and S. Blair, "Exploring Design and Synthesis for Optical Digital Logic", *International Workshop on Logic Synthesis*, 2010.

[34] RSoft Inc., *RSoft Photonics CAD Suite*, www.rsoftdesign.com.

[35] Lumerical Solutions, Inc., *Lumerical Photonics CAD Suite*, http://www.lumerical.com.

[36] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: from Graph Partitioning to Timing Closure*, Springer , 2010.

[37] S. Sait and H. Youssef, *VLSI Physical Design Automation*, IEEE Press, 1995.

[38] M. Marek-Sadowska and M. Sarrafzadeh, "The crossing distribution problem [IC layout]", *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 14, pp. 423–433, Apr.

[39] S. Xiaoyu and W. Yuke, "On the crossing distribution problem", *ACM Trans. Des. Autom. Electron. Syst.*, vol. 4, pp. 39–51, jan 1999.

[40] S. B. Stephen and L. S. Kyu, "QCA channel routing with wire crossing minimization", *in Proceedings of the 15th ACM Great Lakes symposium on VLSI*, GLSVLSI '05, pp. 217–220, New York, NY, USA, 2005. ACM.

[41] C. Cheng and D.N. Deutsch, "Improved channel routing by via minimization and shifting", *in Design Automation Conference, 1988. Proceedings., 25th ACM/IEEE*, pp. 677–680, 1988.

[42] G. Nam, C. Sze, and M. Yildiz , "The ISPD global routing benchmark suite", *in Proceedings of the 2008 international symposium on Physical design*, ISPD '08, pp. 156–159, New York, NY, USA, 2008. ACM.

[43] ACM/SIGDA, "ACM/SIGDA benchmarks", http://www.cbl.ncsu.edu/benchmarks/Benchmarks-upto-1996.html.

[44] J. Roy, D. Papa, S. Adya, H. Chan, A. Ng, J. Lu, and I. Markov, "Capo: robust and scalable open-source min-cut floorplacer", *in Proceedings of the 2005 international symposium on Physical design*, ISPD '05, pp. 224–226, New York, NY, USA, 2005. ACM.

[45] E. A. J. Marcatili and S. E. Miller, "Improved relationships describing directional control in electromagnetic wave guidance", *Bell Syst. Tech J.*, vol. 48, pp. 2161–2188, 1969.

Algorithm 1 2Swap: Main Swap Router

$T_{sol} := [\,]$ ***/* (solution tracks for channel) */***
repeat
  for each side S of the channel **do**
    $T_{src} :=$ copy of current track of *src* side
    $T_{dest} :=$ copy of current track of *dest* side
    Assign all D-net nets in T_{dest} to $\emptyset$
    $T_{unsorted} :=$ array of net-indexes of T_{src} with respect to T_{dest}
    $T_{sorted} :=$ SWAPSORT ($T_{unsorted}$)
    Add T_{sorted} to T_{sol}
    Remove completed S-nets from T_{sorted}
    Set current track for S to T_{sorted}
  end for
until (all B-nets and T-nets are routed) **and** (all X-nets are sorted with respect to each other)
$T_{sol} :=$ POSTSORTROUTING(T_{sol})

Algorithm 2 2Swap: Per-track SwapSort Function

function SWAPSORT (T as array of indexes)
  for $i = 1 \ldots$ sizeOf(T) **do**
    if $T(i) \neq \emptyset$ **and** $T(i+1) \neq \emptyset$ **and** $T(i) > T(i+1)$ **then**
      SWAPVALUES($T(i), T(i+1)$)
      /* Additional increment to skip to next pair */
      $i := i+1$
    end if
  end for
  return T
end function

Algorithm 3 2SwapC: Main Swap Router

$T_{sol} := [\,]$ ***/* (solution tracks for channel) */***
$H_{sol} := [\,]$ ***/* (horizontal spans for channel) */***
repeat
  for each side S of the channel **do**
    $t :=$ sizeOf(T_{sol}^{src})
    $T_{src} := T_{sol}^{src}[t]$ ***/* copy of current track of* src *side */***
    $T_{dest} := T_{sol}^{dest}[t]$ ***/* copy of current track of* dest *side */***
    Assign all D-net nets in T_{dest} to $\emptyset$
    $T_{unsorted} :=$ net-indexes of T_{src} with respect to T_{dest}
    $T_{y-1} := T_{sol}^{src}[t-1]$ ***/* previously routed track */***
    $T_y := T_{unsorted}$ ***/* currently routed track */***
    /* next routed track: */
    $T_{y+1} :=$ CONSTRAINEDSWAPSORT ($T_{unsorted}$)
    $H_{spans} :=$ HORIZONTALSPANS (T_y, T_{y+1}, T_{y-1})
    Remove h-span-completed S-nets from T_{y+1}
    Add H_{spans} to H_{sol}; Add T_{y+1} to T_{sol}
    Set current track for S to T_{y+1}
  end for
until (all B-nets and T-nets are routed) **and** (all X-nets are sorted with respect to each other)

Algorithm 4 2SwapC: Per-track Constrained SwapSort Function

function CONSTRAINEDSWAPSORT (T as array of indexes)
  for $i = 1 \ldots \text{sizeOf}(T)$ **do**
    if $T(i) = \emptyset$ **or** $T(i+1) = \emptyset$ **then**
      continue
    end if
    /* Swap constraints for $T(i)$ ***and*** $T(i+1)$ ****/***
    $x_i^a := \text{COLUMNOF}(T(i))$
    $x_i^b := \text{COLUMNOF}(\text{other route of } T(i))$
    $x_{i+1}^a := \text{COLUMNOF}(T(i+1))$
    $x_{i+1}^b := \text{COLUMNOF}(\text{other route of } T(i+1))$
    $cond_i := \begin{cases} \textbf{true} & (x_i^b - x_i^a) > 0 \\ \textbf{false} & \textbf{otherwise} \end{cases}$
    $cond_{i+1} := \begin{cases} \textbf{true} & (x_{i+1}^b - x_{i+1}^a) < 0 \\ \textbf{false} & \textbf{otherwise} \end{cases}$
    if $cond_i$ **and** $cond_{i+1}$ **then**
      $\text{SWAPVALUES}(T(i), T(i+1))$
      $i := i + 1$
    end if
  end for
  return T
end function

Algorithm 5 2SwapC: Horizontal Spanning Function

function HORIZONTALSPANS (T_{y+1}, T_y, T_{y-1} as arrays of indexes)
  /* Detects horizontal spans for track T_y ****/***
  $H_{spans} := [\,]$ ***/* (output spans) */***
  for $i = 1 \ldots \text{sizeOf}(T_y)$ **do**
    $x_y^a := \text{COLUMNOF}(T_y(i))$
    $x_y^b := \text{COLUMNOF}(\text{other route of } T_y(i))$
    /* Determine whether we can span between x_y^a ***and*** x_y^b ****/***
    $cond_{horz} := \textbf{true}$
    for $j = (x_y^a + 1) \ldots (x_y^b - 1)$ **do**
      $cond_{sort} := (T_y(j) > T_y(i))$ ***/* sorting condition */***
      /* Test for gap or two-track vertical spans */
      $cond_{vert} := \begin{cases} \textbf{true} & T_y(j) = \emptyset \\ \textbf{true} & T_y(j) = T_{y+1}(j) = T_{y-1}(j) \\ \textbf{false} & \textbf{otherwise} \end{cases}$
      $cond_{horz} := cond_{horz} \wedge cond_{sort} \wedge cond_{vert}$
      if $cond_{horz} = \textbf{false}$ **then**
        break from for-loop
      end if
    end for
    if $cond_{horz}$ **then**
      Add (x_y^a, x_y^b) to H_{sol}
      $i := i + (x_y^b - x_y^a)$ ***/* Skip span region */***
    end if
  end for
  return H_{sol}
end function

TABLE I YK = Yoshimura-Kuh; 2SwapC = Constrained 2-sided Swap; LE+KK = Left-edge + Knock Knees; N = # nets; Width = channel width; (new) = {2SwapC, LE+KK } (crossing counts were the same for both routers); * = YK router performed worse than new routers

Design	Nets	Width	Crossings		Bend Loss			Tracks		
			YK	(new)	YK	2SwapC	LE+KK	YK	2SwapC	LE+KK
alu4.0	104	283	214	128	76.54	**63.20**	76.54	10	12	**10**
alu4.1	150	337	258	140	110.40	**101.11**	110.40	* 11	10	**8**
alu4.2	171	372	663	417	125.86	**116.71**	125.86	18	20	**18**
alu4.3	169	368	555	277	124.38	**117.18**	124.38	12	15	**13**
alu4.4	164	368	703	397	120.70	**118.43**	120.70	15	19	19
alu4.5	167	370	747	441	122.91	**118.06**	122.91	14	20	**17**
alu4.6	186	389	752	434	136.90	**134.90**	136.90	15	16	**15**
alu4.7	144	349	598	424	105.98	**101.63**	105.98	13	17	17
alu4.8	157	354	856	480	115.55	**112.59**	115.55	19	**21**	28
alu4.9	153	348	819	393	112.61	**107.71**	112.61	16	**20**	22
alu4.10	179	368	1082	600	131.74	**129.60**	131.74	22	24	**23**
alu4.11	151	355	404	230	111.14	**105.97**	111.14	10	13	13
alu4.12	154	362	530	252	**113.34**	113.70	**113.34**	11	13	**11**
alu4.13	157	364	520	276	115.55	**112.79**	115.55	11	**18**	20
alu4.14	130	324	243	147	95.68	**87.91**	95.68	8	10	**8**
alu4.15	155	340	373	207	114.08	**103.47**	114.08	11	14	14
alu4.16	146	359	296	170	107.46	**100.31**	107.46	10	12	**11**
alu4.17	151	362	238	126	111.14	**105.07**	111.14	9	11	**9**
alu4.18	140	351	485	259	103.04	**96.01**	103.04	10	13	**11**
alu4.19	126	338	214	128	92.74	**87.50**	92.74	8	11	**9**
alu4.20	143	350	340	184	105.25	**99.69**	105.25	11	16	**15**
alu4.21	141	347	277	145	103.78	**97.36**	103.78	9	11	**10**
alu4.22	133	342	236	108	97.89	**93.14**	97.89	8	9	**8**
alu4.23	157	351	561	335	115.55	**112.10**	117.02	11	15	**13**
alu4.24	147	349	418	194	108.19	**105.40**	108.19	9	12	**11**
alu4.25	140	341	562	316	103.04	**100.99**	103.04	14	17	**16**
alu4.26	156	366	504	240	114.82	**109.55**	114.82	15	17	**15**
alu4.27	141	353	486	228	103.78	**99.49**	103.78	15	17	**16**
alu4.28	91	274	103	59	66.98	**56.36**	66.98	7	8	**7**
alu2.0	93	278	279	157	68.45	**56.46**	68.45	17	18	**17**
alu2.1	142	338	311	159	104.51	**100.99**	104.51	7	10	**8**
alu2.2	178	366	780	440	131.01	**125.18**	131.01	19	24	24
alu2.3	147	355	415	253	108.19	**97.97**	108.19	10	14	**13**
alu2.4	178	388	592	316	131.01	**130.33**	131.01	11	13	**12**
alu2.5	177	381	762	462	130.27	**126.21**	130.27	12	20	**18**
alu2.6	149	352	329	199	109.66	**102.39**	109.66	8	11	**10**
alu2.7	175	375	589	363	128.80	**127.68**	128.80	12	17	**13**
alu2.8	160	362	298	200	117.76	**109.97**	117.76	11	13	**11**
alu2.9	91	273	108	62	66.98	**52.21**	66.98	6	8	**6**
ex5p.0	119	343	244	142	87.58	**73.95**	87.58	10	13	**11**
ex5p.1	145	402	211	129	106.72	**97.60**	106.72	* 11	12	**10**
ex5p.2	207	458	755	399	152.35	**145.64**	152.35	18	24	**23**
ex5p.3	183	427	926	598	134.69	**127.22**	134.69	18	22	**18**
ex5p.4	190	439	912	490	139.84	**136.69**	139.84	18	20	20
ex5p.5	209	457	1362	832	153.82	**145.72**	153.82	18	21	**19**
ex5p.6	183	437	634	342	134.69	**127.31**	134.69	15	16	**15**
ex5p.7	195	437	1061	603	143.52	**138.82**	143.52	18	23	**20**
ex5p.8	175	433	550	320	128.80	**124.76**	130.27	12	19	**13**
ex5p.9	173	425	622	318	127.33	**120.91**	127.33	11	15	**14**
ex5p.10	120	340	231	143	88.32	**72.58**	88.32	9	10	10