2019 年度　卒業論文

RHIC-sPHENIX 実験における中間飛跡検出器 INTT 用シリコンセンサーによる宇宙線測定

奈良女子大学　理学部
数物科学科物理学コース　高エネルギー物理学研究室

柴田 実香

2020 年 2 月 21 日

現在、奈良女子大学は、アメリカニューヨーク州のブルックヘブン国立研究所で行われている大型の重イオン衝突実験を目的とした RHIC 加速器を用いた sPHENIX 検出器に導入予定の飛跡検出器の一つ、INTT 検出器の開発に参加してる。INTT グループには、奈良女子大学、理化学研究所、BNL、立教大学、台湾中央大学、海外の研究機関等、複数の研究機関が参加している。今年度は、奈良女子大学から 3 名の学生が参加し、2018 年後期に奈良女子大学に設置されたテストベンチを利用してシリコンセンサーの性能評価にむけ、協力して研究を進めた。性能評価のために行った方法は大きく分けて、キャリブレーションパルスによるノイズ評価と外部トリガーモードを利用した宇宙線による性能評価がある。今回は後者の測定方法と研究結果、研究により浮かび上がった今後の課題と解決について報告する。宇宙線測定の目的は、INTT 用シリコンセンサーとその読み出し回路のデータ読み出し性能を評価することである。トリガーカウンターによる外部トリガーを利用し、シリコンセンサーで測定された宇宙線データの記録、解析を行った。その結果、シリコンセンサーでの宇宙線データでは、エネルギー損失の測定値と予想値がズレていることがわかった。この課題に対して、測定条件の調整と独自開発した宇宙線のエネルギー損失ソフトウェアを用いて原因を検討した。その結果、今まで利用していた予想エネルギー損失の計算式に用いる入力値と設定値の対応関係に誤りがあること、読み出しチップ (FPHX) のチャンネルごとのデータ読み出しにオフセットが存在することを明らかにした。その対応関係を修正後、入力値（ゲイン）とオフセットの最適値を決定した。

目次

図目次

表目次

第 1 章

序論

1.1　素粒子物理学

1.1.1　素粒子の標準模型

身の周りの物質は素粒子で構成されている。素粒子とはそれ以上分けることができない最小単位の粒子であると考えられている。図 1.1 に標準模型（量子色力学（QCD)、電弱統一理論（EW））に表れる素粒子を示した。素粒子はフェルミ粒子とボーズ粒子で構成されている。フェルミ粒子はレプトンとクォークの 2 種類あり、さらにレプトンが、エレクトロン、ミューオン、タウオンの 3 種類と対応するニュートリノの合計 6 種類に分類される。クォークは、アップ (u)、ダウン (d)、チャーム (c)、ストレンジ (s)、トップ (t)、ボトム (b) の 6 種類に分類される。これらが組み合わさることで様々な物質を分類することができる。またこれらのクォークをつなぎとめる力の媒介粒子としてグルーオンが存在している。グルーオンは中性子や陽子中にクォークを束縛する。また π 中間子は原子核中に中性子や陽子は束縛する力を媒介している。

1.1.2　QGP

原子が正イオンと電子に電離した状態をプラズマ状態という。プラズマ状態は固体、液体、気体に次ぐ第四の状態相であり、物質がイオン化エネルギーに相当する高温状態に達した時に実現する。同様に核子に閉じ込められているクォーク間のグルーオンによる結合が長くなることで、隣の核子内のクォークとも結合できるようになり、元の核子中のクォークと隣の核子のクォークの区別がつかなくなる。この現象がたくさん起こることで、クォークとグルーオンが核子による閉じ込めから解放されたように見える状態をクォーク・グルーオン・プラズマ（QGP）という。QGP は高温、高密度下であったとされる宇宙初期のビックバンから数 μ 秒後に実現していたと考えられている。従って QGP 現象の理解を進めることは初期宇宙発展の解明につながる重要な研究テーマである。現在 QGP は大型加速器を利用した原子核衝突実験で人工的に生成できることが確証されている。RHIC 加速器による重イオン衝突実験で、史上初めて QGP の存在が確証された。 [8]

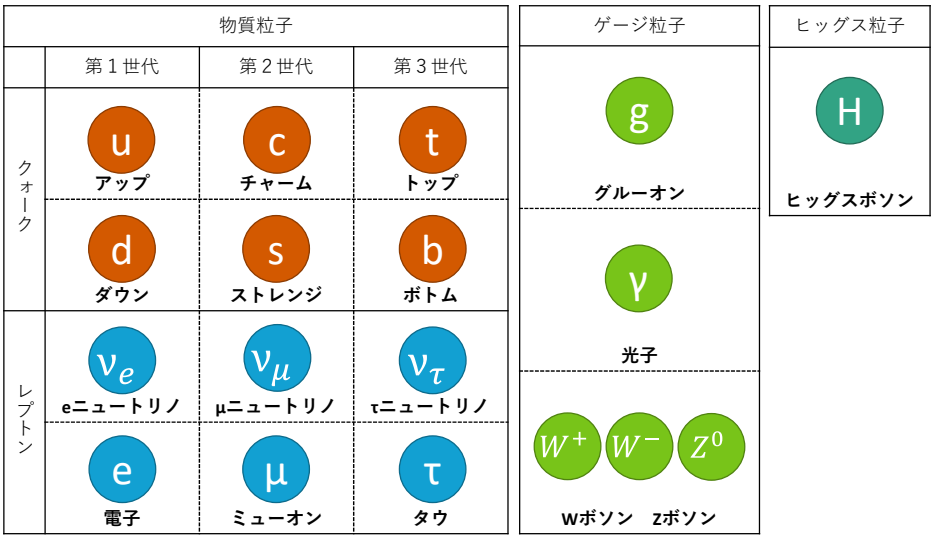

図 1.1　素粒子の標準模型で扱われる素粒子

1.1.3　宇宙線

宇宙線とは宇宙空間を高エネルギーで飛び回る放射線のことである。これらのほとんどは陽子で一次宇宙線と呼ばれ、多量に地球に降り注いでいる。これらが大気圏を通過すると、大気中の原子核と反応し π 中間子など大量の 2 次粒子を生成する。生成された中間子は高速で周りの原子核に衝突することで、さらに粒子の数を増やしながらエネルギーを落としていく。この現象を空気シャワーという。生成された中間子の一種である π 中間子は寿命が短いため、すぐに崩壊し μ 粒子として地上に降り注ぐ。この地表付近で観測できる宇宙線を二次宇宙線という。図 1.2 は宇宙線 μ 粒子の空気シャワーのイメージ図である。この粒子はで地表付近で再現できる最大のエネルギー (0.4GeV) をもつ粒子で、1 分間に 1 平方センチメートル当たり約 1 個観測できる。μ 粒子を地表付近での実験に利用することは、大型の加速器を使わずに MIP（最小電離粒子）のような損失エネルギーを得ることにおいて非常に有効である。 [8]

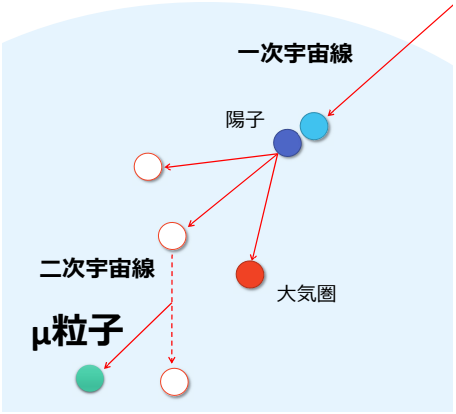

図 1.2　宇宙線の空気シャワー

1.2　研究背景

1.2.1　Relativistic Heavy Ion Collider (RHIC)

Relativistic Heavy Ion Collider (RHIC) は世界で初めての重イオン衝突型加速器であり、アメリカのニューヨーク州ロングアイランドにあるブルックヘブン国立研究所 (BNL：Brookhaven National Laboratory) に建設されている。図 1.3 は RHIC の全体像である。QGP を生成しその性質を研究することを目的に、2000 年から稼働を開始した。主な衝突原子核は金原子核対で、衝突時の核子 1 つあたりの重心系エネルギーは 200GeV である。その他にも、ヘリウム原子核や陽子といった様々な原子核衝突実験を行っており、陽子・陽子衝突での最大重心系エネルギーは 510GeV である。RHIC には 2 つの環状のビームラインが建設されており、それぞれ時計回りと反時計回りの周長 3.8km の加速器リングである。ビームラインが交差する衝突点は計 6 カ所設けられている。ビームは 106ns(9.4MHz) の間隔でバンチ構造を持っており、各リングに 120 バンチが蓄積されている。 [11]

1.2.2　sPHENIX 実験

sPHENIX 実験はブルックヘブン国立研究所の RHIC を用いて行われる新たな実験である。2000 年から 2016 年まで同地で行われていた PHENIX 実験を高度化した次世代実験で、2023 年から稼働が予定されている。図 1.4 は sPHENIX 検出器の全体像を表す。この実験では、QGP における輸送係数の温度依存性や色電荷のデバイ遮蔽長について解明すべく、ジェット、ジェット相関、および Υ を測定する計画である。この測定では、高い効率と大きな検出範囲で、ハドロンと電磁粒子のエネルギー測定と精密な飛跡検出を行うことができる。ジェットや光子の観測を行うため、20 週間の RHIC 加速器の稼働により金・

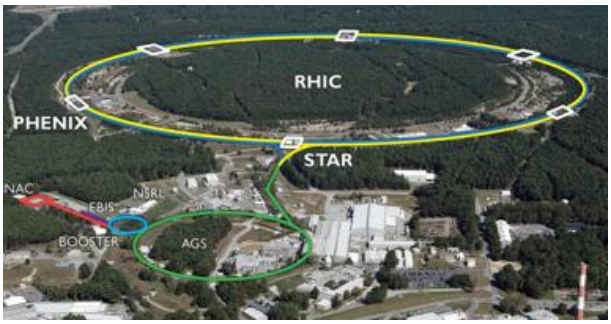

図 1.3　Relativistic Heavy Ion Collider(RHIC)

金衝突を 5×10^{11} 回行い、Υ 粒子の 3 つの状態の区別に十分な質量分解能に達することができる。ハドロンの熱量測定は陽子・陽子衝突や陽子・金衝突においてバイアスのない状態のジェットをトリガーし、全システムにおいてジェットのバイアスのない観測することが可能である。 [5]

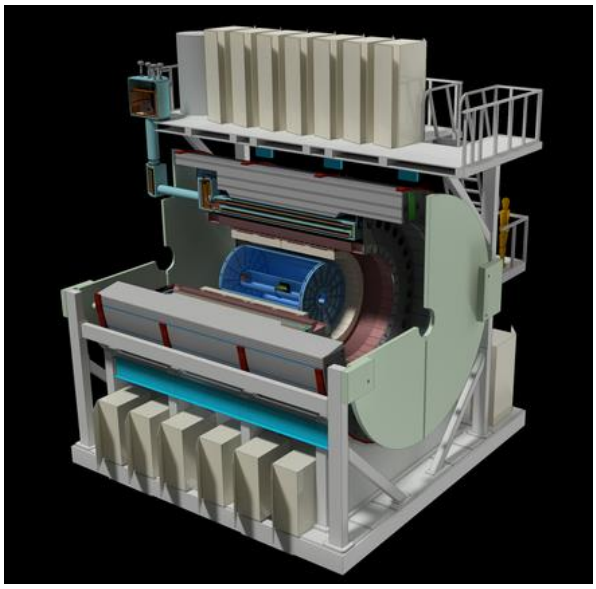

図 1.4　sPHENIX 検出器の全体像

1.2.3　飛跡検出器群

sPHENIX 検出器における飛跡検出器群は MVTX, INTT, TPC の 3 つで構成されている。 [5]

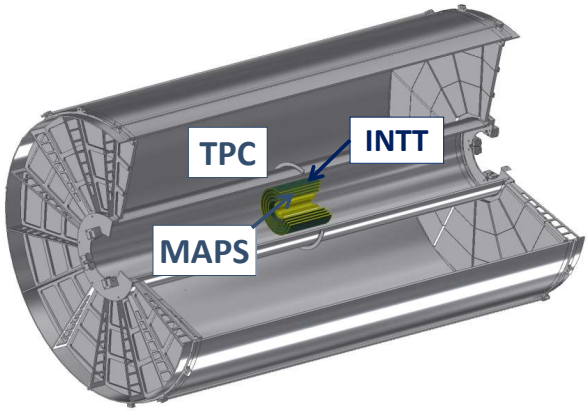

図 1.5　飛跡検出器群の全体像（MVTX, INTT, TPC）

Monolithic-Active-Pixcel-Sensor-based Vertex Detector(MVTX)

MVTX は、MAPS(Mononlithic Active Pixcel Sensor) を用いた 30μm ピッチの半導体ピクセルを用いた三層構造の半導体検出器である。3 つの飛跡検出器のうち最も内側に位置している。方位角方向に対しては 2π、ラピデティ方向に対しては $|\eta| \leq$1.1, 衝突中心からビーム軸方向に ±10cm の範囲を検出する。高精度の飛跡検出が可能であり、衝突点と生成された粒子の最近接距離を測定することで重クォークの検出を行う。 [5]

INTermediate Tracking detector（INTT）

MVTX と TPC の中間 (ビームパイプから 6〜12cm) に位置し、全方位をカバーする 2 層構造の検出器である。方位角方向に 78μ ピッチで半導体ストリップが配備されている。 [5] INTT の目的は、MVTX と TPC 間の飛跡を繋ぎ、運動量分解能を上げるとともに、各トラックに 1 ビームバンチ以下の詳細な時間情報を与えることである。例えば、粒子多重度の高いイベントの場合、INTT によって再構成された飛跡と MVTX, TPC の飛跡のマッチングを調べることで、より確からしい飛跡を選択することができる。図 1.6 の左は INTT のビーム軸に垂直方向からの断面図、図 1.6 の右はビーム軸方向からの図である。 [6]

Time Projection Chamber（TPC）

3 つの検出器のうち最も外側 (ビームパイプから 20m〜78m) に位置し、衝突中心から ±1m ずつを覆うガス検出器である。読み出しパットからの 2 次元情報に加えてドリフト時間からビーム軸方向の位置情報を加えることで、3 次元の位置を測定することができる。図 1.7 は TPC 検出器の全体像である。 [5]

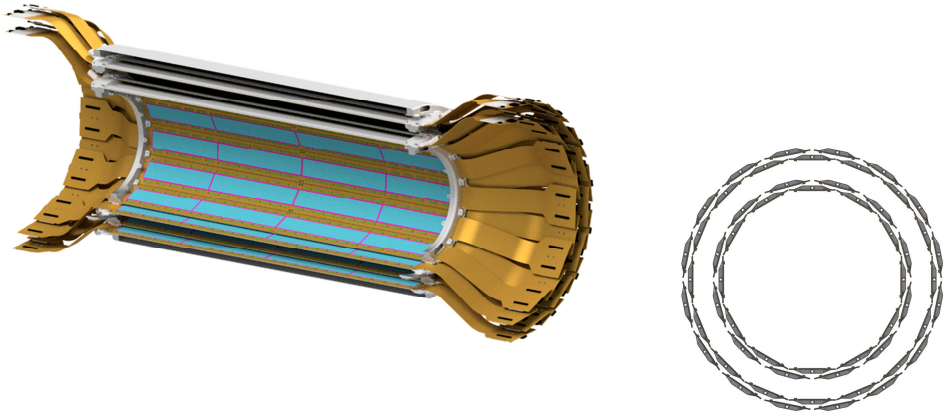

図 1.6　（左図）INTT 検出器, (右図)INTT のビーム軸方向からの図

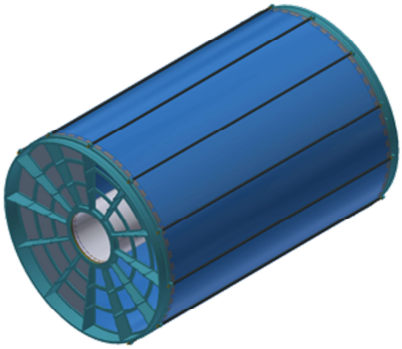

図 1.7　TPC 検出器

1.3　研究目的

現在、奈良女子大学は、INTT 検出器の開発に参加している。2018 年後期から、奈良女子大学高エネルギー物理学研究室において、シリコンセンサーの性能評価を目的としたテストベンチの導入と設置が始まり、無事に設置が完了した。本研究の目的は、そのテストベンチを用いて宇宙線を測定し、INTT 用シリコンセンサーとその読み出し回路のデータ読み出し性能の評価を行うことである。

第 2 章

中間飛跡検出器 (INTT)

2.1　半導体

半導体とは、結晶構造をした、エネルギーバンド構造をとる物質一般のことで、電気伝導性の良い金属などの導体と電気抵抗率の大きい絶縁体の中間的な抵抗率を持つ。代表的なものの 1 つがケイ素（Si）であり、シリコンともいわれる。半導体には n 型と p 型の 2 種類があり、それぞれのキャリアが 3 価の自由電子と 5 価の正孔である。n 型の方が p 型よりも電子を多く有している。 [12]

2.1.1　半導体の性質

半導体の性質の 1 つとして、抵抗が大幅に変化することがある。例としてシリコンについて考える。シリコンは周期律表 4 族に含まれる 4 本の結合手を持つので、純粋な結晶はダイヤモンドと同じ構造である。従って、純粋なシリコンの結晶はほとんど電気を通さない。しかし、ごく少量の不純物が結晶中に加わることで、電気伝導性が生まれる。抵抗率でいって 1 万分の $1\Omega \cdot \mathrm{cm}$ の導電物質から、$1\mathrm{M}\Omega \cdot \mathrm{cm}$ の絶縁体に近い物質まで変化する。さらに、電界を加えることによっても抵抗率が大きく変化する。この性質は絶縁体を挟んで半導体に電界を加えた場合のみ現れる。 [12]

2.1.2　半導体を用いた荷電粒子検出原理

PN 接合した半導体の接合面には正孔が P 側から N 側へ、電子が N 側から P 側へ移動することにより生じる空乏層が存在している。この空乏層にはキャリアが存在していない。電圧をかけた PN 接合半導体の空乏層を荷電粒子が通過すると、荷電粒子の損失エネルギーにより、経路に沿って電子・正孔対が生じる。これらは電圧により引き寄せられ、チップに読み出される。

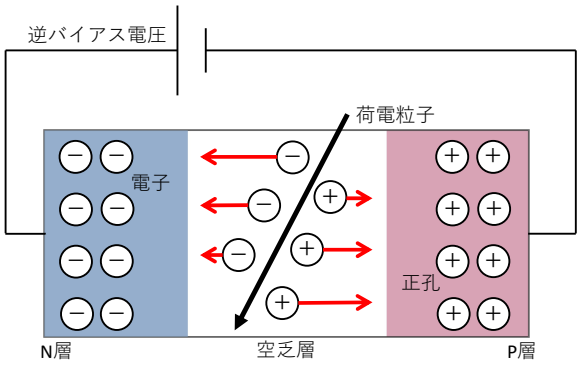

図 2.1　PN 接合した半導体を用いた荷電粒子検出原理

2.1.3　シリコン半導体におけるミューオンのエネルギー損失

荷電粒子が物質を通過するとエネルギーを損失する。この損失エネルギーは通過する物質の種類とその通過距離、入射粒子の運動量に依存している。従って、このエネルギーを測定することは、高エネルギー実験において、粒子の種類の特定だけでなく、時間を遡って親核子を推定したり、衝突時の崩壊反応を再現可能するといった重要なファクターとなる。

高い運動量を持つ荷電粒子が物質中を通過すると、物質を構成する原子や分子中の電子と相互作用し、物質がイオン化するためにエネルギーを失う。このエネルギー移行は運動量移行により表される。

一般的に、損失はベーテブロッホの式 (式 2.2) に従い、単位は $\mathrm{MeVg^{-1}cm^2}$ である。ここで使われるパラメータを表 2.1 に示す。各値は PDG 2019 Passage of Particles Through Matter [2] と PDG 2019 AtomicNuclearProperties [3] を参照されたい。

$$\left\langle -\frac{dE}{dx} \right\rangle = Kz^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} ln \frac{2m_e c^2 \beta^2 \gamma^2 W_{max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right] \tag{2.1}$$

表 2.1　微分断面積計算に用いるパラメター

記号	定義	用いる値・単位
K	：$4\pi N_A r_e^2 m_e c^2$	$0.307075\mathrm{MeVmol}^{-1}\mathrm{cm}^2$
	(微分断面積 dE/dX の定数)	
z	：入射粒子の電荷	1(muon)
Z	：吸収物体の原子番号	14(Si)
A	：吸収物体の原子量	28.0855(3)g/mol(Si)
ρ	：吸収物体の密度	$2.329\mathrm{g/cm}^3$(Si)
β	：入射粒子の $\frac{v}{c}$	
γ	：$\frac{1}{\sqrt{1-\beta^2}}$	
$\beta\gamma$	：	~ 10(1GeVmuon)
$m_e c^2$	：電子の質量 $\times c^2$	0.5109989461(31)MeV
W_{max}	：単一衝突での最大エネルギー移行	$\frac{2m_e c^2\beta^2\gamma^2}{1+2\gamma m_e/M+m_e/M^2}$ [MeV]
I	：平均励起ポテンシャル	173.0[eV](Si)
$\delta(\beta\gamma)$	：イオン化損失エネルギーの密度効率補正	

式 2.2 は Z/A に比例することから、値が吸収物質によることがわかる。また、β の関数であり入射粒子の電荷の 2 乗に比例することから、入射粒子の識別に利用できることがわかる。

図 2.2 はベーテブロッホの式から求められる単位厚さあたりのシリコン中での μ 粒子運動エネルギーとエネルギー損失の関係で、横軸が μ 粒子の運動エネルギー、縦軸が単位密度単位長さあたりのエネルギー損失を示す。これより、入射粒子のもつ運動量エネルギーの大きさによって損失エネルギーが異なることがわかる。

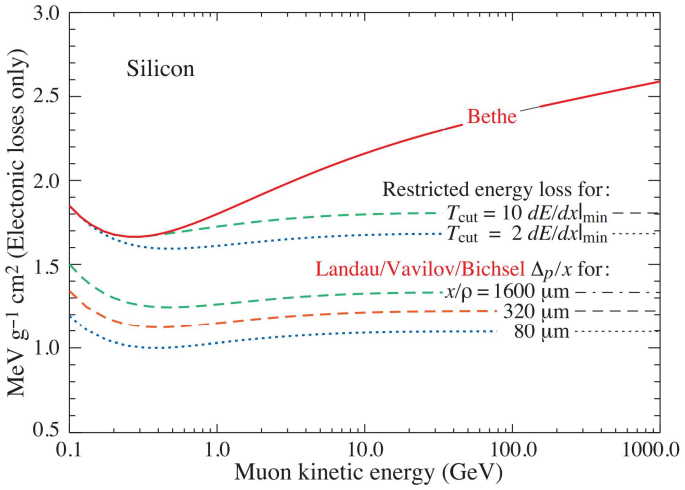

図 2.2　単位厚さあたりのシリコン中での μ 粒子運動エネルギーとエネルギー損失の関係

荷電粒子が厚さ x の物体を通過する場合のエネルギー損失の最確値は、ランダウ分布によって表される。最確値は式 2.2 で求められる。

$$\Delta_p = \xi \left[ln\frac{2mc^2\beta^2\gamma^2}{I} + ln\frac{\xi}{I} + j - \beta^2 - \delta(\beta\gamma) \right] \tag{2.2}$$

ここで $\xi = (K/2)\langle Z/A \langle z^2 (x/\beta^2)$MeV、厚さ x の単位は gcm^{-2}、$j = 0.200$ である。図 2.3 は図 2.2 を Δ_p/x で規格化したもので、500MeV のパイオンがシリコン中を通過した際の Straggling 関数を表している。図 2.3 より、物質の厚さが薄いほど単位飛程あたりに失う運動エネルギーが小さいほうへ偏ってい

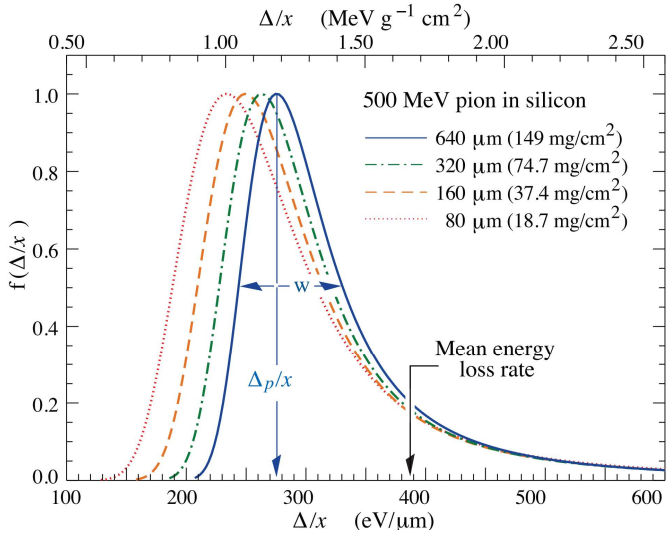

図 2.3　500MeV のパイオンのシリコン中での Straggling 関数の変動

くことがわかる。また、エネルギー損失の最頻値も低くなる。INTT 用シリコンセンサーに使われるシリコンの厚さは 320μm と比較的薄いため、エネルギー損失の確率分布は小さいほうへ偏り、大きなエネルギーを落とすイベントは少なくなる。例えば、1 GeV の μ 粒子が 320μm 厚のシリコンを通過した場合、単位密度単位長さあたりの損失エネルギー（最頻値）は、式 2.2 より

$$\Delta_p \sim 1.15\mathrm{MeVg}^{-1}\mathrm{cm}^2 \tag{2.3}$$

であり、320μm で損失するエネルギーは

$$\Delta_p \sim 1.15\mathrm{MeVg}^{-1}\mathrm{cm}^2 \times 0.032\mathrm{cm} \times 2.33\mathrm{g/cm}^3 \sim 0.086\mathrm{MeV} \tag{2.4}$$

と計算できる。

2.2 シリコンモジュールの構造

INTT 用シリコンストリップ検出器のセンサー部分はシリコンモジュールと呼ばれ、主に、シリコンセンサー、FPHX 読み出しチップ、HDI、Bus Extender の 4 つで構成されている。

2.2.1 シリコンセンサー

INTT 用シリコン検出器は、シリコンストリップセンサーを採用されており、1 チップ分が図 2.4 のように構成されている。INTT 用シリコンセンサーの 1 モジュールは、FPHX チップ 26 個で構成される。最もビームライン近くに位置する最内層から順に L0〜L1 の順でナンバリングされており、実際の実験に導入される時には、バレル上のビームラインを中心にして複数のセンサーモジュールが取り囲むように配置する。センサーの位置による粒子のヒット数の違いからストリップ長の異なる 2 種類のセンサー（TypeA, B）が組み合わさっている。センサーの種類に関係なく、1 モジュールあたり 26 個のストリップ型シリコンセンサーが配置されている。また 26 個の FPHX チップがそれぞれ各ストリップ型シリコンセンサーの横に配置されている。また、シリコンセンサーは n 型と p 型半導体で構成されている。

2.2.2 HDI（High Density Interconnect）

HDI は、FPHX チップへの入出力配線、センサーおよび FPHX チップへの電源供給を行う基板である。ROC とはバスエクステンダーを通して接続される。

2.2.3 Bus Extender

Bus Extender は HDI、ROC 間を接続するためのケーブルである。HDI と ROC 間は距離が離れており、狭いことから長さに由来する問題やフレキシビリティ確保に伴うの問題の解決が必要とされている。現在のバスエクステンダーの長さは約 1.2m である。 [5]

2.3 データ読み出し回路

sPHENIX で用いられるデータ読み出しシステムは PHENIX で使用したものを再利用する。図 2.6 に、そのデータ読み出しシステムを示す。sPHENIX 実験は複数のサブシステムからなるため、データ量が膨大になるが、各システムのデータが sPHENIX の DAQ（Data Acquiaition）システムに全サブシステムで共通の同期信号が RHIC から供給されている。この同期信号を Beam Clock（BCO）といい、その頻度は 9.4MH z（106ns）である。BCO は MTM（Master Timing Module）と呼ばれるタイミング統括回路、GT1（Global Level 1）と呼ばれるトリガー統括回路へ送る。GL1 では、各サブシステムからのトリガーに応じた間引きが行われ、データ収集判断のためのトリガー（Level 1 Trigger）を発行す

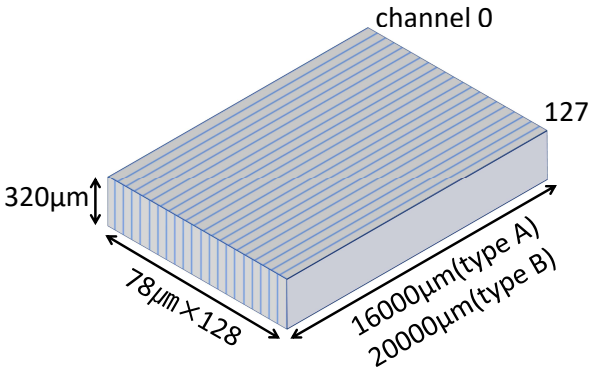

図 2.4　INTT 用シリコンセンサー（1 チップ）の模式図

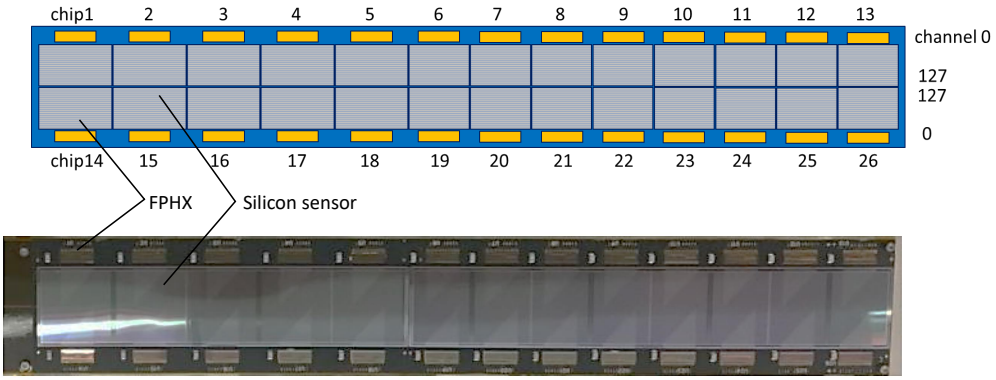

図 2.5　INTT 用シリコンセンサーモジュールの模式図（上）と写真（下）

る。Level 1 Trigger は、MTM、GTM を介して各サブシステムの FEM（Front End Module）へ返される。この Level 1 Trriger を受け取ったタイミングでの INTT シリコンモジュールから送られたデータが、DCM（Data Collection Module）と呼ばれるデータ収集回路に送られ、バッファの後各サブシステムのイベント情報が統合されたデータとして出力される。以下データ読み出しシステムににおける各モジュールを説明する。 [10]

2.3.1　FPHX

FPHX チップは PHENIX 実験で使用されていたシリコンセンサー用の読み出しチップであり、sPHENIX 実験における INTT 用シリコンセンサーの読み出しチップとしても採用されている。それぞれのチップが 128ch の読み出しチャンネルを持ち、各チャンネルで波形整形、3bit の ADC 機能を持つ。ユーザーは、外部からゲイン値や ADC 閾値、各チップ、チャンネルのマスクなどの稼働条件を制御でき

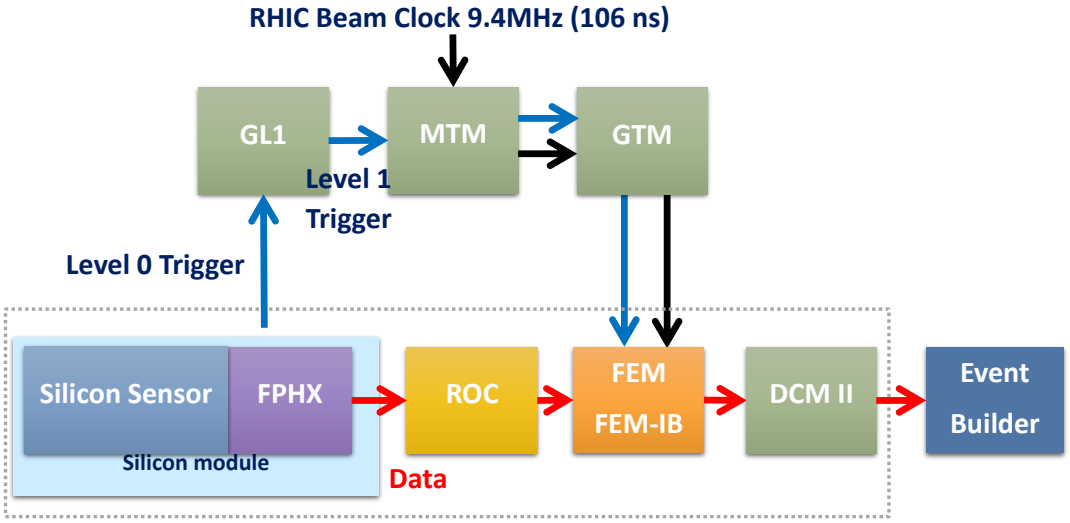

図 2.6　sPHENIX 実験での INTT のデータの読み出しシステム

る。また、本チップからタイムスタンプ、ADC 値、ヒットチャンネルが出力される。 [10]

2.3.2　Read Out Card (ROC)

FPHX から出力されたデータは下流の読み出し回路 Read Out Card(ROC) へ送られる。図 2.7 は接続後の ROC である。ここでは複数の FPHX からのデータの同期と結合を行い、下流のエレクトロニクスへと転送する。また。FPHX チップへの測定条件や閾値といった情報を指定されたチップへ転送する機能や FPHX チップの読み出し回路の動作確認を行うためのキャリブレーションパルスを発生させる機能、センサーモジュールへの電源供給機能も持つ。 [10]

2.3.3　Front End Module (FEM)

ROC からの出力データは Front End Module(FEM) に入力される。図 2.8 の右側に位置するボードが FEM である。VME 規格の読み出しボードで、FPHX からのデータをまとめ、PHENIX 共通のフォーマットに変換し、他の検出器の情報と結合を行うモジュールに転送する役割を持つ。FEM1 枚は ROC の出力の半分を賄っており、ROC1 枚の出力には FEM2 枚が必要である。一方、ROC への命令信号は 1 枚で賄うことができる。また、FPGA が搭載されており、トリガーシステムの構築が可能である。 [10]

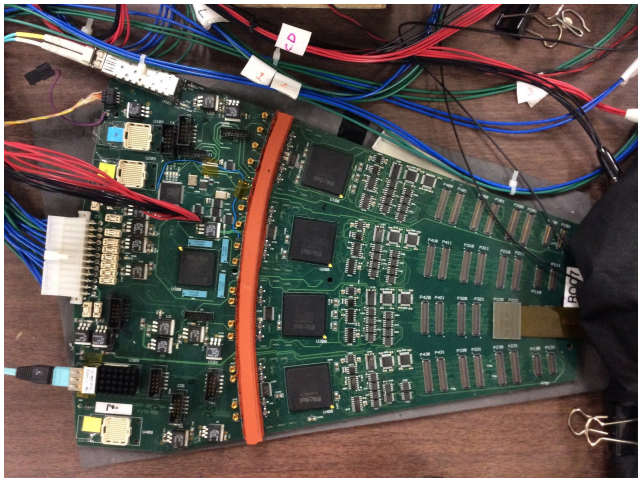

図 2.7　Read Out Card (ROC)

2.3.4　FEM - Interface Board (FEM-IB)

FEM 全体を制御するモジュールで、FEM 同様 VME 規格の読み出しボードである。図 2.8 の左側に位置するボードが FEM-IB である。本モジュールは主に検出群全体を統括するクロック信号やトリガー信号、FEM 制御信号を受け取る。 [10]

2.4　テストベンチでの INTT シリコンモジュールのテスト

INTT の検出器の性能評価や読み出しシステムの動作確認として、2018 年に奈良女子大学に構築されたテストベンチを利用し、キャリブレーションモードを利用したキャリブレーションテストと外部トリガーモードを利用した宇宙線測定の 2 つを行った。図 2.10 は奈良女子大学の INTT テストベンチである。また、奈良女子大学の INTT テストベンチでは High Volutage NIM モジュールを使用している。テストベンチに使用した NIM モジュール（図 2.9）の機能を表 2.2 に示す。

2.4.1　キャリブレーションテスト

キャリブレーションテストはテストベンチ全体の動作確認を行うために行う。キャリブレーションテストでのセットアップ状況は図 2.11 のとおりである。ROC に HDI を接続した状態で ROC より生成したシリコンセンサーからの出力を模したアナログ信号（test pulse）を FPHX チップのテストパルス入力に入れ、FPHX チップ内で変換されたデジタルのヒット信号を確認する。このヒット信号は再び HDI を通り ROC に送られ、FEM と FEM-IB 通過後 PC に送られる。ROC で生成されるテストパルスは 1 チップ、1 チャンネルあたり 1 0 個であり、複数のパルス波高が全チップ、全チャンネルへ順次スキャンさ

図 2.8　左：FEM - Interface Board (FEM-IB)、右：Front End Module (FEM)

表 2.2　使用した NIM モジュールとその機能

NIM 規格モジュール	機能
DUAL GATE GENERATOR	入力されたデジタル信号の width を調節し、delay を与える。
QUAD DISCRIMINATOR	任意の閾値以上の電圧パルスの信号のみをアクセプトする。
DUAL 4-FOLD COINCIDENCE	シリコンセンサー 2 台を使用した時に AND 信号を出力する。
LOGIC LEVEL ADAPTER	NIM 規格のデジタル信号をロジック回路のフォーマットである TTL 信号に変換
DUAL HIGH VOLTAGE POWER SUPPLY (POSITIVE)	シリコンセンサーに電圧を印加する。
DUAL HIGH VOLTAGE POWER SUPPLY (NEGATIVE)	シンチレーションカウンタに電圧を印加する。
100MHz CLOCK GENERATOR	任意の周波数を持った信号を出力することができる。
8CH VISUAL SCALER	入力端子が 8ch あり、それぞれエントリー数をカウントすることができる。

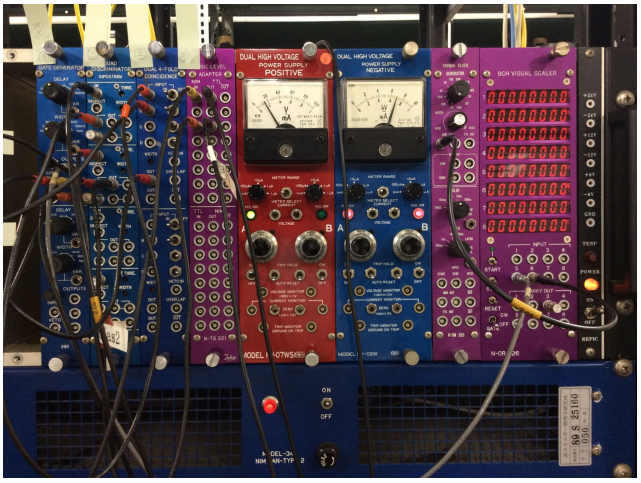

図 2.9　使用した NIM モジュール

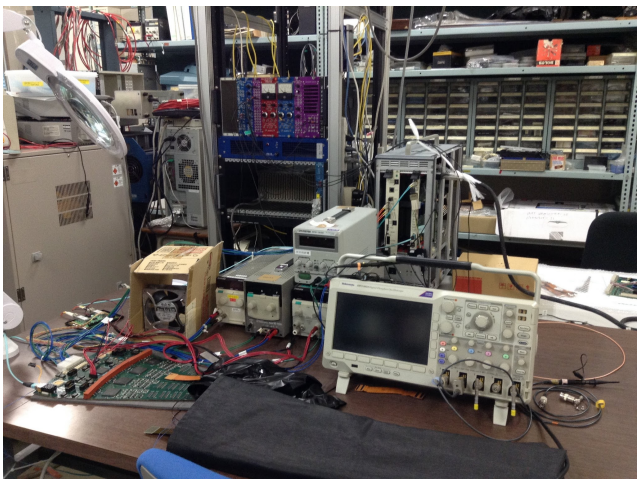

図 2.10　INTT テストベンチの設置状況

れ全てチャンネルに対し入力される。キャリブレーションテストでの DAC 閾値設定は表 2.3 のとおりある。DAC 閾値設定については、 3.2.4 で詳しく述べる。このテストにおいて入力テストパルスの波高を amplitude と呼ぶ。

2.4.2　キャリブレーションテスト結果

図 2.12 の右は、1 チップでの入力 amplitude と出力 ADC の相関関係である。このヒストグラムはボックス型と呼ばれ、ひし形のようなヒストグラムの形がヒット数を表している。この図より、chip 1 において入力 amplitude と出力 ADC は DAC 設定値に基づいた比例関係にあることがわかる。この図を 1 モジュール分（chip1〜26）まとめて表示したものが図 である。上段は左が chip1 で一番左が chip13、下段は左が chip14 で一番右は chip26 であり、全チップで比例相関関係が確認できることから正常に動作

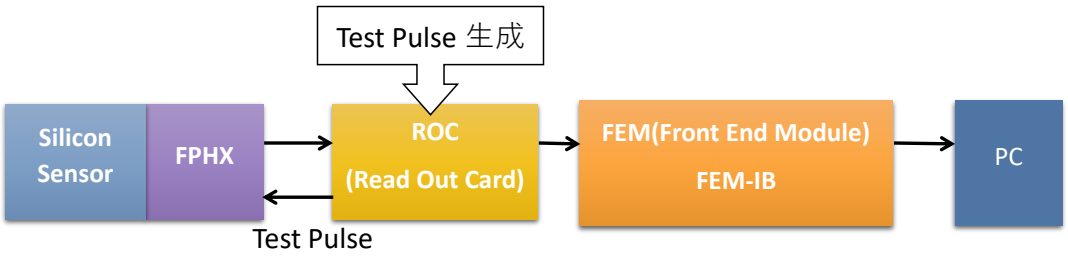

図 2.11　キャリブレーションモードにおける読み出し回路

表 2.3　キャリブレーションモードにおける DAC 閾値設定

DAC	DAC 設定値	対応電圧 [mV]
DAC0	20	290mV
DAC1	25	310mV
DAC2	30	330mV
DAC3	35	350mV
DAC4	40	370mV
DAC5	45	390mV
DAC6	50	410mV
DAC7	55	430mV

しているといえる。図 の左は、1 チップでの入力 amplitude とヒットチャンネル位置の相関関係であり、色がヒット数を示している。この図より、DAC0 閾値付近でのヒット数が徐々に増えていき、ある閾値を超えると常に同じヒット数であることがわかる。この図を 1 モジュール分（chip1〜26）まとめて表示したものが図 2.14 である。上段は左が chip1 で一番左が chip13、下段は左が chip14 で一番右は chip26 であり、全チップ全チャンネルで同等のヒット数が確認できるので正常に動作しているといえる。

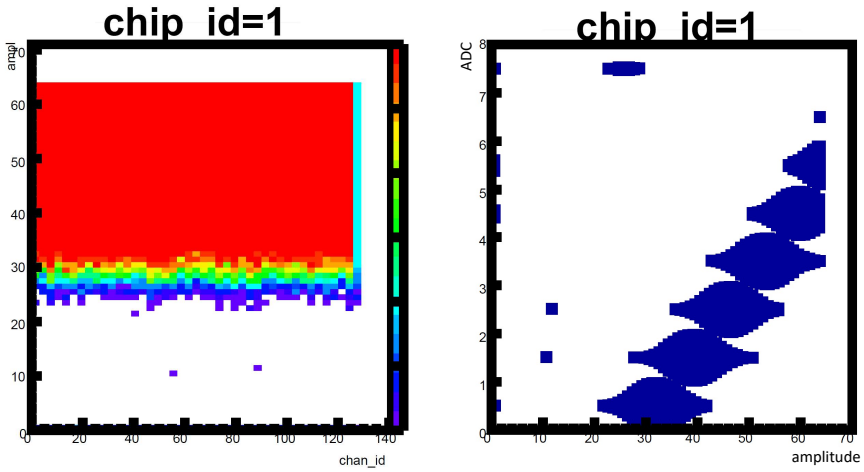

図 2.12　(右図)chip 1 でのテストパルスの入力波高（amplitude）と出力 ADC の相関分布, 横軸：入力波高（amplitude）, 縦軸：出力 ADC, (左図)chip 1 での全チャンネルでのテストパルスのヒット数分布, 横軸：チャンネル番号, 縦軸：入力波高（amplitude）

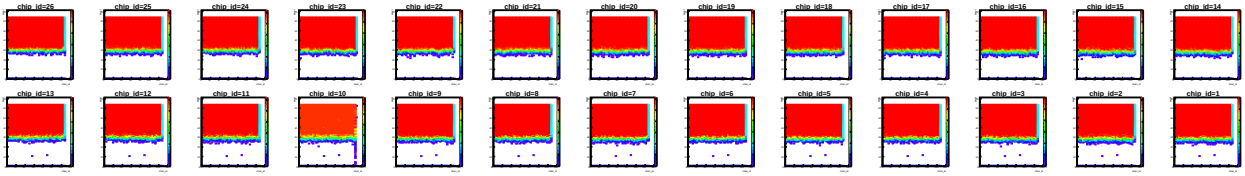

図 2.13　チップごとの全チャンネルでのテストパルスのヒット数分布

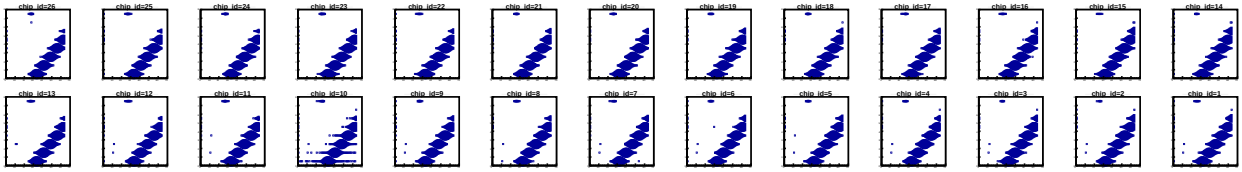

図 2.14　チップごとのテストパルスの入力波高（amplitude）と出力 ADC の相関分布

第 3 章

宇宙線測定

3.1　目的

本研究では宇宙線を利用し、シリコンセンサーの性能を評価する。宇宙線は実験室内で再現できる最大エネルギーを持ち、μ 粒子がシリコンを通過した時のエネルギー損失は 100MeV〜1GeV 程度で観測されるので、sPHENIX 実験での条件に近い状態を作り出すことができる。

3.2　測定方法

本実験では、シリコンセンサーからの信号と、シンチレーターからの信号の AND 信号をトリガー信号として採用し測定を行った。2 つのシンチレーショカウンターの AND 信号を採用している。本測定は宇宙線のレートが低く、長時間の測定が必要となるため、宇宙線由来でないノイズを拾う確率が大きい。AND 信号を利用することで、このノイズの削減することができる。AND 処理は FEM の基板上に乗った FPGA で行われる。また、低エネルギー粒子はトリガー用のシンチレーター内で多くのエネルギーを損失し静止しすることで、さらに宇宙線レートが下がってしまう可能性がある。宇宙線は高エネルギー粒子であるため、シンチレーターを通過した後もエネルギーを有することができる。従って、この測定方法は高エネルギーの測定対象に適している。

今回使用したシンチレーションカウンタの一仕様と性能については下記表 5.1 にまとめた。用いたシンチレータは 2 種類存在し、1 つはシリコンセンサー全体を、もう一つは一部を覆っている。シンチレーションカウンタの信号レートは、NIM 規格の Visual Scalor を用いて一定時間あたりの信号数をカウントし、概算し求めた数値である。

3.2.1　読み出し回路

宇宙線測定の読み出しシステムは図 3.1 と図 3.2 の通りである。図 3.1 は外部トリガーの生成回路をしめす。宇宙線がシンチレーションカウンターを通過すると NIM モジュールの DISCRIMINATOR に電圧パルスが送られる。ここで閾値以上である信号のみ COINCIDENCE に送られる。AND が取れた

表 3.1　使用したシンチレーションカウンタの一仕様と性能

シンチレーション	シンチレーターのサイズ			光電子倍増管	陽極－陰極間
カウンター	縦	横	厚み	品番	印加電圧
A	128 mm	24 mm	5 mm	H3178-51	-1150 V
B	20 mm	30 mm	10 mm	H3165-10	-1400 V

電圧パルス信号は DUAL GATE GENERATOR で波形整形される。この時、外部トリガー信号をシリコンセンサーにタイミング同期させるために、外部トリガー信号の幅や時間を調節する。最後に Level Adapter で NIM 規格のデジタル信号を TTL 規格に変換されたものが宇宙線外部トリガー信号である。さらに、図 3.2 はシリコンセンサーからのデータ読み出し回路をしめす。シリコンセンサーで信号が読み取られると、FPHX チップに送られ、電荷パルス信号は増幅・整形後、デジタル化 (ADC) される。デジタル信号は ROC で集積・同期され、シリコンセンサーからのトリガー信号として FEM に入力される。この信号と外部トリガー信号が FEM に同時入力された場合、FEM ロジック回路でトリガーが発行され、デジタル信号を (ハードディスクに) 記録する。この信号は sPHENIX 共通のデータフォーマットに変換された後、PC に入力される。

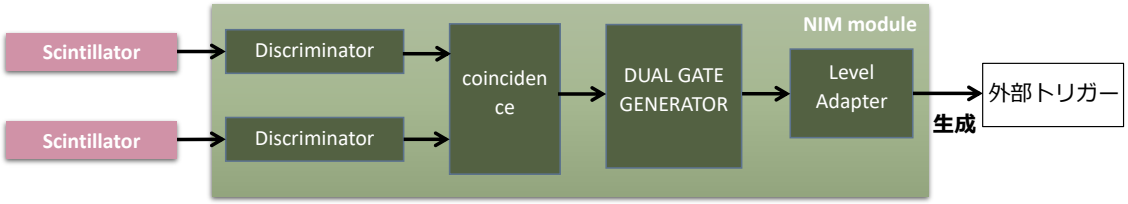

図 3.1　宇宙線測定における外部トリガー生成回路

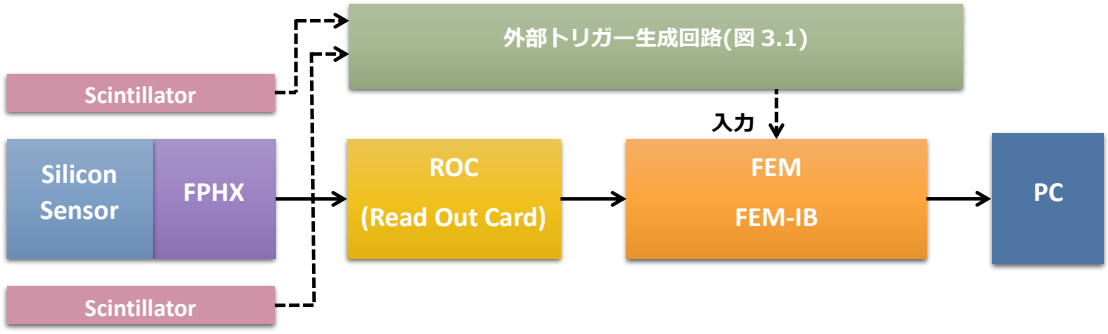

図 3.2　宇宙線測定におけるデータの読み出し回路

3.2.2　宇宙線トリガーのタイミング調節

シリコンセンサーから FEM で内部トリガー判断が行われる間、FPHX チップにおける信号処理時間、ROC 基板での信号処理時間分、信号が遅延する。従って、この遅延時間分シンチレーションカウンターからの信号を FEM のロジック回路へ入るまでに遅らせ、内部トリガー信号と AND をとれるようにタイミングに合わせる必要がある。処理時間は FPHX チップでの信号処理時間 0.9μs と ROC 基盤での信号処理時間 1.7μs を足し合わせた 2.6μs である。この遅延処理には NIM 規格の DUAL GATE GENERATOR モジュールを用い、幅や遅延時間を調節した。シンチレーションカウンターのアナログ信号に対するスレッショルドは 150mV とした。この値に設定した理由は、印加電圧を上げることで発生する波形の乱れにより、宇宙線でない低波高の信号をトリガーとして採用してしまうのを防ぐためである。

オシロスコープ上で確認した波形を図 3.3 に示す。各波形が表す出力信号は下に示す通りである。

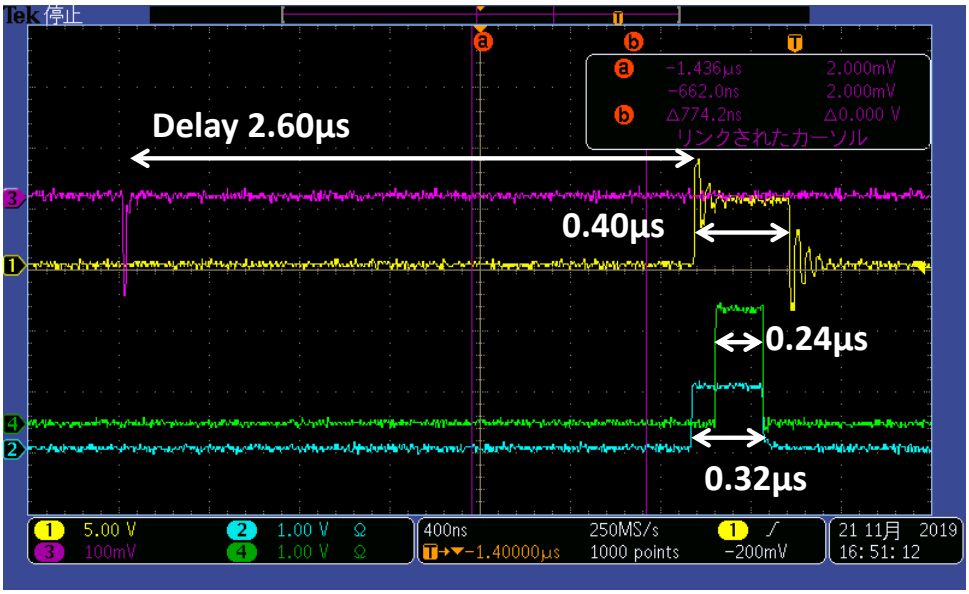

図 3.3　オシロスコープ画面出力信号

オシロスコープ画面出力信号

赤：　シンチレーションカウンターからの宇宙線アナログ信号

黄：　FEM に入力されるシンチレーションカウンターからの宇宙線トリガー信号（TTL 規格）

緑：　FEM に入力されるシリコンセンサーからのトリガー信号

青：　2 つのトリガー信号 (2. 黄色と 3. 緑) の AND 処理により FEM で発行されたトリガー信号

赤のシンチレーションカウンターからの宇宙線アナログ信号が発行されてから、黄色の FEM に入力されるシンチレーションカウンターからの宇宙線トリガー信号が発行されるまでに約 2.6μs かかっているこ

とがわかる。シンチレーションカウンターからの宇宙線トリガーの方形波の幅は約 0.40μs で 5beamCLK 分に等しい。このデジタル信号と緑のシリコンセンサーからのトリガーのデジタル信号が同時に発行され、AND 処理による青のデジタル信号が発行されていることから、FEM での宇宙線トリガー判断が正常に機能しており、宇宙線トリガー信号とシリコンセンサーからのトリガー信号の AND が取れていることがわかる。従って、FEM の AND 処理に最適な外部トリガー信号の幅や時間を調節が行えたといえる。

3.2.3 セットアップ

本測定では、トリガー用シンチレーションカウンター 2 つでシリコンセンサー 1 つまたは 2 つを挟んで設置した。シリコンセンサー 2 つによる同時測定は測定効率の上昇につながる他、ある宇宙線のエネルギー損失を 2 つのセンサーで測定できるため、センサーの出力性能の評価がしやすくなる。図 3.4 と図 3.5 はそれぞれシリコンセンサー 1 つの場合の設置状況とその模式図、図 3.6 と図 3.7 はそれぞれシリコンセンサー 2 つの場合の設置状況とその模式図である。

地上で観測される μ 粒子のエネルギーは十分に大きいため、空気中やシンチレーター内、シリコンセンサー以外でのエネルギー損失は考えない。よって、シリコンセンサーの破損を避けるために、シリコンセンサー本体を、アルミニウム製の保護カバーで覆った状態で測定した。また、この保護カバーは、HDL 部分と冷却チューブ接続部分に隙間が空いており、室内灯や太陽光からの光子を遮断するのには不十分なため、上から暗箱と暗幕で厚く覆い測定を行った。

3.2.4 DAC 閾値設定

FPHX 読み出しチップは 1ch あたり 3bit の ADC と 8bit の DAC をもつ。各 ADC の閾値は対応する DAC により、任意に設定できる。DAC 閾値設定は表 3.2 のとおりである。DAC 設定値と対応電圧の関係は、

対応電圧=DAC 設定値× 4+210mV

という式で表せる。本実験ではできるだけ小さなノイズ信号をカットして測定を行うために、DAC0 に対応する電圧値を 270mV に決定した。また ADC0 と ADC7 以外は対応電圧の幅が一定になるように設定した。これはヒストグラム作成時にヒット数の傾向を見やすくするためである。

表 3.2　DAC 閾値設定

DAC	DAC 設定値	対応電圧 [mV]
DAC0	15	270mV
DAC1	23	300mV
DAC2	60	450mV
DAC3	98	600mV
DAC4	135	750mV
DAC5	173	900mV
DAC6	210	1050mV
DAC7	248	1200mV

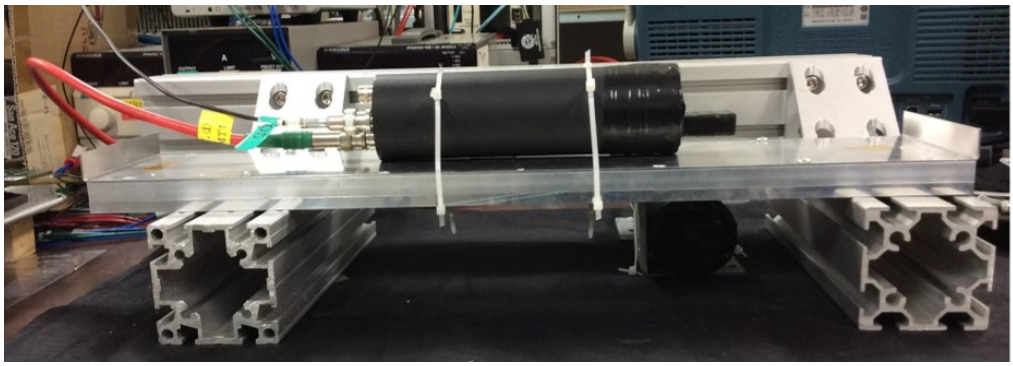

図 3.4　シンチレーションカウンター (A, B) とシリコンセンサー 1 つの設置状況

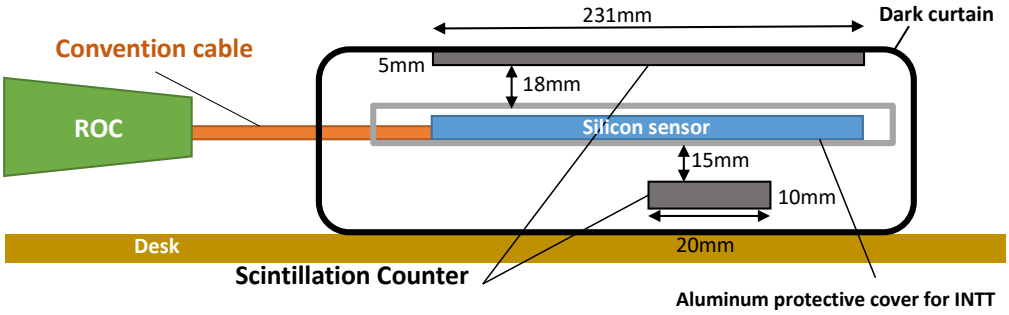

図 3.5　シンチレーションカウンター (A, B) とシリコンセンサー 1 つの設置状況の模式図

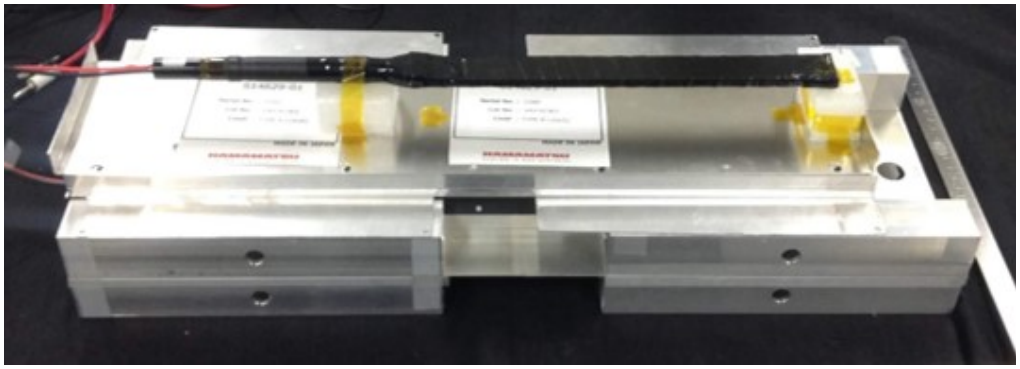

図 3.6　シンチレーションカウンター (A) とシリコンセンサー 2 つの設置状況

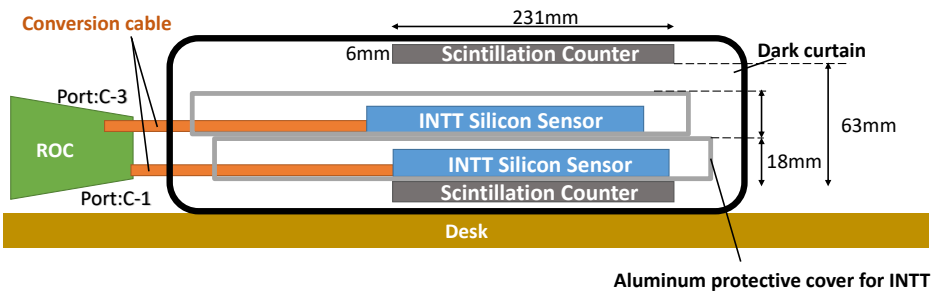

図 3.7　シンチレーションカウンター (A) とシリコンセンサー 2 つの設置状況の模式図

第 4 章

宇宙線データの解析

本測定の解析は、イベント選定、クラスター化、グラフの測定の 3 つで構成されている。シリコンセンサー 1 つで行った測定データのイベント選定ではノイズの除去のみ、シリコンセンサー 2 つで行った測定データのイベント選定ではノイズ除去と宇宙線イベントの選定の 2 つを行った。作成した解析プログラムは付録 readtree1, readtreeh, run_readtree, plot_readtree2 のとおりである。 これらの実行方法については付録 A で詳しく述べる。全体の流れは次のようである。

1. イベント選定：ノイズの除去
2. クラスター化解析
3. イベント選定：宇宙線イベントの選定（シリコンセンサー 2 つで行った測定データの場合）
4. グラフの測定

4.1 イベント選定：ノイズの除去

本解析では、2 つのノイズデータを除去した。1 つ目は存在しないモジュールから出力されるデータの除去である。本来はモジュールが接続されている ROC 上のポートからのデータのみが FEM に送られ PC に入力される。しかし実際のデータには、このような誤データも含まれている。これらのデータは解析したいモジュールが接続されているポートを選択することにより、その他のイベントを除去できる。2 つ目は、同じヒットが 2 イベント連続でくるイベントの除去である。このようなイベントは、片方もしくは両方のモジュールからの出力に含まれていた。

4.2 クラスター化解析

4.2.1 クラスター化解析の必要性

宇宙線がシリコンセンサー表面に対して垂直方向に入射した場合、1 チップの 1 ストリップのみ通過することが多い。一方、宇宙線が斜め方向に入射した場合、複数のストリップを通過することがある。この

時、1 粒子のイベントが複数ストリップに分かれて記録されている。ここで隣り合ったヒットを 1 つにまとめ（クラスター化)、1 粒子のイベントを得ることで、宇宙線のシリコン内でのエネルギー損失の状態とシリコンセンサーの動作確認を行うことができる。複数ストリップを通過した宇宙線イベントのクラスター化イメージ図は図 4.1 の通りである。

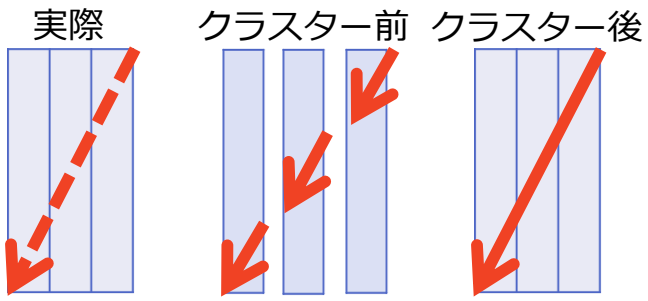

図 4.1　複数ストリップを通過した宇宙線イベントのクラスター化イメージ図

4.2.2　クラスター化解析方法

本実験で得られるデータはブランチ構造になっており、同じ beam CLK で記録された entry を、1 event ずつ区別し、1 event ごとの連続ストリップ通過数を得た。クラスター化解析の流れを次に示す。

クラスター化解析の流れ

1. 不要データ除去後のイベントを配列に入れる。
2. チップ番号ごとにチャンネル番号で配列のデータを並び替える。
3. 1event に含まれるクラスター数を測定する。
4. クラスター数が 1 以上のチップ番号のデータをクラスター化する。
   1) 1 チップのヒット数がクラスターに使われるヒット数（ストリップ数）に等しい時→ Nhits=(1 チップののヒット数) とし、クラスター化する。
   2) チャンネル番号が連続の時→ヒット数 (ストリップ数) 増やす。
   3) チャンネル番号が連続でない時→ヒット数 (ストリップ数) (Nhits) 分をクラスター化する。

1 event ごとに読み出されたヒットデータから不要イベントを除去した後、チップごとのヒットチャンネルが連続しているかを確認する。連続したヒットチャンネルは宇宙線 1 event 分のヒットであると考えられるので、クラスター化する。ここでは、連続ヒット数とその損失エネルギー和、ヒットチップ番号、平均チャンネル番号を得る。連続ヒットがない場合は、1 ヒットを 1 クラスターとする。全チップに対しクラスター化を行った後、1 event あたりの全クラスター数を測定する。

全 event に対しこのクラスター化と測定を行った後、損失エネルギー分布とクラスター数分布、連続ヒット数分布などを測定する。

4.3　イベント選定：宇宙線イベントの選定

シリコンセンサー 2 つを利用した測定では、イベント選定として宇宙線イベントの選定を行った。その条件は次の 3 つである。

1) 宇宙線が 2 つのモジュールを同時に通過するイベント
2) 垂直に位置しているシリコンセンサーとストリップにヒットがあるイベント
3) 1 イベントあたり 1 クラスターのみのイベント

イベント選定 1) と 2) より、2 つのモジュールに同じ μ 粒子が通過したイベントのみを解析することができる。2) のシリコンセンサーとストリップの選択は、各センサーに対応する FPHX チップの番号とチャンネルの番号の組み合わせで決定した。

チップの組み合わせでは 2 つのモジュールのヒットチップ番号の差が 2 または 3 のイベントを選定した。図 4.2 はチップ選定に用いる分布である。図 4.2 の左は上下のモジュールのヒットチップ番号の相関分布である。横軸と縦軸がそれぞれ上層と下層のモジュールのチップ番号を、色がヒット数を表す。黄色の部分のチップ番号の組が多いことがわかる。図 4.2 の右は図 4.2 左のチップ番号の差のヒストグラムである。この分布より、1σ 以内のチップ差は 2 または 3 であることがわかる。

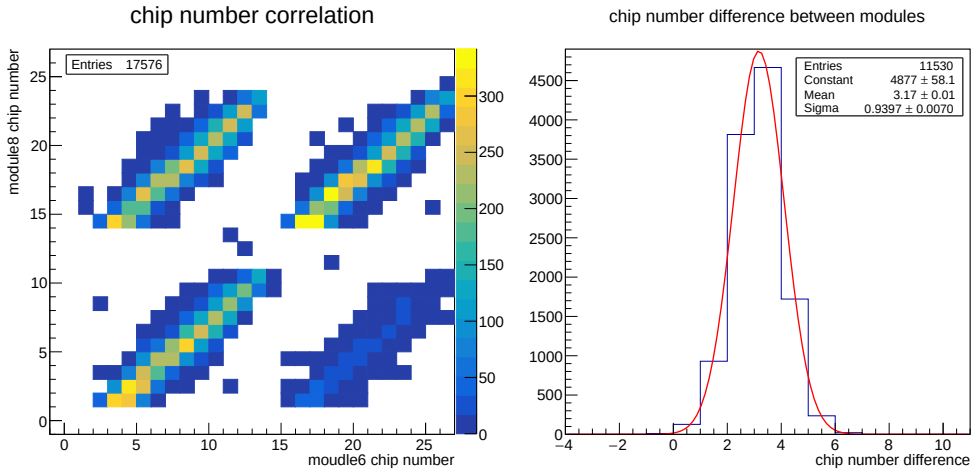

図 4.2　（左図）上下の INTT モジュールのヒットチップ番号の相関分布,　横軸：上層のモジュールのチップ番号,　縦軸：下層のモジュールのチップ番号,　（右図）上下の INTT モジュールのヒットチップ番号の差の分布

また、チャンネルの組み合わせでは 2 つのモジュールのヒットチャンネル番号の差が-19 から 27 のイ

ベントを選定した。どちらの選定においてもガウス分布の 1 シグマいないのデータを選定した。図 4.3 はチップ選定後のチャンネル選定に用いる分布である。図 4.3 の左は上下の INTT モジュールにヒットした選定チップのチャンネル番号の相関分布である。横軸と縦軸がそれぞれ上層と下層のモジュールのチャンネル番号を、色がヒット数を表す。ヒットチャンネルを見やすくするため、軸を 8 等分したビンに設定した。黄色の部分のチャンネル番号の組が多いことがわかる。図 4.3 の右は図 4.3 の左のチェンネル番号の組み合わせの差のヒストグラムである。この分布より、1σ 以内のチャンネル差は-21〜27 あることがわかる。

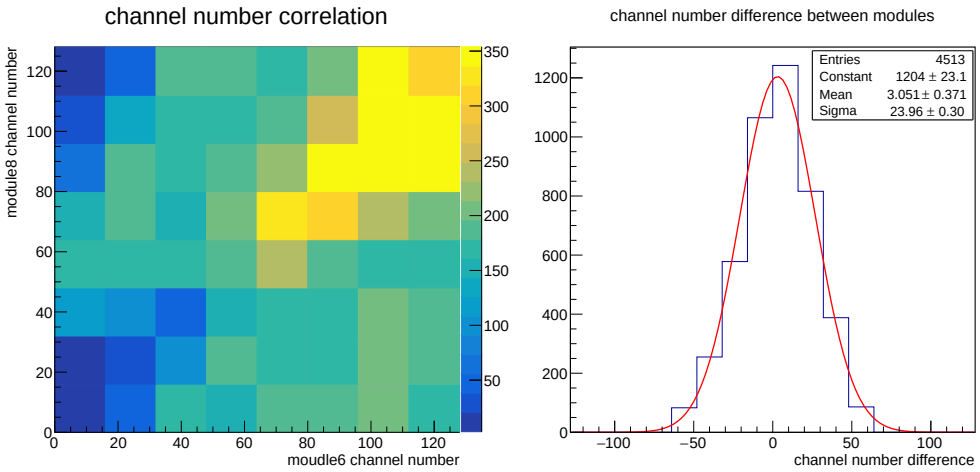

図 4.3　（左図）上下の INTT モジュールのチップ選定後のヒットチャンネル番号の相関分布，　横軸：上層のモジュールのチャンネル番号，　縦軸：下層のモジュールのチャンネル番号，　（右図）上下の INTT モジュールのチップ選定後のヒットチャンネル番号の差の分布

イベント選定 3) より、不要ノイズによるデータを除き、宇宙線とみられるイベントのみを解析することができる。この解析ではクラスター化を行った。クラスター化については 5.2 節で詳しく述べる。

第 5 章

宇宙線データの解析結果

5.1　測定データ

測定したデータの種類と各データのイベント数、測定時間は次のとおりである。

表 5.1　測定したデータの種類、各データのイベント数、測定時間

シリコンセンサー個数 [個]	シンチレーターの種類	測定時間 [時間]	全データ数 [個]
1	A と B	208.5	35671
2	A のみ	162.5	218606

5.2　イベント選定

図 5.1 はクラスター化前のエネルギー損失分布のイベント選定による変化を表す。左図はノイズイベント除去前、中央図はノイズイベント除去後の損失エネルギー分布、右図はノイズ除去と宇宙線イベント選定後のエネルギー損失分布を電圧表示したものである。

5.2.1　ノイズイベントの除去

ノイズイベントの除去により、図 5.1 の左図から中央図へ損失エネルギーが変化した。その結果 270〜300mV に存在する大量のヒットが減り、270〜300mV と 450〜600mV にピークが確認できた。

ノイズ除去後のチップに読み込まれたヒットチャンネル分布は図 5.2 と 5.3 と 5.4 である。図 5.2 は 1 チップあたりチャンネル分布で、縦軸がチャンネル番号、横軸がチャンネルのヒット数を表す。図 5.3 と 5.4 は図 5.2 の各チップの分布を 26 個集め、1 モジュール分のチャンネル分布を表している。図 5.3 は上層のシリコンセンサー A、 5.4 は下層のシリコンセンサー B である。これらの分布より、宇宙線がシリコンセンサーにヒットし、各チップのチャンネルに読み込まれていることが確認できる。また、2 つ

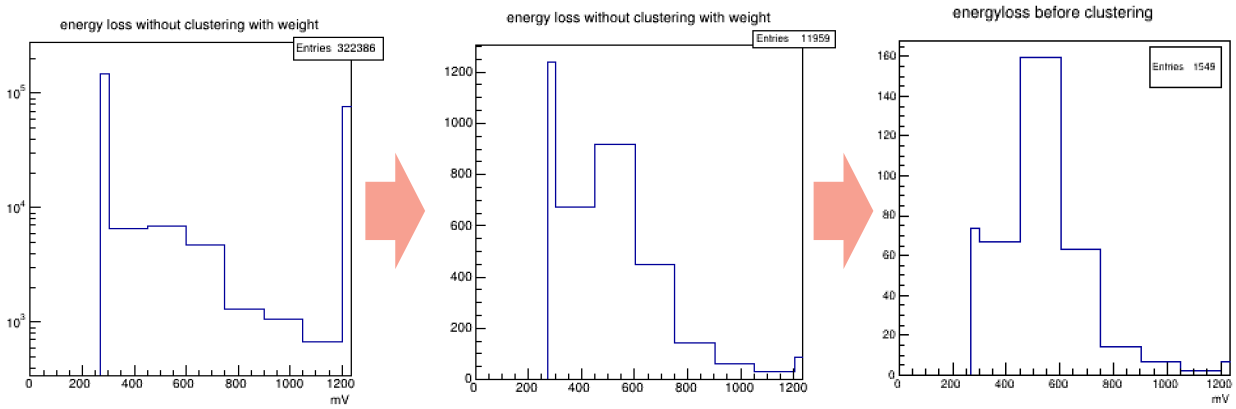

図 5.1　クラスター化前のエネルギー損失分布 (電圧表示),　（左図）ノイズ除去前,　（中央図）ノイズ除去後,（右図）ノイズ除去と宇宙線イベント選定後

のシリコンセンサーは ROC のポートとコンバージョンケーブルの接続の関係により、真上ではなくずらして設置してある。図 5.3 と図 5.4 からわかるように、上層のシリコンセンサー A は右側、下層のシリコンセンサー B は左側に多くのヒットがある。これより、両シリコンセンサーが重なっている部分に多くのヒットがあることが確認できる。

5.2.2　宇宙線イベントの選定

宇宙線イベント選定は、トリガー用シンチレーションカウンター 2 つでシリコンセンサー 2 つを挟んで設置し、測定したデータにのみ行った。宇宙線イベント選定により、図 5.1 の中央図から右図へ損失エネルギーが変化した。その結果、ヒストグラムの確認できるピークが 1 つになり、270〜300mV のピークはノイズ、450〜600mV のピークは宇宙線損失エネルギーであることが確認できた。これより、2 つのシリコンセンサーで測定を行うことはシリコン中での宇宙線損失エネルギー分布を確認する方法として有効であるといえる。

5.3　クラスター化

クラスター化は、 1 CLK あたりに含まれる連続チャンネルのヒットを足し合わせ、その足し合わせたヒット数別に損失エネルギー分布を測定した。

図 5.5 の左はノイズ除去後の 1 CLK あたりに含まれるヒット数分布、図 5.5 の右はノイズ除去後の 1 CLK あたりに含まれるクラスター数分布を表す。図 5.5 の左より、1 CLK あたり約 7 割が 1 ヒット、約 3 割が 2 ヒットであることがわかる。従って、測定した宇宙線はほぼ垂直にシリコンセンサーに入射し、1 〜 2 個のストリップを通過したといえる。図 5.5 の右より 1 CLK あたりのクラスター数はほとんどが

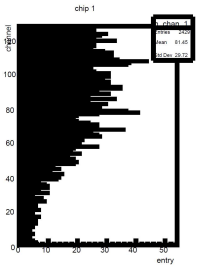

図 5.2　シリコンセンサーの 1 チップあたりチャンネル分布（ノイズ除去後）

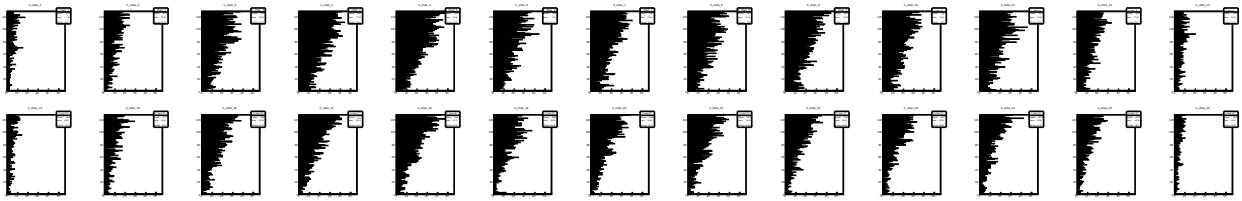

図 5.3　上層のシリコンセンサー A のチップごとのチャンネル分布（ノイズ除去後）

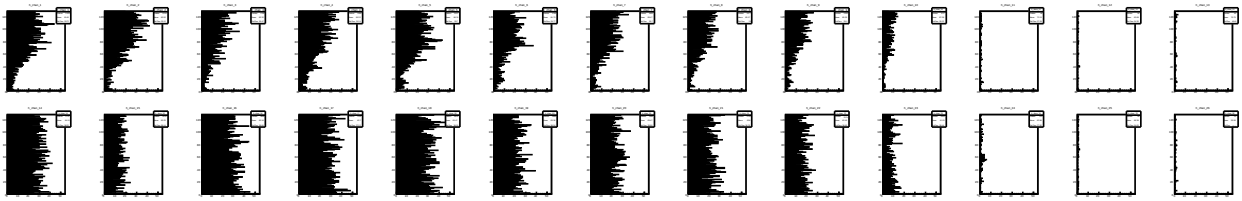

図 5.4　下層のシリコンセンサー B のチップごとのチャンネル分布（ノイズ除去後）

1 であることがわかる。これらの分布は、ノイズ除去後と宇宙線イベント選定後で結果が変化しなかった。従って、測定したイベントのデータは不要ノイズによるものではなく、宇宙線のイベントであるといえる。

図 5.6 はイベント選定（ノイズ除去、宇宙線イベント選定）された測定データのクラスター化後のヒット数別の損失エネルギー分布であり、赤が 1 ヒット、青が 2 ヒットのエネルギー損失を表す。これより、宇宙線のシリコン内での損失エネルギーは 1 ヒットでは 450～600 m V、2 ヒットでは 750～900mV にピークがあることがわかる。従って、ヒット数により損失エネルギーが異なることがわかる。

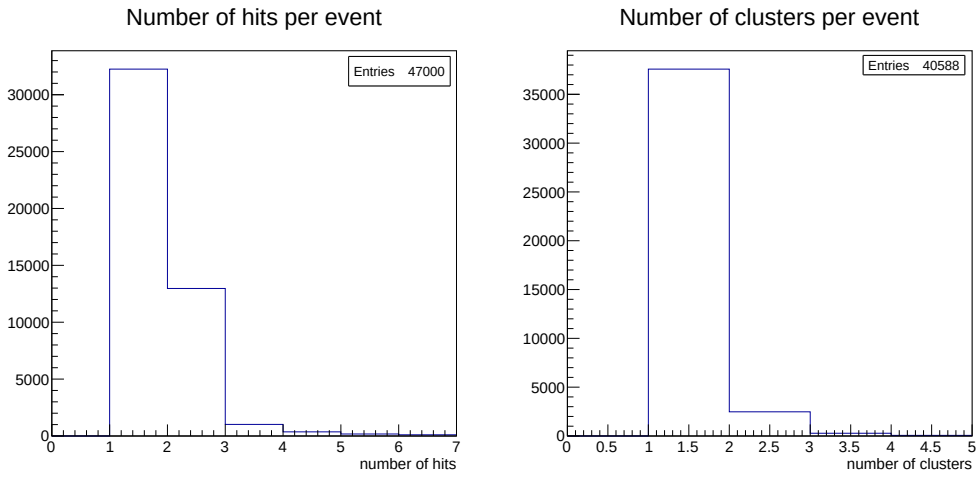

図 5.5　（左図）ノイズ除去後の 1 CLK あたりに含まれるヒット数分布,　（右図）ノイズ除去後の 1 CLK あたりに含まれるクラスター数分布

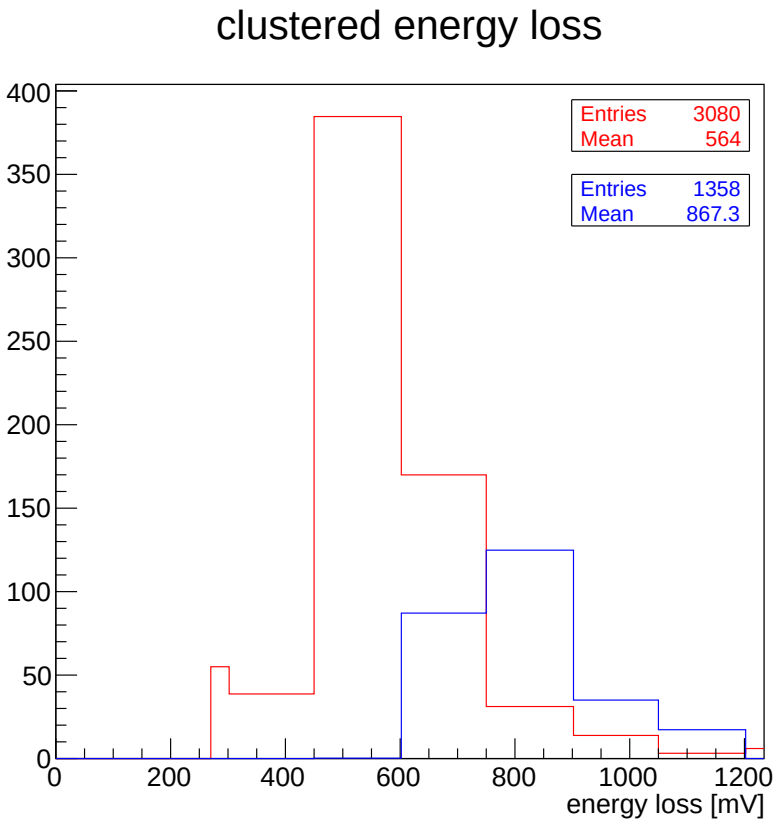

図 5.6　クラスター化後のエネルギー損失分布 (電圧表示)

第 6 章

測定で明らかになった問題点と改善

6.1 問題点

宇宙線測定の結果から 2 つの問題点が上がった。

1. 予想エネルギー損失と測定結果の違い
2. 1 ヒットと 2 ヒットのエネルギー損失ピーク位置の違い

6.1.1 項と 6.1.2 項でこれらの問題について説明する。

6.1.1 予想エネルギー損失と測定結果の違い

宇宙線がシリコン内で損失する予想エネルギー値を計算し測定結果と比較する。1 GeV の μ 粒子が 320μm 厚のシリコンを通過した時の MIP(Minimum Ionizing Particle) のエネルギー損失を基準に計算した。まず、320μm におけるシリコンのエネルギー損失は 2.1.3 節より $1.15\mathrm{MeV}\cdot\mathrm{g}^{-1}\cdot\mathrm{cm}^2$ である。また、シリコンの密度は $2.33\mathrm{g}\cdot\mathrm{cm}^{-3}$ であるので、損失エネルギーは式 6.1 のように計算できる。

$$1.15\mathrm{MeV}\cdot\mathrm{g}^{-1}\cdot\mathrm{cm}^2\cdot 2.33\mathrm{g}\cdot\mathrm{cm}^{-3}\cdot 0.032\mathrm{cm}\cdot 10^6 \simeq 85700\mathrm{eV} \tag{6.1}$$

本実験で読み出されたデータは、式のように損失エネルギーを電荷に変換し、ゲイン値をかけて電圧表示することで評価する。式 6.3 におけるゲイン値は、表の改善前の GUI 上での入力値 GSel=2 とした時の値 $60\times 5\mathrm{mV/fC}$ を利用している。

$$\frac{85700\mathrm{eV}\times 1.60\times 10^{-19}\times 10^{15}}{3.67\mathrm{eV}} \simeq 3.79\mathrm{fC} \tag{6.2}$$

$$3.79\mathrm{fC}\times 60\mathrm{mV/fC}\times 5 \simeq 1100\mathrm{mV} \tag{6.3}$$

従って、1 GeV の μ 粒子が 320μm 厚のシリコンを通過した時に損失するエネルギー損失は 85700eV、電圧表示すると 1100mV である。一方、本実験で行った測定において、損失エネルギーのピーク位置は 1 ヒットで 450〜600mV、2 ヒットで 650〜900mV である。従って予想損失エネルギー値と測定値が異な

ることが判明した。

6.1.2　1 ヒットと 2 ヒットのエネルギー損失ピーク位置の違い

ヒット数ごとに予想損失エネルギーを計算する。その結果、1 ヒットと 2 ヒットの予想値の差は約 100mV となる。一方、測定結果では 1 ヒットと 2 ヒットのピーク位置の差は約 200mV であるとこから、予想値よりも 1 ヒットと 2 ヒットの損失エネルギーピーク差が大きいことが判明した。

6.1.3　問題点の対応

6.1.1 項と 6.1.2 項の問題の原因として、

1. 予想損失エネルギー計算に利用するゲイン値が異なる。
2. 測定データにオフセットが関与している。

の 2 つが考えられた。 6.2 節と 6.3 節で 2 点の原因を検討する。

6.2　ゲイン値

6.1 節の問題点の原因として、予想損失エネルギー計算に利用するゲイン値が異なるということが考えられた。測定で出力される損失エネルギー値に使われるゲイン値は、測定前にユーザーが GUI の画面上に入力する値 (GSel) と対応関係にある。今回は以前まで使われていた対応関係を見直し、それぞれの入力値で実験を行った。

表 6.1　ゲイン値と GSel 入力値の対応関係

GSel		0	1	2	3	4	5	6	7
ゲイン	以前	46 × 5	50 × 5	60 × 5	67 × 5	85 × 5	100 × 5	150 × 5	200 × 5
[eV/fC]	今回	200	150	100	85	67	60	50	46

6.2.1　宇宙線測定によるゲイン値の確認

GSel 入力値を 0〜7 まで変えて、測定した損失エネルギーは図 6.2 である。全て 1 ヒットの損失エネルギーを電圧表示している。

図 6.2 より、GSel 入力値が大きくなるにつれ、エネルギー損失のピーク位置が低いほうシフトしていることが確認できる。つまり、GSel 入力値を大きくするとゲイン値が小さくなり、出力される損失エネルギーが小さくなるということである。従って、今回新たに考えた GSel 入力値とゲイン値の対応関係は正しかったといえる。

6.2.2　キャリブレーションテストによるゲイン値の確認

GSel 入力値を 0〜7 まで変えてキャリブレーションテストを行った際のゲイン値による出力結果の変化が測定されている。詳しくは森田 [7] を参照されたい。各 GSel での測定で比較したのは図 2.12 の右の分布で、図 6.1 の通りである。それぞれが各 GSel 入力値において、1 チップでの入力 amplitude と出力 ADC の相関関係である。上段は左が GSel=0 で一番左が GSel=3、下段は左が GSel=で一番右は GSel=7 である。これより、GSel 入力値を大きくなるにつれ入力 amplitude に対する出力 ADC の値が小さくなることがわかる。つまり、GSel 入力値を大きくするとゲイン値が小さくなり、出力される損失エネルギーが小さくなるということである。従って、今回新たに考えた GSel 入力値とゲイン値の対応関係は、キャリブレーションテストにおいても正しかったといえる。

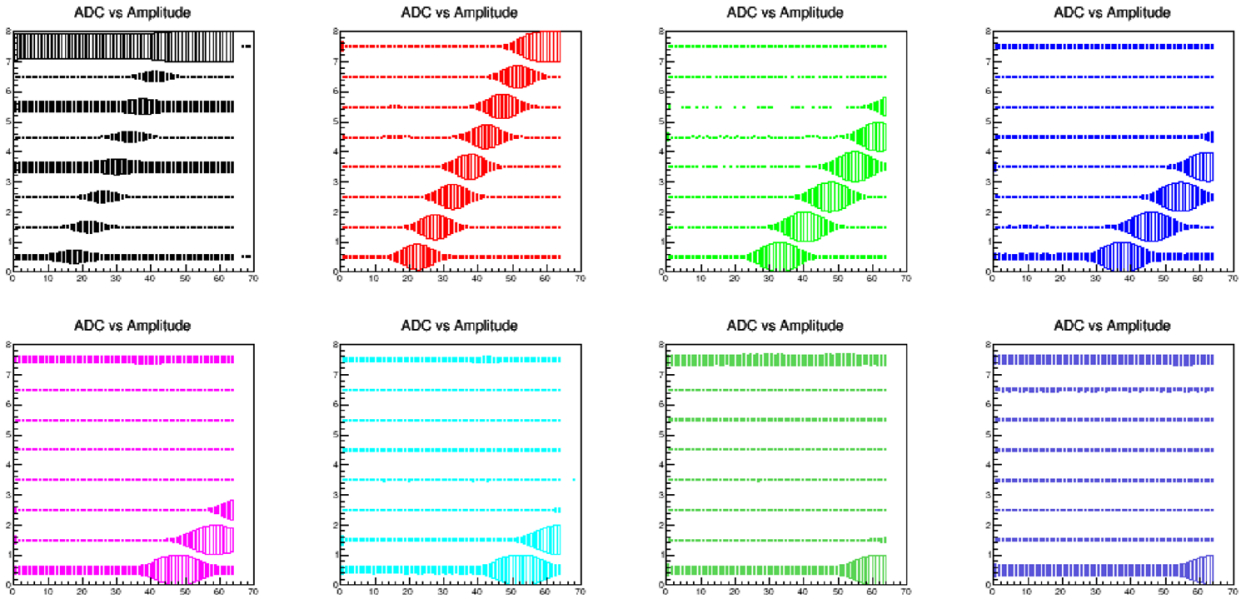

図 6.1　各 GSel 入力値でのテストパルスの入力波高（amplitude）と出力 ADC の相関分布, 横軸：入力波高（amplitude）, 縦軸：出力 ADC

6.2.3　適切なゲイン値の決定

適切な GSel 入力値とゲイン値の対応関係（表 6.1）を利用すると、今回の測定での GSel 入力値=2 に対応するゲイン値は 100eV/fC である。これより、1 GeV の宇宙線がシリコン 320μm を通過した際の予想損失エネルギーは約 370mV となる。

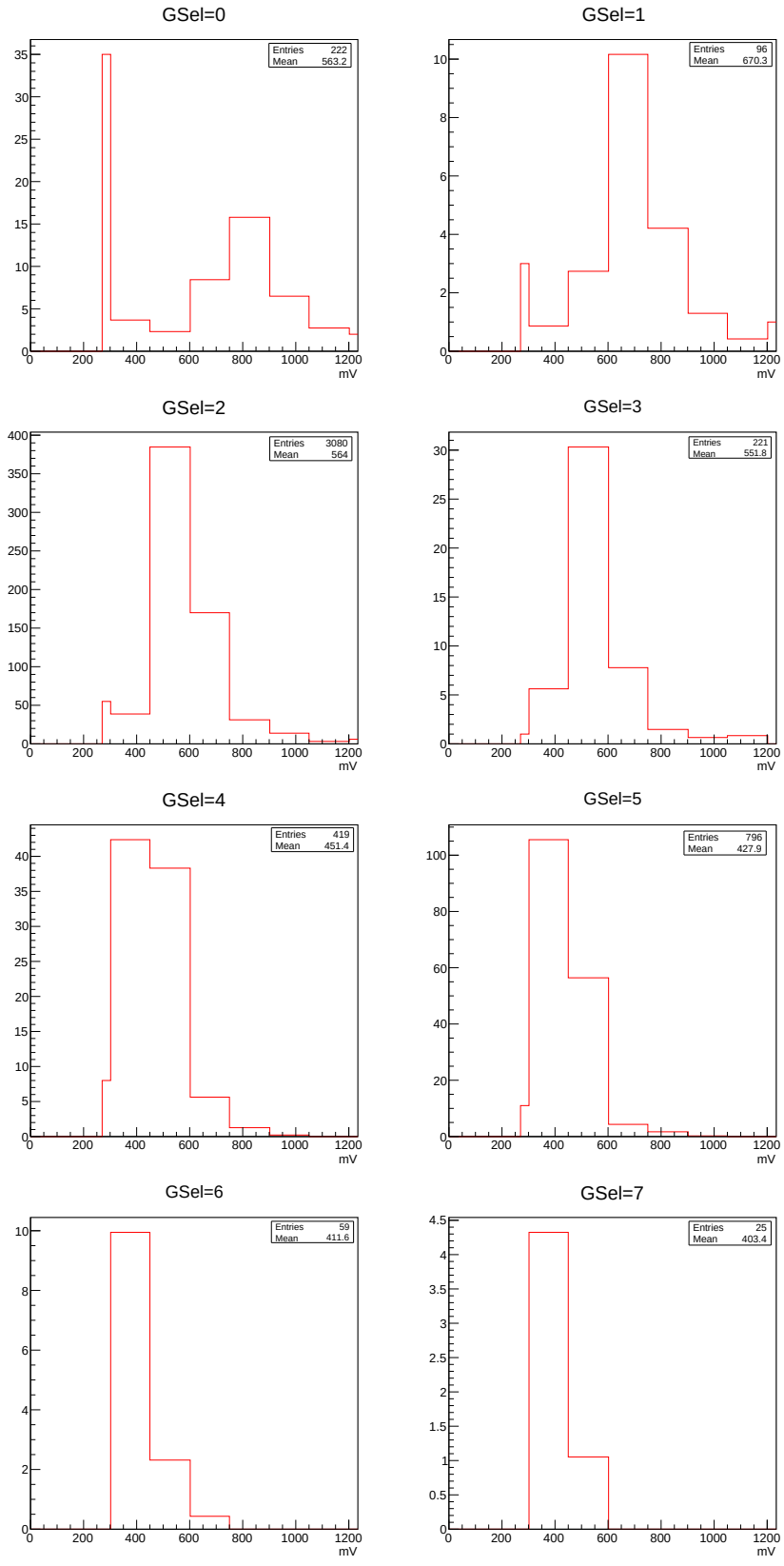

図 6.2　GSel の入力値を変えて測定をした 1 ヒットのエネルギー損失 (電圧表示)

6.3　オフセット

6.2.3 節で決定したゲイン値を用いても、 6.1 節で述べた 2 つの問題点は解決しなかった。そこで、ゲイン値とは別の値、オフセットが関与しているのではないかと考えた。ここでいうオフセットとは測定基準点の 0 点からのズレの値を表す。つまり、全く関係のない値 (オフセット) がもともと測定値に足されているということだ。また、今回の測定に関与したオフセットは 1 チップあたりの読み出しごとに存在すると考えた。図 6.3 にヒットに関与するオフセットの模式図を示す。μ 粒子（赤と青の矢印）がシリコンセンサーを通過する際、各チャンネルに対し閾値以上の損失エネルギーの読み出しにオフセットが関与している。つまり、赤の μ 粒子に関与するオフセットは 1 つ、青の μ 粒子に関与するオフセットは 2 つである。

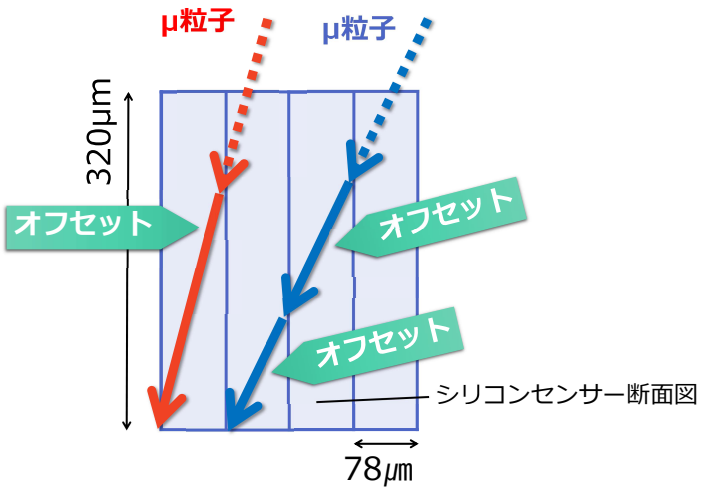

図 6.3　ヒットに関与するオフセットの模式図

6.3.1　シミュレーションによるオフセットの検討

上で述べたように、オフセットが 1 チップの読み出しに対し 1 つ存在するようなシミュレーションプログラムを作成し、測定結果と比較することでオフセットの存在の妥当性について議論した。オフセットが関与するのは読み出し回路中でエネルギーの単位が eV から電圧 mV に変換される所なので、このエネルギー損失の変換過程を詳しく測定した。シミュレーションプログラムの内容については付録 B で述べる。

読み出し回路中での損失エネルギーの変換

読み出し回路中での損失エネルギーの変換を、宇宙線 μ 粒子がシリコンセンサーで損失したトータルエネルギーを電圧表示したものであらわす。

1 ゲイン (増幅係数) を用い、電荷 [C] を電圧 [mV] に変換する。
2 電圧値 [mV] にオフセット [mV] を足す。
3 損失エネルギーを ADC で 8 つのビンに分ける。

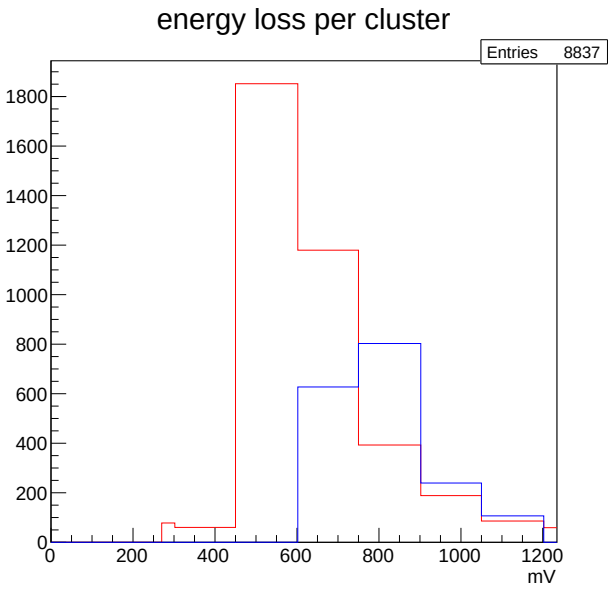

図 6.4　シミュレーションで生成した宇宙線イベントのクラスター化後のエネルギー損失分布 (電圧表示)

図 6.4 はシミュレーションで生成した宇宙線イベントのクラスター化後のエネルギー損失分布であり、赤が 1 ヒット、青が 2 ヒットのエネルギー損失を表す。この分布を測定した際のゲイン値は 88、オフセットは 198mV である。これより、シミュレーションで生成した宇宙線においてシリコン内での損失エネルギーは 1 ヒットでは 450〜600 m V、2 ヒットでは 750〜900mV にピークがあることがわかる。これは実験データを解析して測定したエネルギー損失の分布がしめす結果に等しい。従って、本シミュレーションにより、宇宙線測定を正しく再現できていることがわかり、シミュレーションイベントにおいてもヒット数により損失エネルギーが異なることがわかる。一番左の分布はゲイン値が関与し、エネルギー損失が電圧表示されたものである。ここで変換された値はゲイン値により異なる。本シミュレーションのゲイン値は、90mV/fC である。この分布から分かるように、ゲイン値による増幅後での 1 ヒットと 2 ヒットのエネルギー損失のピークはほぼ同じである。左から 2 番目は、電圧表示された値にオフセットが関与した後の分布である。この段階で、ヒット数ごとのエネルギー損失ピークに差が生じる。これは FPHX チッ

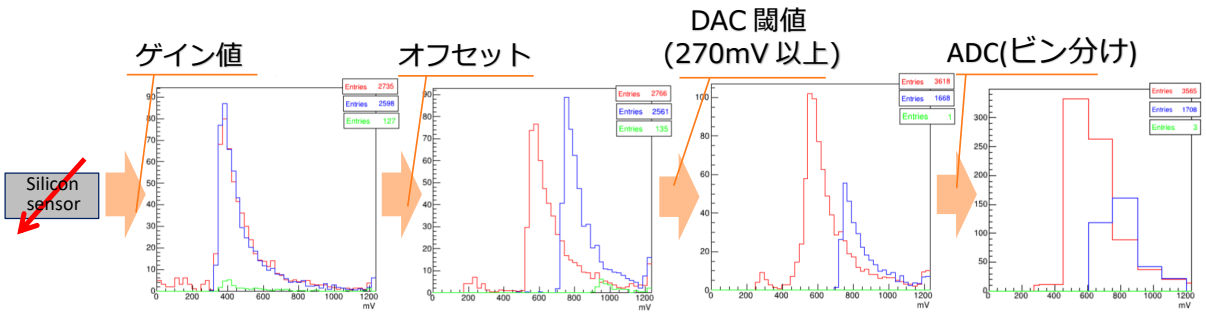

図 6.5　読み出し回路中での損失エネルギーの変換の流れ, 宇宙線 μ 粒子がシリコンセンサーで損失した全エネルギーの電圧表示, 横軸：電圧 [mV]

プのチャンネルごとにオフセットが存在することにより、全損失エネルギー内にクラスターにまとめられたヒットのチャンネル数分のオフセットが足されているためである。従って、2 ヒットによるクラスターの場合、全エネルギー損失に 2 チャンネル分のオフセットが含まれている。左から 3 番目は、オフセットが足された値を DAC 閾値判定した後の分布である。DAC 閾値設より閾値以下のエネルギー損失が読み出されないので 1 ヒットと 2 ヒットのエントリー数割合が変化する。これは、2 ヒットと判断されていたイベントのうち片方のストリップの読み出しエネルギーが閾値以下だった場合、1 ヒットと判断されるからである。この後 ADC でエネルギーがビン分けされることにより、宇宙線測定で得られた ADC 分布をシミュレーションでも測定することができる。

ここでオフセットが関与する前後のヒストグラムを比較する。はじめは 1 ヒットと 2 ヒットの損失エネルギーのピークは同じところに位置しているが、オフセットが足されることにより 1 ヒットと 2 ヒットの損失エネルギー 1 チップの読み出しに対し 1 つのオフセットが関与することで宇宙線測定の結果を再現できた。従って、オフセットの関与が予想エネルギー損失と測定結果の違いと 1 ヒットと 2 ヒットのエネルギー損失ピーク位置の違いを生み出していることがわかった。

6.3.2　キャリブレーションテストによるオフセットの検討

キャリブレーションテストでもオフセットの存在を確認することができる。取得データの解析の結果、GSel 入力値に関係なくオフセットの値は約 198mV であることがわかった。詳しくは森田 [7] を参照されたい。

6.3.3　宇宙線測定によるオフセットの検討

宇宙線測定においてオフセットの存在を確認するために、より厳しい条件でイベント選定を行い損失エネルギー測定した。第 6 章で述べた問題点である 1 ヒットと 2 ヒットのエネルギー損失ピーク位置の違い

がオフセットによるものであることを確かめるためには、ヒット数以外の条件が等しいイベント同士を比較しで議論する必要がある。そのため、同じ宇宙線 (同じ運動量、入射角) で 2 つのシリコンセンサーに異なるヒット数を読み出しがあるイベントを選定した。図 6.6 は同一宇宙線が 2 つのシリコンセンサーで異なるヒット数の損失エネルギーを記録するイベントのイメージ図である。

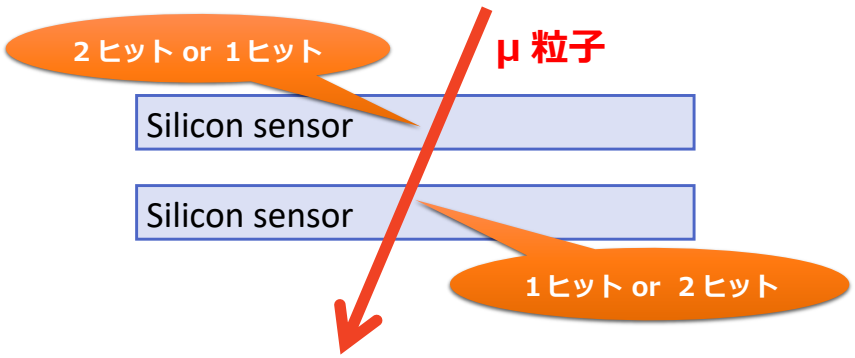

図 6.6　同一宇宙線が 2 つのシリコンセンサーで異なるヒット数の損失エネルギーを記録するイベントのイメージ図

同一宇宙線イベント選定条件

ここでのイベント選定は、第 5 章で述べたノイズ除去と宇宙線イベント選定を行った後に行う。追加イベント選定条件は次の 2 つである。

1. 1 CLK あたりに 2 つのシリコンセンサーでヒットがあるイベント
2. 1 CLK あたりに 2 つのシリコンセンサーで異なるヒット数の損失エネルギーを記録するイベント

同一宇宙線イベント選定結果

図 6.7 は同一宇宙線が 2 つのシリコンセンサーで異なるヒット数を記録するイベントの損失エネルギー相関分布で、横軸と縦軸がそれぞれ 1 ヒットと 2 ヒットのエネルギー損失、色がヒット数を表す。この図より、同一宇宙線で 2 つのシリコンセンサーに異なるヒット数を記録したイベントのそれぞれのエネルギー損失において、1 ヒットでは 450〜600mV、2 ヒットでは 750〜900mV という組み合わせが最も多いことがわかる。図 6.8 は同一宇宙線選定後の、同一宇宙線で 2 つのシリコンセンサーに異なるヒット数を記録したイベントのそれぞれのエネルギー損失は 1 ヒットでは 450〜600mV、2 ヒットでは 750〜900mV という組み合わせのイベントが最も多いことがわかる。従って、入射粒子の運動量や入射角度は同じであっても読み出された際のヒット数のみによって測定されるエネルギー損失が異なるということがわかった。

この結果はシミュレーションでオフセットの関与を踏まえた際の損失エネルギーの測定結果に等しい。

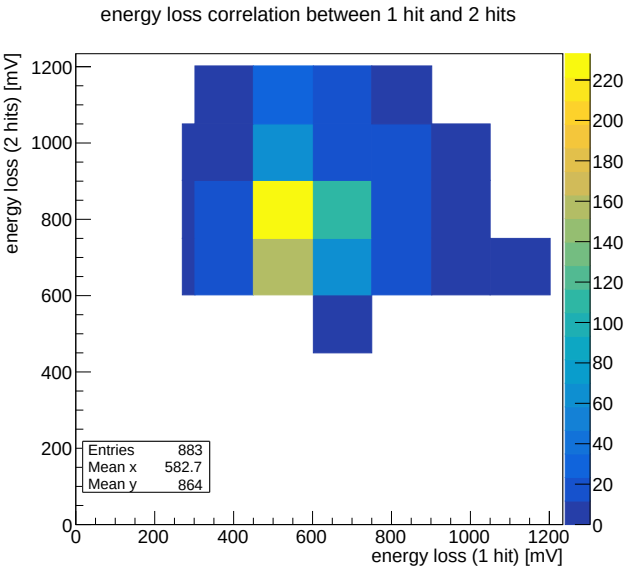

図 6.7　同一宇宙線が 2 つのシリコンセンサーで異なるヒット数を記録するイベントの損失エネルギー相関分布, 横軸：1 ヒットのエネルギー損失, 縦軸：2 ヒットのエネルギー損失

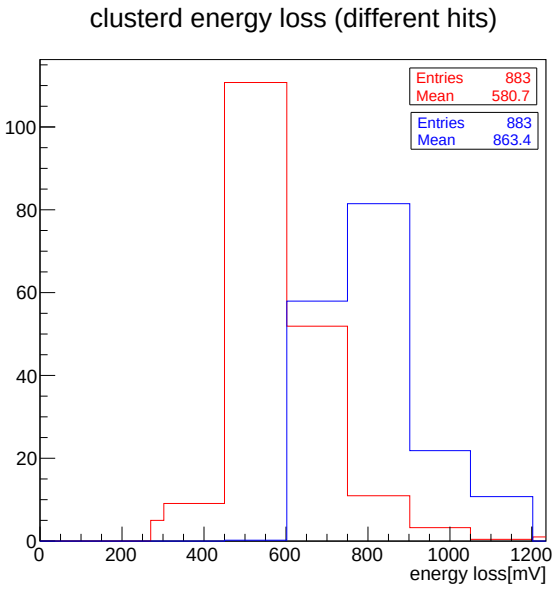

図 6.8　同一宇宙線が 2 つのシリコンセンサーで異なるヒット数を記録するイベントの損失エネルギー分布, 赤：1 ヒットのエネルギー損失、青：2 ヒットのエネルギー損失

すなわち、オフセットの存在は妥当であるといえる。

6.3.4　適切なオフセットの決定

宇宙線測定、シミュレーション、キャリブレーションテストにおいて、オフセットの存在が尤もらしいと考えた。よって、シミュレーションとキャリブレーションテストの結果を用いると尤もらしいオフセットは 198mV と考えられる。今回の測定での GSel 入力値に対応するゲイン値 100eV/fC とオフセット 198mV を用いると、1 GeV の宇宙線がシリコン 320μm を通過した際の予想損失エネルギーは約 568mV となる。

6.4　問題の改善

宇宙線測定、キャリブレーションテスト、シミュレーションの結果より、尤もらしいゲイン値とオフセットの存在が明らかになった。今回の測定から推測すると、ゲイン値はおおよそ 85、オフセットはおおよそ 198mV であると考えられる。これらの値を利用したシミュレーション結果は図 6.9 の通りである。上段が宇宙線測定、下段がシミュレーションの結果を表している。Cosmic1 と simulation1 はヒット数ごとに色分けしたクラスター化前のエネルギー損失分布を電圧表示したもの、Cosmic2 と simulation2 はヒット数ごとに色分けしたクラスター化後のエネルギー損失分布を電圧表示したもの、Cosmic3 と simulation3 は 2 ヒットを構成する 2 つのエネルギー損失がもつ ADC 値の相関分布である。これらの比較から、エネルギー損失ピークのシミュレーションと宇宙線測定の結果が一致していること、2 ヒットを構成する ADC 値の相関分布が似ていることから、シミュレーションに用いたゲイン値とオフセットは妥当であるといえる。

表 6.2　尤もらしいゲイン値とオフセット

GSel(入力値)	ゲイン値	オフセット
2	85 eV/fC	198mV

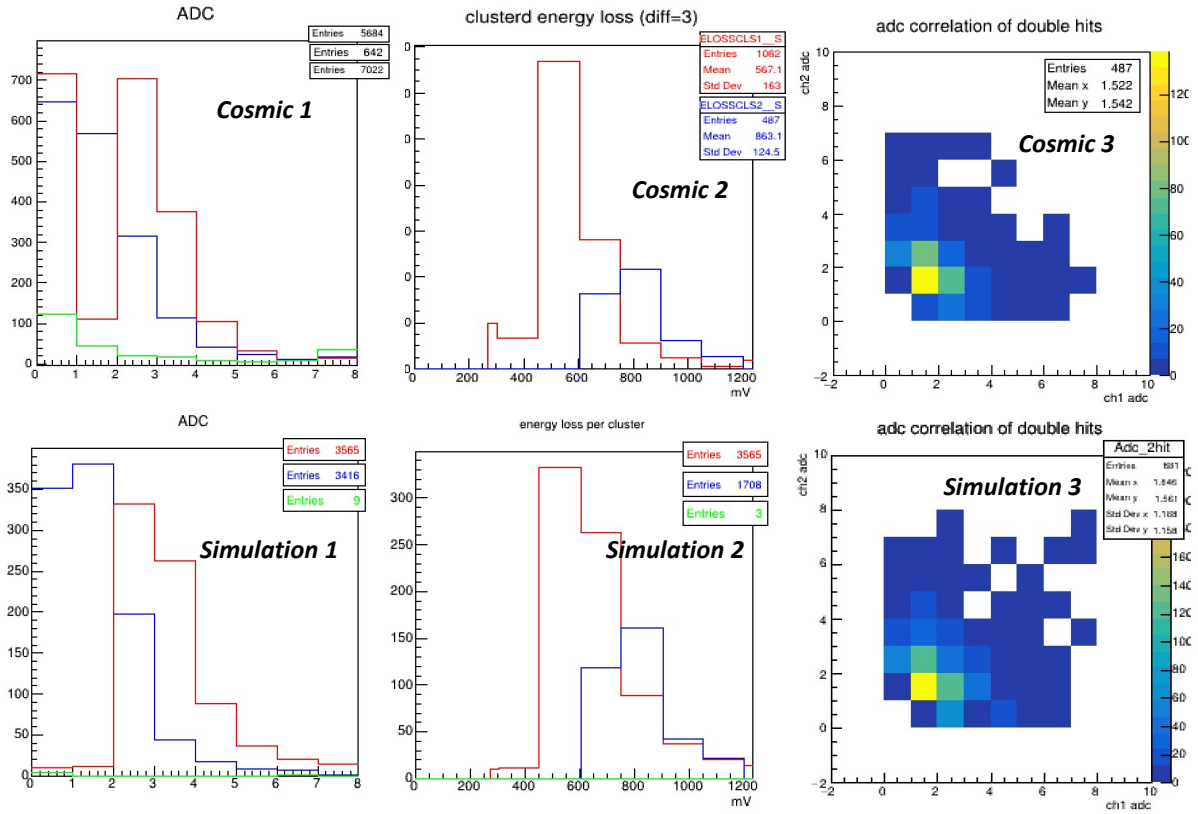

図 6.9　宇宙線測定とシミュレーションの比較 (上段) 宇宙線測定, (下段) シミュレーション　cosmic 1 , simulation1：ヒット数ごとに色分けしたクラスター化前のエネルギー損失分布 (電圧表示), cosmic2, simulation2：ヒット数ごとに色分けしたクラスター化後のエネルギー損失分布 (電圧表示), cosmic3, simulation3：2 ヒットを構成する 2 つのエネルギー損失がもつ ADC 値の相関分布

第 7 章

結論

今回、奈良女子大学における INTT のテストベンチを利用し、キャリブレーションテストと宇宙線測定によりデータ読み出しシステムの動作確認やデータ解析による評価を行った。測定においては 2 つのシリコンセンサーの同時使用に成功し、多くのデータを測定した。データ解析においては、宇宙線ヒットのクラスター化解析プログラムとそのシミュレーションプログラムを作成した。これまで MIP ピークの測定値と期待値がずれる問題があったが、これらの利用により①今まで利用していた予想エネルギー損失の計算式に用いる入力値とゲイン設定値の対応関係に誤りがあること、②チップから読み出されたデータにオフセットがあることが原因であるとわかった。これら 2 点に対して、宇宙線測定、シミュレーション、キャリブレーションテストの 3 つの方法で原因の妥当性と尤もらしい値の検討を行った。その結果、入力値 2 に対応するゲイン値は 85eV/fC（対応表においては約 100eV/fC)、オフセットは 198mV がもっともらしい値であることがわかった。

さらに今回の研究を行い、データの読み出しシステム、解析までの動作の一連の流れと、DAQ、ハードウェアの理解、シリコンセンサーの特質とエネルギー損失の関係、MC を用いたシミュレーションプログラムについての知見を広げることができた。その原因の一つとして、宇宙線測定とそのシミュレーションを並行して実施したことがあげられる。宇宙線測定においては、より効率的で安定した測定環境にセットアップを改善し、様々な条件で何度も変更し測定した。その得られたデータにより、様々な条件にも対応して解析できるプログラムを作成することができた。シミュレーションプログラムにおいても、宇宙線測定の様々な条件に対応したシミュレーションを行うために普遍的なものを作成することができた。このプログラムでは、MC による宇宙線の生成やエネルギー損失の計算式の入力にいたるまで全ての工程を自作で製作した。これにより、行った宇宙線測定のセットアップに自由に近づけられることだけでなく、宇宙線測定全体の深い理解につなげることができた。

謝辞

本研究を行うにあたり、お世話になりました方々に紙面を借りてお礼申し上げます。

はじめに、このような素晴らしい実験に携わり勉強する機会を与えて下さった、高エネルギー物理学研究室の林井先生、宮林先生、下村先生、蜂谷先生に感謝いたします。直接ご指導いただきました蜂谷先生には、研究に関する知識だけでなく、物理という学問を学ぶ楽しさを教えていただきました。また、研究に対する姿勢や考え方についても、アドバイスしてくださいました。様々なことに興味を持ち挑戦する姿勢は、大学院に進学後の学生生活においても大切にしていきたいと思っています。また、INTT グループの中川さん、秋葉さんをはじめ、学生ミーティングにおいて、私たちの実験報告や疑問に対し長時間熱心に聞きアドバイスしてくださったおかけで、モチベーションを保ち研究に励むことができました。鈴木先輩には、テストベンチの操作方法やよく発生するハプニングなど日々の実験に欠かせない多くの知識を教えていただきました。私たちの知識不足な質問や疑問に、優しく的確に教えて下さいました。一緒に研究に取り組んだ 4 年生の森田さんには、1 人では気づくことのできない多くのアドバイスもらいと実験を進めるモチベーションを高めあうことができました。

私はこの 1 年、INTT グループに参加させていただき、テストベンチによる宇宙線測定や国内外での解析ワークショップ、英語での電話ミーティングなど学部生では経験できないような貴重な経験をさせていただきました。ハードウェアやプログラミングに対する知識が全くなく苦しんだ時もありましたが、多くの方の支えやアドバイスのおかげで少しずつ知識を増やし、自らの手で研究を成し遂げることができました。私がこの 1 年で教わり、身に着けたことは、これからの学生生活だけでなく社会に出てからも大切にしなければならないことで、心から感謝しております。

参考文献

[1] PDG 2018 Passage of Particles Through Matter.

[2] PDG 2019 Passage of Particles Through Matter.

[3] PDG 2019 AtomicNuclearProperties.

[4] TDR 2019 sPHENIX Technical Design Report.

[5] SPHENIX (2019, August 31) sPHENIX Retrieved 04:29, April 7, 2020 from `https://wiki.bnl.gov/sPHENIX/index.php?title=SPHENIX&oldid=5968`.

[6] 鈴木彩香 2020 「RHIC-sPHENIX　実験における中間飛跡検出器 INTT 用シリコンセンサーの性能評価, ビームテスト実験のデータ解析」, 修士論文, 奈良女子大学.

[7] 森田美羽 2020 「RHIC-sPHENIX　実験におけるバスエクステンダーの性能評価」, 卒業論文, 奈良女子大学.

[8] 一色萌衣 呉羽広子 杉野和音 2019 「RHIC-sPHENIX　実験における INTT シリコンモジュールの性能評価とテストベンチ構築」, 卒業論文, 奈良女子大学.

[9] 益田英知 2017 「RHIC-sPHENIX 実験におけるシリコンストリップ検出器の開発」, 修士論文, 立教大学大学院.

[10] 長島徹 2015 「RHIC-sPHENIX 実験 Run15　における FVTX 検出器を用いた高多重度トリガーシステムの開発と陽子＋陽子衝突系における方位角異方性の検証」, 修士論文, 立教大学大学院.

[11] 秋葉康之 2014『クォーク・グルーオン・プラズマの物理実験室で再現する宇宙の始まり』, 共立出版.

[12] 岸野正剛 2007 『現代　半導体デバイスの基礎』, オーム社.

付録 A

宇宙線解析プログラム

本解析は CERN によって開発が行われている「ROOT（ルート）」を用いて行った。

A.1　宇宙線解析プログラムの実行方法

??readtree.c, readtree.h, run_readtree.C, plot_readtree2.C の 4 つのファイルと宇宙線データの root ファイルを所定の箇所にいれ行う。
まずプログラムと root データファイルを取り出せるディレクトリで解析したい root ファイルと指定し、ROOT を立ち上げる。ROOT の中で readtree.c を読み込んだあと、r un_readtree.C を実行することで解析が行われ測定したヒストグラムの root ファイルが作成される。ファイル作成後、plot_readtree2.C を実行することで、ヒストグラムの確認することができる。

A.2　宇宙線解析のコード

Listing A.1　readtree.c

```
1  //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2  // Analysis program for INTT silicon sensor
3  // This is one of the files to get clustered hits hisotgram.
4  // You can draw dac amplitude(mV) histograms, and N hits per event hisrogram on a Canvas.
5  //
6  // Before drawing, you have to get four files, "readtree.c", "run_readtree.C", "plot_readtree2.C", "readtree.h".
7  // Please change root file name on "run_readtree.C" and "plot_readtree2.C".
8  // To draw histograms, you can write
9  // $ root -l rootfilename
10 //  ROOT> .L readtree.c;
11 // ROOT> .x run_readtree.C;   get "histofilename.root" file
12 // ROOT> .x plot_readtree2.C;  get histograms
13 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
14
15 #define readtree_cxx
16 #include "readtree.h"
17 #include <TH2.h>
18 #include <TStyle.h>
19 #include <TCanvas.h>
20 #include <algorithm>
21 #include <TH1.h>
22
23
24 double mV_adc2(double SmV){
25   float mV[9] = {270, 300, 450, 602, 750, 902, 1050, 1202, 1234};
26   int Adc;
```

```
27    for(int ii=0;ii<8;ii++){
28      if( mV[ii]<=SmV && SmV<mV[ii+1]){
29        Adc=ii;
30      }
31    }
32    return Adc;//*Adc;
33  }
34  double weight(int hitAdc){
35    float Dac[9] = {15, 23, 60, 98, 135, 173, 210, 248, 256};
36    float weight[8];
37    float tmpw = Dac[1]-Dac[0];
38    for(int i=0; i<8; i++){
39      weight[i] = (Dac[i+1]-Dac[i])/tmpw;
40    }
41    float w = 1./weight[hitAdc];
42    return w;
43  }
44  double mV_weight(int hitAdc){
45    float mV[9] = {270, 302, 450, 602, 750, 902, 1050, 1202, 1234};
46    float weight[8];
47    float tmpw = mV[1]-mV[0];
48    for(int i=0; i<8; i++){
49      weight[i] = (mV[i+1]-mV[i])/tmpw;
50    }
51    float w = 1./weight[hitAdc];
52    return w;
53  }
54  double mV_weight2(int hitmV){
55    float mV[9] = {270, 302, 450, 602, 750, 902, 1050, 1202, 1234};
56    float weight[8];
57    float w;
58    float tmpw = mV[1]-mV[0];
59    for(int i=0; i<8; i++){
60      weight[i] = (mV[i+1]-mV[i])/tmpw;
61    }
62    for(int i=0;i<8;i++){
63      if(hitmV>=mV[i]&&hitmV<mV[i+1]){
64        int tmp = i;
65        w = 1./weight[i];
66      }
67    }
68    return w;
69  }
70  double adc_mV(int hitAdc){
71    double Dac[9] = {15., 23., 60., 98., 135., 173., 210., 248., 256.};
72    double mV = ((Dac[hitAdc+1]*4.+210.)+(Dac[hitAdc]*4.+210.))/2.-2.;
73    return mV;
74  }
75  void readtree::process_hits(int nhit, int* chipArray, int* chanArray, int* adcArray, int* eventArray, int* bcoArray, int* bco_fullArray,
          double* summV_Ret, int* Nhits_Ret, int* chan_Ret, int* chip_Ret, int nclsnChip_Ret, int(*adc_Ret)[10]){
76    int hitChip[27][800];
77    int hitChan[27][800];
78    int hitAdc [27][800];
79    int hitEvent[27][800];
80    int hitBco[27][800];
81    int hitBco_full[27][800];
82    int nchip=0;
83    int nHitChip[27];
84    for(int i=0;i<27;i++){
85      nHitChip[i] = 0;
86    }
87    int tmp1, tmp2, tmp3, tmp4, tmp5, tmp6;
88    for(int ihit=0; ihit<nhit; ihit++){
89      int ichip = chipArray[ihit];
90      int ii = nHitChip[ichip];
91      nHitChip[ichip]++;
92      hitChip[ichip][ii] = chipArray[ihit];
93      hitChan[ichip][ii] = chanArray[ihit];
94      hitAdc [ichip][ii] = adcArray[ihit];
95      hitEvent[ichip][ii]= eventArray[ihit];
96      hitBco[ichip][ii]= bcoArray[ihit];
97      hitBco_full[ichip][ii]= bco_fullArray[ihit];
98    }
99    double nhit_1[9], nhit_2[9], nhit_3[9];
100   for(int ii=0; ii<9;ii++){
101     nhit_1[ii]=0.;
102     nhit_2[ii]=0.;
103     nhit_3[ii]=0.;
104   }
105   double nhit_adc1[9], nhit_adc2[9], nhit_adc3[9];
106   for(int ii=0; ii<9;ii++){
107     nhit_adc1[ii]=0.;
108     nhit_adc2[ii]=0.;
109     nhit_adc3[ii]=0.;
110   }
111 //////// sort channel order///////////////
112   for(int ichip=0; ichip<27; ichip++){
113     if(nHitChip[ichip]>0){
114       for(int ihit=0; ihit<nHitChip[ichip]; ihit++){
115         for(int jhit=ihit+1;jhit<nHitChip[ichip];jhit++){
116           if(hitChan[ichip][ihit]>hitChan[ichip][jhit]){
117             tmp1 = hitChan[ichip][ihit];
118             tmp2 = hitChip[ichip][ihit];
119             tmp3 = hitAdc[ichip][ihit];
```

```
120               tmp4 = hitEvent[ichip][ihit];
121               tmp5 = hitBco[ichip][ihit];;
122               tmp6 = hitBco_full[ichip][ihit];
123               hitChan[ichip][ihit] = hitChan[ichip][jhit];
124               hitChip[ichip][ihit] = hitChip[ichip][jhit];
125               hitAdc[ichip][ihit] = hitAdc[ichip][jhit];
126               hitEvent[ichip][ihit] = hitEvent[ichip][jhit];
127               hitBco[ichip][ihit]= hitBco[ichip][jhit];
128               hitBco_full[ichip][ihit]= hitBco_full[ichip][jhit];
129               hitChan[ichip][jhit] = tmp1;
130               hitChip[ichip][jhit] = tmp2;
131               hitAdc[ichip][jhit] = tmp3;
132               hitEvent[ichip][jhit] = tmp4;
133               hitBco[ichip][jhit]= tmp5;
134               hitBco_full[ichip][jhit]= tmp6;
135             }
136           }
137         }
138       }
139     }
140     for(int ichip=0; ichip<27; ichip++){
141       if(nHitChip[ichip]>1){
142         int tmpchip[5]={-1,-1,-1,-1,-1};
143         int tmpchan[5]={-1,-1,-1,-1,-1};
144         int noise =1;
145         int tmptmp=0;
146         for(int ihit=0; ihit<nHitChip[ichip]; ihit++){
147           if(hitChip[ichip][ihit]==hitChip[ichip][ihit+1]&&hitChan[ichip][ihit+1]==hitChan[ichip][ihit]){
148             noise++;
149             if(tmpchan[tmptmp]!=hitChan[ichip][ihit]&&tmpchip[tmptmp]!=hitChip[ichip][ihit]){
150               tmptmp++;
151             }
152             tmpchan[tmptmp]=hitChan[ichip][ihit];
153             tmpchip[tmptmp]=hitChip[ichip][ihit];
154           }
155         }
156         for(int ttt=0;ttt<tmptmp+1;ttt++){
157           for(int ihit=0; ihit<nHitChip[ichip]+1; ihit++){
158             if(hitChip[ichip][ihit]==tmpchip[ttt]&&hitChan[ichip][ihit]==tmpchan[ttt]){
159             }
160           }
161         }
162       }
163     }
164   //++++++++++++check strip number+++++++++++++++
165     int nhitmap=0;
166     double summV;
167     int sumChan;
168     int ncls = 0;
169     for(int ichip=0; ichip<27; ichip++){
170       if(nHitChip[ichip]>0){
171         int start = 0;
172         int Nhits =1;
173         int end;
174         for(int ihit=0; ihit<nHitChip[ichip]; ihit++){
175           if(Nhits==nHitChip[ichip]){
176             end = ihit;
177             summV=0;
178             sumChan =0;
179             for(int i=start; i<end+1; i++){
180               summV += adc_mV(hitAdc[ichip][i]);
181               sumChan += hitChan[ichip][i];
182             }
183             if(Nhits==1){
184               for(int ii=0; ii<8;ii++){
185                 if(mV_adc2(summV)==ii)nhit_1[ii]++;
186               }
187             }
188             else if(Nhits==2){
189               for(int ii=0; ii<8;ii++){
190                 if(mV_adc2(summV)==ii)nhit_2[ii]++;
191               }
192             }
193             else if(Nhits==3){
194               for(int ii=0; ii<8;ii++){
195                 if(mV_adc2(summV)==ii)nhit_3[ii]++;
196               }
197             }
198             start = end+1;
199             ncls++;
200             Nhits = 1;
201           }
202           else if(hitChan[ichip][ihit+1]-hitChan[ichip][ihit]==1){
203             Nhits++;
204           }
205           else{
206             end = ihit;
207             summV=0;
208             sumChan =0;
209             for(int i=start; i<end+1; i++){
210               summV += adc_mV(hitAdc[ichip][i]);
211               sumChan += hitChan[ichip][i];
212             }
213             if(Nhits==1){
```

```
214               for(int ii=0; ii<8;ii++){
215                 if(mV_adc2(summV)==ii)nhit_1[ii]++;
216               }
217             }
218             else if(Nhits==2){
219               for(int ii=0; ii<8;ii++){
220                 if(mV_adc2(summV)==ii)nhit_2[ii]++;
221               }
222             }
223             else if(Nhits==3){
224               for(int ii=0; ii<8;ii++){
225                 if(mV_adc2(summV)==ii)nhit_3[ii]++;
226               }
227             }
228             start = end+1;
229             ncls++;
230             Nhits = 1;
231           }
232         }
233       }
234     }
235   //+++++++++++++++++++++++++++++++++++
236     if(ncls>0){
237       for(int ichip=0; ichip<27; ichip++){
238         if(nHitChip[ichip]>0){
239           for(int ihit=0; ihit<nHitChip[ichip]; ihit++){
240             multiclschip->Fill(hitChan[ichip][ihit],hitChip[ichip][ihit]);
241           }
242         }
243       }
244     }
245   //++++++++++Fill per strip number+++++++++++++++
246     if(ncls>0){                    //cluster number for analysis
247       nhitmap=0;
248       int Ncls = 0;                  //number of strips (same as ncls)
249       for(int ichip=0; ichip<27; ichip++){
250         int NHITCHIP=0;  //number of single hits per chip per event
251         int single_ihit[20];
252         int single_0_ihit[20];
253         if(nHitChip[ichip]>0){
254           int start = 0;
255           int Nhits =1;
256           int end;
257           for(int ihit=0; ihit<nHitChip[ichip]; ihit++){
258             if(Nhits==nHitChip[ichip]){
259               end = ihit;
260               summV=0;
261               sumChan =0;
262               for(int i=start; i<end+1; i++){
263                 summV += adc_mV(hitAdc[ichip][i]);  //clustered hit's energy
264                 sumChan += hitChan[ichip][i];      //clustered hit's sum channel number
265               }
266               summV_Ret[Ncls] = summV;
267               Nhits_Ret[Ncls] = Nhits;
268               chip_Ret[Ncls] = hitChip[ichip][ihit]; //RET clustered hit's channel number
269               chan_Ret[Ncls] = sumChan/Nhits;      //RET clustered hit's avarage channel number
270               hnhits->Fill(Nhits);
271   //+++++++++ Fill single hit ++++++++++++
272               if(Nhits==1){
273                 adc_Ret[Ncls][0] = hitAdc[ichip][start];
274                 single_ihit[NHITCHIP]=start;
275                 NHITCHIP++;
276                 if(hitAdc[ichip][start]!=-1){      //ADC value for analysis
277                   SinglehitAdc2->Fill(summV,mV_weight2(summV));
278                   SingleAdc->Fill(adc_mV(hitAdc[ichip][start]), mV_weight2(adc_mV(hitAdc[ichip][start])));
279                   if(adc_mV(hitAdc[ichip][start])>5000.&&adc_mV(hitAdc[ichip][start])<0.)cout<<"Single mV="<<adc_mV(hitAdc[ichip][start
                          ])<<endl;
280                   for(int ii=0; ii<8;ii++){
281                     if(mV_adc2(summV)==ii)nhit_1[ii]++;
282                   }
283                 }
284                 else if(hitAdc[ichip][start]==0)single_0_ihit[NHITCHIP]=start;
285               }
286   //+++++++++ Fill double hits ++++++++++++
287               else if(Nhits==2){
288                 adc_Ret[Ncls][0] = hitAdc[ichip][start];
289                 adc_Ret[Ncls][1] = hitAdc[ichip][start+1];
290                 SumDoublehitAdc2->Fill(summV,mV_weight2(summV));
291                 DAdc->Fill(adc_mV(hitAdc[ichip][start]),mV_weight2(adc_mV(hitAdc[ichip][start])));
292                 DAdc->Fill(adc_mV(hitAdc[ichip][start+1]), mV_weight2(adc_mV(hitAdc[ichip][start+1])));
293                 if(adc_mV(hitAdc[ichip][start])>5000.&&adc_mV(hitAdc[ichip][start])<0.) cout<<"D mV="<<adc_mV(hitAdc[ichip][start])<<
                        endl;
294                   if(adc_mV(hitAdc[ichip][start+1])>5000.&&adc_mV(hitAdc[ichip][start+1])<0.)cout<<"D2 mV="<<adc_mV(hitAdc[ichip][
                          start+1])<<endl;
295                 Adc_2hit->Fill(hitAdc[ichip][start],hitAdc[ichip][start+1]);
296                 chan_2hit->Fill(hitChan[ichip][start],hitChan[ichip][start+1]);
297                 if(hitAdc[ichip][start]==0&&hitAdc[ichip][start+1]==0){
298                 }
299
300                 for(int ii=0; ii<8;ii++){
301                   if(mV_adc2(summV)==ii)nhit_2[ii]++;
302                 }
303               }
304   //+++++++++ Fill triple hits ++++++++++++
```

```
305             else if(Nhits==3){
306               SumTriplehitAdc2->Fill(summV,mV_weight2(summV));
307               TAdc->Fill(adc_mV(hitAdc[ichip][start]), mV_weight2(adc_mV(hitAdc[ichip][start])));
308               TAdc->Fill(adc_mV(hitAdc[ichip][start+1]), mV_weight2(adc_mV(hitAdc[ichip][start+1])));
309               TAdc->Fill(adc_mV(hitAdc[ichip][start+2]), mV_weight2(adc_mV(hitAdc[ichip][start+2])));
310                 if(adc_mV(hitAdc[ichip][start])>5000.&&adc_mV(hitAdc[ichip][start])<0.)cout<<"T mV="<<adc_mV(hitAdc[ichip][start])
                        <<endl;
311                 if(adc_mV(hitAdc[ichip][start+1])>5000.&&adc_mV(hitAdc[ichip][start+1])<0.)cout<<"T2 mV="<<adc_mV(hitAdc[ichip][
                        start+1])<<endl;
312                 if(adc_mV(hitAdc[ichip][start+2])>5000.&&adc_mV(hitAdc[ichip][start+2])<0.)cout<<"T2 mV="<<adc_mV(hitAdc[ichip][
                        start+1])<<endl;
313               for(int ii=0; ii<8;ii++){
314                 if(mV_adc2(summV)==ii)nhit_3[ii]++;
315               }
316             }
317             start = end+1;
318             Ncls++;
319             Nhits = 1;
320           }
321           else if(hitChan[ichip][ihit+1]-hitChan[ichip][ihit]==1){
322             Nhits++;
323           }
324           else{
325             end = ihit;
326             summV=0;
327             sumChan =0;
328             for(int i=start; i<end+1; i++){
329               summV += adc_mV(hitAdc[ichip][i]);
330               sumChan += hitChan[ichip][i];
331             }
332             summV_Ret[Ncls] = summV;
333             Nhits_Ret[Ncls] = Nhits;
334             chip_Ret[Ncls] = hitChip[ichip][ihit];
335             chan_Ret[Ncls] = sumChan/Nhits;
336             hnhits->Fill(Nhits);
337 //********* Fill single hit ++++++++++++
338             if(Nhits==1){
339               adc_Ret[Ncls][0] = hitAdc[ichip][start];
340               single_ihit[NHITCHIP]=start;
341               NHITCHIP++;
342               if(hitAdc[ichip][start]!=-1){
343                 SinglehitAdc2->Fill(summV,mV_weight2(summV));
344                 SingleAdc->Fill(adc_mV(hitAdc[ichip][start]), mV_weight2(adc_mV(hitAdc[ichip][start])));
345                 if(adc_mV(hitAdc[ichip][start])>5000.&&adc_mV(hitAdc[ichip][start])<0.)cout<<"single mV="<<adc_mV(hitAdc[ichip][start
                        ])<<endl;
346                 for(int ii=0; ii<8;ii++){
347                   if(mV_adc2(summV)==ii)nhit_1[ii]++;
348                 }
349               }
350               else if(hitAdc[ichip][start]==0)single_0_ihit[NHITCHIP]=start;
351             }
352 //********* Fill double hits ++++++++++++
353             else if(Nhits==2){
354               adc_Ret[Ncls][0] = hitAdc[ichip][start];
355               adc_Ret[Ncls][1] = hitAdc[ichip][start+1];
356               SumDoublehitAdc2->Fill(summV,mV_weight2(summV));
357               DAdc->Fill(adc_mV(hitAdc[ichip][start]), mV_weight2(adc_mV(hitAdc[ichip][start])));
358               DAdc->Fill(adc_mV(hitAdc[ichip][start+1]), mV_weight2(adc_mV(hitAdc[ichip][start+1])));
359                 if(adc_mV(hitAdc[ichip][start])>5000.&&adc_mV(hitAdc[ichip][start])<0.)cout<<"D mV="<<adc_mV(hitAdc[ichip][start])
                        <<endl;
360                 if(adc_mV(hitAdc[ichip][start+1])>5000.&&adc_mV(hitAdc[ichip][start+1])<0.)cout<<"D2 mV="<<adc_mV(hitAdc[ichip][
                        start+1])<<endl;
361               Adc_2hit->Fill(hitAdc[ichip][start],hitAdc[ichip][start+1]);
362               chan_2hit->Fill(hitChan[ichip][start],hitChan[ichip][start+1]);
363               for(int ii=0; ii<8;ii++){
364                 if(mV_adc2(summV)==ii)nhit_2[ii]++;
365               }
366             }
367 //********* Fill triple hits ++++++++++++
368             else if(Nhits==3){
369               SumTriplehitAdc2->Fill(summV,mV_weight2(summV));
370               TAdc->Fill(adc_mV(hitAdc[ichip][start]) , mV_weight2(adc_mV(hitAdc[ichip][start])));
371               TAdc->Fill(adc_mV(hitAdc[ichip][start+1]), mV_weight2(adc_mV(hitAdc[ichip][start+1])));
372               TAdc->Fill(adc_mV(hitAdc[ichip][start+2]), mV_weight2(adc_mV(hitAdc[ichip][start+2])));
373                 if(adc_mV(hitAdc[ichip][start])>5000. &&adc_mV(hitAdc[ichip][start])<0.)cout<<"T mV="<<adc_mV(hitAdc[ichip][start])
                        <<endl;
374                 if(adc_mV(hitAdc[ichip][start+1])>5000.&&adc_mV(hitAdc[ichip][start+1])<0.)cout<<"T2 mV="<<adc_mV(hitAdc[ichip][
                        start+1])<<endl;
375                 if(adc_mV(hitAdc[ichip][start+2])>5000.&&adc_mV(hitAdc[ichip][start+2])<0.)cout<<"T2 mV="<<adc_mV(hitAdc[ichip][
                        start+1])<<endl;
376               for(int ii=0; ii<8;ii++){
377                 if(mV_adc2(summV)==ii)nhit_3[ii]++;
378               }
379             }
380             start = end+1;
381             Ncls++;
382             Nhits = 1;
383           }
384         }
385       }
386 //***************single and adc=0 hit**************************************
387       if(NHITCHIP>0){
388         for(int j=0;j<NHITCHIP;j++){
389           if(hitAdc[ichip][single_0_ihit[j]]==0){
390             adc_0hit->Fill(hitChan[ichip][single_0_ihit[j]],hitChip[ichip][single_0_ihit[j]]);
```

```
391             }
392           }
393         }
394
395 //+++++++++++++++separate single hits++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
396         if(NHITCHIP==2){
397           for(int j=0;j<NHITCHIP;j++){
398             NHITChipmV->Fill(adc_mV(hitAdc[ichip][single_ihit[j]]),mV_weight2(adc_mV(hitAdc[ichip][single_ihit[j]])));
399           }
400           NHITChip->Fill(hitAdc[ichip][single_ihit[0]],hitAdc[ichip][single_ihit[1]]);
401           chan_1hit->Fill(hitChan[ichip][single_ihit[0]],hitChan[ichip][single_ihit[1]]);
402           chan_1hit2->Fill(hitChan[ichip][single_ihit[1]]-hitChan[ichip][single_ihit[0]],hitChip[ichip][single_ihit[0]]);
403         }
404       }
405           nclsnChip->Fill(Ncls);
406           nclsnChip_Ret = Ncls;
407   }
408 };
409 void readtree::Loop(const int entry)
410 {
411     if (fChain == 0) return;
412     Long64_t nentries = fChain->GetEntriesFast();
413     if(entry>=0) m_jentry=entry;
414     Long64_t nbytes = 0, nb = 0;
415     int nhit=0;
416     int prenhit=0;
417     int noise=0;
418     int m_nbco[50000];
419     int m_nbco_full[50000];
420     int m_chanArray[50000];
421     int m_chipArray[50000];
422     int m_adcArray[50000];
423     int m_eventArray[50000];
424     int m_bcoArray[50000];
425     int m_bco_fullArray[50000];
426     int m_prenbco_full[50000];
427     int m_prechanArray[50000];
428     int m_prechipArray[50000];
429     int m_preadcArray[50000];
430     int m_preeventArray[50000];
431     int m_prebcoArray[50000];
432     int m_prebco_fullArray[50000];
433     int sa_bco;
434     int sa_bcofull;
435     int noisehit=0;
436     int cosmichit=0;
437     int prenoisehit=0;
438     int precosmichit=0;
439     int processhit=0;
440     int nhit2=0;
441     int prenhit2=0;
442     int noise2=0;
443     int m_nbco2[50000];
444     int m_nbco_full2[50000];
445     int m_chanArray2[50000];
446     int m_chipArray2[50000];
447     int m_adcArray2[50000];
448     int m_eventArray2[50000];
449     int m_bcoArray2[50000];
450     int m_bco_fullArray2[50000];
451     int m_prenbco_full2[50000];
452     int m_prechanArray2[50000];
453     int m_prechipArray2[50000];
454     int m_preadcArray2[50000];
455     int m_preeventArray2[50000];
456     int m_prebcoArray2[50000];
457     int m_prebco_fullArray2[50000];
458     int sa_bco2;
459     int sa_bcofull2;
460     int noisehit2=0;
461     int cosmichit2=0;
462     int prenoisehit2=0;
463     int precosmichit2=0;
464     int processhit2=0;
465     double summV_Ret[10];
466     int  Nhits_Ret[10];
467     double summV_Ret8[10];
468     double summV_Ret6[10];
469     int  Nhits_Ret6[10];
470     int  Nhits_Ret8[10];
471     int  chan_Ret[10] ;
472     int  chan_Ret6[10];
473     int  chan_Ret8[10];
474     int  chip_Ret[10] ;
475     int  chip_Ret6[10];
476     int  chip_Ret8[10];
477     int  ncls_Ret;
478     int  ncls_Ret6;
479     int  ncls_Ret8;
480     int  adc_Ret[10][10] ;
481     int  adc_Ret6[10][10];
482     int  adc_Ret8[10][10];
483     for(int i=0;i<10;i++){
484       summV_Ret[i] = -1;
```

```
485         Nhits_Ret[i] = -1;
486         summV_Ret8[i] = -1;
487         summV_Ret6[i] = -1;
488         Nhits_Ret6[i] = -1;
489         Nhits_Ret8[i] = -1;
490         chan_Ret[i]  = -1;
491         chan_Ret6[i] = -1;
492         chan_Ret8[i] = -1;
493         chip_Ret[i]  = -1;
494         chip_Ret6[i] = -1;
495         chip_Ret8[i] = -1;
496         for(int ii=0;ii<10;ii++){
497           adc_Ret[ii][i] = -1;
498           adc_Ret6[ii][i] = -1;
499           adc_Ret8[ii][i] = -1;
500         }
501       }
502       int nhit_Ret8=0;
503       int nhit_Ret6=0;
504       int nstrip8=0;
505       int nstrip6=0;
506       ncls_Ret6 = -1;
507       ncls_Ret8 = -1;
508       ncls_Ret = -1;
509       for (; m_jentry<nentries+1;m_jentry++) {
510         Long64_t ientry = LoadTree(m_jentry);
511         if (ientry < 0) break;
512         nb = fChain->GetEntry(m_jentry); nbytes += nb;
513   // ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
514         if( (m_prebco!=bco) && (m_prebcofull!=bco_full) ){
515           sa_bco = m_prebco-bco;
516           sa_bcofull = m_prebcofull-bco_full;
517           bco_interval->Fill(sa_bco);
518           bcofull_interval->Fill(sa_bcofull);
519           sa_correlation->Fill(sa_bco,sa_bcofull);
520           if(sa_bcofull!=-1){
521             for(int i=0;i<nhit;i++){
522                   adc_bcofull->Fill(m_adcArray[i],weight(m_adcArray[i]));
523             }
524           }
525           m_prebco = bco;
526           m_prebcofull = bco_full;
527   //++++++++++++check same channel and chip data+++++++++++++++++++++++++
528           for(int i=0;i<nhit;i++){
529             for(int ii=0;ii<prenhit;ii++){
530               if(m_prechanArray[ii]==m_chanArray[i]&&m_prechipArray[ii]==m_chipArray[i]){
531                 noise++;
532                 m_noiseAdc->Fill(m_preadcArray[ii],weight(m_preadcArray[ii]));
533                 m_noiseAdc->Fill(m_adcArray[i],weight(m_adcArray[i]));
534                 noisehit=1;
535               }
536               else {
537                 cosmichit=1;
538               }
539             }
540           }
541   //++++++++++++check same channel and chip data+++++++++++++++++++++++++
542           for(int i=0;i<nhit2;i++){
543             for(int ii=0;ii<prenhit2;ii++){
544               if(m_prechanArray2[ii]==m_chanArray2[i]&&m_prechipArray2[ii]==m_chipArray2[i]){
545                 noise2++;
546                 m_noiseAdc->Fill(m_preadcArray2[ii],weight(m_preadcArray2[ii]));
547                 m_noiseAdc->Fill(m_adcArray2[i],weight(m_adcArray2[i]));
548                 noisehit2=1;
549               }
550               else{
551                 cosmichit2=1;
552               }
553             }
554           }
555           int HIT=0;
556           processhit=0;
557           if(noisehit==0&&prenoisehit==0)processhit=1;
558           processhit2=0;
559           if(noisehit2==0&&prenoisehit2==0)processhit2=1;
560           //!!!!!!!!!!!!!!!!!!for two modules!!!!!!!!!!!!!!!!!!!!
561           if(prenhit>0&&prenhit2>0){
562             if(processhit==1){
563               if(processhit2==1){
564           //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
565           //!!!!!!!!!!!!!!!!!!for one modules(8)!!!!!!!!!!!!!!!!!!!!
566   /*    if(prenhit>0){
567             if(cosmichit==1&&precosmichit==1&&noisehit==0&&prenoisehit==0){
568   */    //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
569                   for(int i=0;i<10;i++){
570                     summV_Ret[i] = -1;
571                     summV_Ret6[i] = -1;
572                     summV_Ret8[i] = -1;
573                     Nhits_Ret[i] = -1;
574                     Nhits_Ret6[i] = -1;
575                     Nhits_Ret8[i] = -1;
576                     chan_Ret[i]  = -1;
577                     chan_Ret6[i] = -1;
578                     chan_Ret8[i] = -1;
```

```
579                 chip_Ret[i]  = -1;
580                 chip_Ret6[i] = -1;
581                 chip_Ret8[i] = -1;
582                 for(int ii=0;ii<10;ii++){
583                   adc_Ret[ii][i] = -1;
584                   adc_Ret6[ii][i] = -1;
585                   adc_Ret8[ii][i] = -1;
586                 }
587               }
588               nhit_Ret8=0;
589               nhit_Ret6=0;
590               nstrip8=0;
591               nstrip6=0;
592               ncls_Ret6 = -1;
593               ncls_Ret8 = -1;
594               ncls_Ret = -1;
595               for(int i=0;i<prenhit;i++){
596                 Wadc->Fill(m_preadcArray[i], weight(m_preadcArray[i]));
597                 //--singlemV->Fill(adc_mV(m_preadcArray[i]),mV_weight2(adc_mV(m_preadcArray[i])));
598               }
599               for(int i=0;i<prenhit2;i++){
600                 Wadc->Fill(m_preadcArray2[i], weight(m_preadcArray2[i]));
601                 //--singlemV->Fill(adc_mV(m_preadcArray2[i]),mV_weight2(adc_mV(m_preadcArray2[i])));
602               }
603 // ++++++++++ process hits++++++++++++++++++++++
604               process_hits(prenhit, m_prechipArray, m_prechanArray, m_preadcArray, m_preeventArray, m_prebcoArray,
                       m_prebco_fullArray, summV_Ret, Nhits_Ret, chan_Ret, chip_Ret, ncls_Ret, adc_Ret);
605               if(ncls_Ret>0){
606                 NCLS8->Fill(ncls_Ret);
607               }
608               ncls_Ret8 = ncls_Ret;
609               for(int I=0;I<10;I++){
610                 if(summV_Ret[I]!=-1){
611                   summV_Ret8[nhit_Ret8] = summV_Ret[I];
612                   Nhits_Ret8[nhit_Ret8] = Nhits_Ret[I];
613                   chan_Ret8[nhit_Ret8] = chan_Ret[I];
614                   chip_Ret8[nhit_Ret8] = chip_Ret[I];
615                   for(int II=0;II<10;II++){
616                     adc_Ret8[nhit_Ret8][nstrip8] = adc_Ret[I][II];
617                     nstrip8++;
618                   }
619                   nhit_Ret8++;
620                 }
621               }
622               for(int i=0;i<10;i++){
623                 summV_Ret[i] = -1;
624                 Nhits_Ret[i] = -1;
625                 chan_Ret[i]  = -1;
626                 chip_Ret[i]  = -1;
627                 for(int ii=0;ii<10;ii++){
628                   adc_Ret[ii][i] = -1;
629                 }
630               }
631               ncls_Ret = 0;
632               process_hits(prenhit2, m_prechipArray2, m_prechanArray2, m_preadcArray2, m_preeventArray2, m_prebcoArray2,
                       m_prebco_fullArray2, summV_Ret, Nhits_Ret, chan_Ret, chip_Ret, ncls_Ret, adc_Ret);
633 // ++++++++++++++analysis per module++++++++++++
634               if(ncls_Ret>0){
635                 NCLS6->Fill(ncls_Ret);
636               }
637               ncls_Ret6 = ncls_Ret;
638               for(int I=0;I<10;I++){
639                 if(summV_Ret[I]!=-1){
640                   summV_Ret6[nhit_Ret6] = summV_Ret[I];
641                   Nhits_Ret6[nhit_Ret6] = Nhits_Ret[I];
642                   chan_Ret6[nhit_Ret6] = chan_Ret[I];
643                   chip_Ret6[nhit_Ret6] = chip_Ret[I];
644                   for(int II=0;II<10;II++){
645                     adc_Ret6[nhit_Ret6][nstrip6] = adc_Ret[I][II];
646                     nstrip6++;
647                   }
648                   nhit_Ret6++;
649                 }
650               }
651               for(int i=0;i<10;i++){
652                 summV_Ret[i] = -1;
653                 Nhits_Ret[i] = -1;
654                 chan_Ret[i]  = -1;
655                 chip_Ret[i]  = -1;
656                 for(int ii=0;ii<10;ii++){
657                   adc_Ret[ii][i] = -1;
658                 }
659               }
660               int chipselection=0;
661               int chanselection=0;
662               ncls_Ret = 0;
663               if(nhit_Ret8==1&&nhit_Ret6==1){
664                 NSTRIP6->Fill(Nhits_Ret6[0]);
665                 NSTRIP8->Fill(Nhits_Ret8[0]);
666                 CHANDIFF->Fill(chan_Ret6[0], chan_Ret8[0]);
667                 CHIPDIFF->Fill(chip_Ret6[0], chip_Ret8[0]);
668                 if(chip_Ret6[0]>2&&chip_Ret6[0]<14&&chip_Ret8[0]<12){
669                   CHIPDIFFSAh->Fill(chip_Ret6[0]-chip_Ret8[0]);
670                   CHANDIFFS->Fill(chan_Ret6[0], chan_Ret8[0]);
```

```
671                 if(chan_Ret6[0]>63&&chan_Ret8[0]>63){
672                   CHANDIFFSAh->Fill(chan_Ret6[0]-chan_Ret8[0]);
673                   chanselection=1;
674                 }
675                 chipselection=1;
676               }
677               else if(chip_Ret6[0]>15&&chip_Ret8[0]>13&&chip_Ret8[0]<25){
678                 CHIPDIFFSAh->Fill(chip_Ret6[0]-chip_Ret8[0]);
679                 CHANDIFFS->Fill(chan_Ret6[0], chan_Ret8[0]);
680                 if(chan_Ret6[0]>63&&chan_Ret8[0]>63){
681                   CHANDIFFSAh->Fill(chan_Ret6[0]-chan_Ret8[0]);
682                   chanselection=1;
683                 }
684                 chipselection=1;
685               }
686               //event selection
687               if(chipselection==1&&chanselection==1&&(chip_Ret6[0]-chip_Ret8[0])<2.673+1.017&&(chip_Ret6[0]-chip_Ret8[0])
                      >2.673-1.017&&(chan_Ret6[0]-chan_Ret8[0])<27&&(chan_Ret6[0]-chan_Ret8[0])>-19){
688                 for(int I=0;I<nhit_Ret6;I++){
689                     for(int II=0;II<nstrip6;II++){
690                       if(adc_Ret6[II][I]>=0)originalenergyloss->Fill(adc_mV(adc_Ret6[II][I]),mV_weight2(adc_mV(adc_Ret6[II][I])));
691                     }
692                 }
693                 for(int I=0;I<nhit_Ret8;I++){
694                     for(int II=0;II<nstrip8;II++){
695                       if(adc_Ret8[II][I]>=0)originalenergyloss->Fill(adc_mV(adc_Ret8[II][I]),mV_weight2(adc_mV(adc_Ret8[II][I])));
696                     }
697                 }
698                 double gain = 190.;
699                 double RecmV1 = 0.;
700                 double RecmV2 = 0.;
701                 if(Nhits_Ret8[0]==1){
702                   RecmV1 = adc_mV(adc_Ret8[0][0])-gain;//+-75mV
703                   RECELOSS1->Fill(RecmV1);//+-75mV
704                 }
705                 if(Nhits_Ret6[0]==1){
706                   RecmV1 = adc_mV(adc_Ret6[0][0])-gain;//+-75mV
707                   RECELOSS1->Fill(RecmV1);//+-75mV
708                 }
709                 if(Nhits_Ret8[0]==2){
710                   RecmV2 += adc_mV(adc_Ret6[0][0])-gain;//+-75mV
711                   RecmV2 += adc_mV(adc_Ret6[0][1])-gain;//+-75mV
712                   RECELOSS2->Fill(RecmV2);//+-75mV
713                 }
714                 if(Nhits_Ret6[0]==2){
715                   RecmV2 += adc_mV(adc_Ret6[0][0])-gain;//+-75mV
716                   RecmV2 += adc_mV(adc_Ret6[0][1])-gain;//+-75mV
717                   RECELOSS2->Fill(RecmV2);//+-75mV
718                 }
719                 if(Nhits_Ret8[0]==1)ELOSSCLS1__S->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
720                 if(Nhits_Ret6[0]==1)ELOSSCLS1__S->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
721                 if(Nhits_Ret8[0]==2){
722                   ELOSSCLS2__S->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
723                   ADCDIFF->Fill(adc_mV(adc_Ret8[0][0]),adc_mV(adc_Ret8[0][1]));
724                 }
725                 if(Nhits_Ret6[0]==2){
726                   ELOSSCLS2__S->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
727                   ADCDIFF->Fill(adc_mV(adc_Ret6[0][0]),adc_mV(adc_Ret6[0][1]));
728                 }
729                 if(Nhits_Ret8[0]==1&&Nhits_Ret6[0]==2){
730                   ELOSSTRANS ->Fill(summV_Ret8[0], summV_Ret6[0]);
731                   ELOSSCLS1_S ->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
732                   ELOSSCLS2_S ->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
733                 }
734                 if(Nhits_Ret6[0]==1&&Nhits_Ret8[0]==2){
735                   ELOSSTRANS ->Fill(summV_Ret6[0], summV_Ret8[0]);
736                   ELOSSCLS1_S ->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
737                   ELOSSCLS2_S ->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
738                 }
739               }
740               if(chip_Ret6[0]<15&&chip_Ret8[0]<14){
741                 CHANDIFFSA1->Fill(TMath::Abs(chan_Ret6[0]-chan_Ret8[0]));
742                 ELOSSDIFF1->Fill(summV_Ret6[0], summV_Ret8[0]);
743                 if(Nhits_Ret8[0]==1)ELOSSCLS1_8_1->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
744                 if(Nhits_Ret6[0]==1)ELOSSCLS1_6_1->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
745                 if(Nhits_Ret8[0]==2)ELOSSCLS2_8_1->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
746                 if(Nhits_Ret6[0]==2)ELOSSCLS2_6_1->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
747                 if(Nhits_Ret8[0]==1&&Nhits_Ret6[0]==2){
748                   ELOSSTRAN1->Fill(summV_Ret8[0], summV_Ret6[0]);
749                   ELOSSCLS1_1->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
750                   ELOSSCLS2_1->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
751                 }
752                 if(Nhits_Ret6[0]==1&&Nhits_Ret8[0]==2){
753                   ELOSSTRAN1->Fill(summV_Ret6[0], summV_Ret8[0]);
754                   ELOSSCLS1_1->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
755                   ELOSSCLS2_1->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
756                 }
757               }
758               else if(chip_Ret6[0]<15){
759                 CHANDIFFSA2->Fill(TMath::Abs(chan_Ret6[0]-chan_Ret8[0]));
760                 ELOSSDIFF2->Fill(summV_Ret6[0], summV_Ret8[0]);
761                 if(Nhits_Ret8[0]==1)ELOSSCLS1_8_2->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
762                 if(Nhits_Ret6[0]==1)ELOSSCLS1_6_2->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
763                 if(Nhits_Ret8[0]==2)ELOSSCLS2_8_2->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
```

```
764                     if(Nhits_Ret6[0]==2)ELOSSCLS2_6_2->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
765                     if(Nhits_Ret8[0]==1&&Nhits_Ret6[0]==2){
766                       ELOSSTRAN2->Fill(summV_Ret8[0], summV_Ret6[0]);
767                       ELOSSCLS1_2->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
768                       ELOSSCLS2_2->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
769                     }
770                     if(Nhits_Ret6[0]==1&&Nhits_Ret8[0]==2){
771                       ELOSSTRAN2->Fill(summV_Ret6[0], summV_Ret8[0]);
772                       ELOSSCLS1_2->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
773                       ELOSSCLS2_2->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
774                     }
775                   }
776                   else if(chip_Ret6[0]>14&&chip_Ret8[0]>13){
777                     CHANDIFFSA3->Fill(TMath::Abs(chan_Ret6[0]-chan_Ret8[0]));
778                     ELOSSDIFF3->Fill(summV_Ret6[0], summV_Ret8[0]);
779                     if(Nhits_Ret8[0]==1)ELOSSCLS1_8_3->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
780                     if(Nhits_Ret6[0]==1)ELOSSCLS1_6_3->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
781                     if(Nhits_Ret8[0]==2)ELOSSCLS2_8_3->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
782                     if(Nhits_Ret6[0]==2)ELOSSCLS2_6_3->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
783                     if(Nhits_Ret8[0]==1&&Nhits_Ret6[0]==2){
784                       ELOSSTRAN3->Fill(summV_Ret8[0], summV_Ret6[0]);
785                       ELOSSCLS1_3->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
786                       ELOSSCLS2_3->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
787                     }
788                     if(Nhits_Ret6[0]==1&&Nhits_Ret8[0]==2){
789                       ELOSSTRAN3->Fill(summV_Ret6[0], summV_Ret8[0]);
790                       ELOSSCLS1_3->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
791                       ELOSSCLS2_3->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
792                     }
793                   }
794                   else if(chip_Ret6[0]>14){
795                     CHANDIFFSA4->Fill(TMath::Abs(chan_Ret6[0]-chan_Ret8[0]));
796                     ELOSSDIFF4->Fill(summV_Ret6[0], summV_Ret8[0]);
797                     if(Nhits_Ret8[0]==1)ELOSSCLS1_8_4->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
798                     if(Nhits_Ret6[0]==1)ELOSSCLS1_6_4->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
799                     if(Nhits_Ret8[0]==2)ELOSSCLS2_8_4->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
800                     if(Nhits_Ret6[0]==2)ELOSSCLS2_6_4->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
801                     if(Nhits_Ret8[0]==1&&Nhits_Ret6[0]==2){
802                       ELOSSTRAN4->Fill(summV_Ret8[0], summV_Ret6[0]);
803                       ELOSSCLS1_4->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
804                       ELOSSCLS2_4->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
805                     }
806                     if(Nhits_Ret6[0]==1&&Nhits_Ret8[0]==2){
807                       ELOSSTRAN4->Fill(summV_Ret6[0], summV_Ret8[0]);
808                       ELOSSCLS1_4->Fill(summV_Ret6[0], mV_weight2(summV_Ret6[0]));
809                       ELOSSCLS2_4->Fill(summV_Ret8[0], mV_weight2(summV_Ret8[0]));
810                     }
811                   }
812                   ELOSSDIFF->Fill(summV_Ret6[0], summV_Ret8[0]);
813                   ELOSSDIFFSA->Fill(TMath::Abs(summV_Ret6[0]-summV_Ret8[0]));
814
815                   if(Nhits_Ret8[0]==1&&Nhits_Ret6[0]==2){
816                     ELOSSTRAN->Fill(summV_Ret8[0], summV_Ret6[0]);
817                     ELOSSTRANSA->Fill(TMath::Abs(summV_Ret6[0]-summV_Ret8[0]));
818                   }
819                   if(Nhits_Ret6[0]==1&&Nhits_Ret8[0]==2){
820                     ELOSSTRAN->Fill(summV_Ret6[0], summV_Ret8[0]);
821                     ELOSSTRANSA->Fill(TMath::Abs(summV_Ret6[0]-summV_Ret8[0]));
822                   }
823                 }
824               } //!!!!!!!!!!!!!!!!!for two modules!!!!!!!!!!!!!!!!!!!!!!!!!!
825             }
826           }
827   //+++++++++++++++copy previous data++++++++++++++++++++++
828           precosmichit = cosmichit;
829           precosmichit2 = cosmichit2;
830           prenoisehit = noisehit;
831           prenoisehit2 = noisehit2;
832           cosmichit=0;
833           cosmichit2=0;
834           noisehit=0;
835           noisehit2=0;
836           prenhit = nhit;
837           prenhit2 = nhit2;
838           processhit=0;
839           processhit2=0;
840           nhit=0;
841           nhit2=0;
842           for(int jhit=0; jhit<prenhit; jhit++){
843             m_prechipArray[jhit] = m_chipArray[jhit] ;
844             m_prechanArray[jhit] = m_chanArray[jhit] ;
845             m_preadcArray[jhit] = m_adcArray[jhit] ;
846             m_preeventArray[jhit] = m_eventArray[jhit] ;
847             m_prebcoArray[jhit] = m_bcoArray[jhit] ;
848             m_prebco_fullArray[jhit]= m_bco_fullArray[jhit];
849           }
850           for(int jhit=0; jhit<prenhit2; jhit++){
851             m_prechipArray2[jhit] = m_chipArray2[jhit] ;
852             m_prechanArray2[jhit] = m_chanArray2[jhit] ;
853             m_preadcArray2[jhit] = m_adcArray2[jhit] ;
854             m_preeventArray2[jhit] = m_eventArray2[jhit] ;
855             m_prebcoArray2[jhit] = m_bcoArray2[jhit] ;
856             m_prebco_fullArray2[jhit]= m_bco_fullArray2[jhit];
857           }
```

```
858         }
859         singlemV->Fill(adc_mV(adc),mV_weight2(adc_mV(adc)));
860   //+++++++++++++++copy new data++++++++++++++++++++++++++++++
861         if(module==8&&ampl==0&&adc>-1.&&adc<8&&chip_id>0&&chip_id<27&&chan_id>-1&&chan_id<128){
862           m_chipArray[nhit] = chip_id;
863           m_chanArray[nhit] = chan_id;
864           m_adcArray[nhit] = adc;
865           m_eventArray[nhit]= event;
866           m_bcoArray[nhit] = bco;
867           m_bco_fullArray[nhit]=bco_full;
868           nhit++;
869         }
870         if(module==6&&ampl==0&&adc>-1.&&adc<8&&chip_id>0&&chip_id<27&&chan_id>-1&&chan_id<128){
871           m_chipArray2[nhit2] = chip_id;
872           m_chanArray2[nhit2] = chan_id;
873           m_adcArray2[nhit2] = adc;
874           m_eventArray2[nhit2]= event;
875           m_bcoArray2[nhit2] = bco;
876           m_bco_fullArray2[nhit2]=bco_full;
877           nhit2++;
878         }
879       }
880       if(chan_id<0||128<=chan_id) {
881         cout<<"exceed chan_id"<<endl;
882         cout<<event<<" clk:"<<bco<<" "<<bco_full<<" :"<<chip_id<<" "<<chan_id<<" :"<<module<<" "<<adc<<(adc>0?"hit:"
                " )<<endl;;
883       }
884     cout<<"noise"<<noise<<endl;
885   };
886   void readtree::init_histo(){
887     int m=9;
888     double xbins[] = {0, 270, 302, 450, 602, 750, 902, 1050, 1202, 1234};
889     singlemV = new TH1F("singlemV","energy loss without clustering with weight",m,xbins);
890     hnhits = new TH1F("hnhits","N hits per event",7,0,7);
891     SingleAdc = new TH1F("SingleAdc","energyloss before clustering;mV",m,xbins);
892     DAdc = new TH1F("DAdc","energyloss before clustering;mV",m,xbins);
893     TAdc = new TH1F("TAdc","energyloss before clustering;mV",m,xbins);
894     SingleAdc->SetLineColor(kRed);
895     DAdc->SetLineColor(kBlue);
896     TAdc->SetLineColor(kGreen);
897     m_noiseAdc = new TH1F("m_noiseAdc", "ADC of noise data;adc", 8, 0, 8);
898     Wadc = new TH1F("Wadc", "ADC with weight;adc", 8, 0, 8);
899     NHITChip = new TH2F("NHITChip","ADC correlation of separate double hits;ch1 ADC;ch2 ADC",12,-2.,10.,12,-2.,10.);
900     chan_1hit = new TH2F("chan_1hit","channel correlation of separate double hits;ch1 channel;ch2 channel",130,0.,130.,130,0.,130.);
901     chan_1hit2 = new TH2F("chan_1hit2","channel difference of single vs chip;channel difference;chip_id",128,0.,128.,27,0,27);
902     adc_0hit = new TH2F("adc_0hit","single hit of ADC=0;chan_id;chip_id",128,0,128,26,0,26);
903     multiclschip = new TH2F("multiclschip","chip vs channel in multiple clsters event;chan_id;chip_id",128,0,128,26,0,26);
904     nclsnChip  = new TH1F("nclsnChip","Number of clusters per event;number of clusters",5,0,5);
905     NCLS6    = new TH1F("NCLS6","Number of clusters per event;number of clusters",5,0,5);
906     NCLS8    = new TH1F("NCLS8","Number of clusters per event;number of clusters",5,0,5);
907     NSTRIP6   = new TH1F("NSTRIP6","Number of strips per event;number of strips",5,0,5);
908     NSTRIP8   = new TH1F("NSTRIP8","Number of strips per event;number of strips",5,0,5);
909     NHITChipmV = new TH1F("NHITChipmV","2 single hits in same chip;mV",m,xbins);
910     SumDoublehitAdc2 = new TH1F("SumDoublehitAdc2","energy loss per cluster;mV",m,xbins);
911     SinglehitAdc2 = new TH1F("SinglehitAdc2","energy loss per cluster;mV",m,xbins);
912     SumTriplehitAdc2 = new TH1F("SumTriplehitAdc2","energy loss per cluster;mV",m,xbins);
913     Adc_2hit = new TH2F("Adc_2hit","adc correlation of 2 hits;ch1 adc;ch2 adc",12,-2.,10.,12,-2.,10.);
914     chan_2hit = new TH2F("chan_2hit","channel correlation of 2 hits;ch1 channel;ch2 channel",130,0.,130.,130,0.,130.);
915     sa_correlation = new TH2F("sa_correlation","correlation of bco and bcofull interval;bco interval;bcofull interval"
            ,1000,-500.,500.,10000,-5000,5000.);
916     SumTriplehitAdc2 = new TH1F("SumTriplehitAdc2","energy loss per cluster;mV",m,xbins);
917     bco_interval  = new TH1F("bco_interval","bco interval;bco interval",300,-150,150);
918     bcofull_interval = new TH1F("bcofull_interval","bcofull interval;bco interval",140000,-70000,70000);
919     adc_bcofull   = new TH1F("adc_bcofull","ADC (bcofull interval==-1);adc",8,0,8);
920     ADCDIFF   = new TH2F("ADCDIFF","energyloss correlation of 2 hits;channel1 mV;channel2 mV",m,xbins,m,xbins);
921     ELOSSDIFF = new TH2F("ELOSSDIFF","energy loss correlation of modules;moudle6 mV;module8 mV",m,xbins,m,xbins);
922     ELOSSDIFFSA = new TH1F("ELOSSDIFFSA","energy loss difference between modules;mV",130,0.,1300.);
923     ELOSSTRAN = new TH2F("ELOSSTRAN","energy loss correlarion between number of hits;1 hit mV;2 hits mV",m,xbins,m,xbins);
924     ELOSSTRANSA = new TH1F("ELOSSTRANSA","energy loss difference between number of hits;mV",130,0.,1300.);
925     ELOSSTRAN1 = new TH2F("ELOSSTRAN1","energy loss differense between number of hits(1);1 hit mV;2 hits mV",m,xbins,m,xbins);
926     ELOSSTRAN2 = new TH2F("ELOSSTRAN2","energy loss differense between number of hits(2);1 hit mV;2 hits mV",m,xbins,m,xbins);
927     ELOSSTRAN3 = new TH2F("ELOSSTRAN3","energy loss differense between number of hits(3);1 hit mV;2 hits mV",m,xbins,m,xbins);
928     ELOSSTRAN4 = new TH2F("ELOSSTRAN4","energy loss differense between number of hits(4);1 hit mV;2 hits mV",m,xbins,m,xbins);
929     ELOSSTRANS = new TH2F("ELOSSTRANS","energy loss differense between number of hits(hit position);1 hit mV;2 hits mV",m,xbins,m,xbins);
930     CHANDIFF = new TH2F("CHANDIFF","channel correlation of modules;moudle6 channel;module8 channel",65,-1,129,65,-1,129);
931     CHANDIFFS = new TH2F("CHANDIFFS","channel correlation of modules(hit position);moudle6 channel;module8 channel",8,0,128,8,0,128);
932     CHANDIFFSA1 = new TH1F("CHANDIFFSA1","channel difference between chips;chan (1)",130,-1,129);
933     CHANDIFFSA2 = new TH1F("CHANDIFFSA2","channel difference between chips;chan (2)",130,-1,129);
934     CHANDIFFSA3 = new TH1F("CHANDIFFSA3","channel difference between chips;chan (3)",130,-1,129);
935     CHANDIFFSA4 = new TH1F("CHANDIFFSA4","channel difference between chips;chan (4)",130,-1,129);
936     CHANDIFFSAh = new TH1F("CHANDIFFSAh","channel difference between chips(hit position);channel difference",16,-128,128);
937     CHIPDIFFSAh = new TH1F("CHIPDIFFSAh","chip difference between modules(hit position) ;chip difference",52,-26,26);
938     ELOSSDIFF1  = new TH2F("ELOSSDIFF1","energy loss correlation of modules (1);moudle6 mV;module8 mV",m,xbins,m,xbins);
939     ELOSSDIFF2  = new TH2F("ELOSSDIFF2","energy loss correlation of modules (2);moudle6 mV;module8 mV",m,xbins,m,xbins);
940     ELOSSDIFF3  = new TH2F("ELOSSDIFF3","energy loss correlation of modules (3);moudle6 mV;module8 mV",m,xbins,m,xbins);
941     ELOSSDIFF4  = new TH2F("ELOSSDIFF4","energy loss correlation of modules (4);moudle6 mV;module8 mV",m,xbins,m,xbins);
942     ELOSSCLS1_8_1  = new TH1F("ELOSSCLS1_8_1","clusterd energy loss (1);mV",m,xbins);
943     ELOSSCLS1_6_1  = new TH1F("ELOSSCLS1_6_1","clusterd energy loss (1);mV",m,xbins);
944     ELOSSCLS2_8_1  = new TH1F("ELOSSCLS2_8_1","clusterd energy loss (1);mV",m,xbins);
945     ELOSSCLS2_6_1  = new TH1F("ELOSSCLS2_6_1","clusterd energy loss (1);mV",m,xbins);
946     ELOSSCLS1_8_2  = new TH1F("ELOSSCLS1_8_2","clusterd energy loss (2);mV",m,xbins);
947     ELOSSCLS1_6_2  = new TH1F("ELOSSCLS1_6_2","clusterd energy loss (2);mV",m,xbins);
948     ELOSSCLS2_8_2  = new TH1F("ELOSSCLS2_8_2","clusterd energy loss (2);mV",m,xbins);
949     ELOSSCLS2_6_2  = new TH1F("ELOSSCLS2_6_2","clusterd energy loss (2);mV",m,xbins);
```

```
950   ELOSSCLS1_8_3  = new TH1F("ELOSSCLS1_8_3","clusterd energy loss (3);mV",m,xbins);
951   ELOSSCLS1_6_3  = new TH1F("ELOSSCLS1_6_3","clusterd energy loss (3);mV",m,xbins);
952   ELOSSCLS2_8_3  = new TH1F("ELOSSCLS2_8_3","clusterd energy loss (3);mV",m,xbins);
953   ELOSSCLS2_6_3  = new TH1F("ELOSSCLS2_6_3","clusterd energy loss (3);mV",m,xbins);
954   ELOSSCLS1_8_4  = new TH1F("ELOSSCLS1_8_4","clusterd energy loss (4);mV",m,xbins);
955   ELOSSCLS1_6_4  = new TH1F("ELOSSCLS1_6_4","clusterd energy loss (4);mV",m,xbins);
956   ELOSSCLS2_8_4  = new TH1F("ELOSSCLS2_8_4","clusterd energy loss (4);mV",m,xbins);
957   ELOSSCLS2_6_4  = new TH1F("ELOSSCLS2_6_4","clusterd energy loss (4);mV",m,xbins);
958   CHIPDIFF = new TH2F("CHIPDIFF","chip correlation of modules;moudle6 chip_id;module8 chip_id",28,-1,27,28,-1,27);
959   ELOSSCLS1_1 = new TH1F("ELOSSCLS1_1","clusterd energy loss (1);mV",m,xbins);
960   ELOSSCLS2_1 = new TH1F("ELOSSCLS2_1","clusterd energy loss (1);mV",m,xbins);
961   ELOSSCLS1_2 = new TH1F("ELOSSCLS1_2","clusterd energy loss (2);mV",m,xbins);
962   ELOSSCLS2_2 = new TH1F("ELOSSCLS2_2","clusterd energy loss (2);mV",m,xbins);
963   ELOSSCLS1_3 = new TH1F("ELOSSCLS1_3","clusterd energy loss (3);mV",m,xbins);
964   ELOSSCLS2_3 = new TH1F("ELOSSCLS2_3","clusterd energy loss (3);mV",m,xbins);
965   ELOSSCLS1_4 = new TH1F("ELOSSCLS1_4","clusterd energy loss (4);mV",m,xbins);
966   ELOSSCLS2_4 = new TH1F("ELOSSCLS2_4","clusterd energy loss (4);mV",m,xbins);
967   ELOSSCLS1_S = new TH1F("ELOSSCLS1_S","clusterd energy loss (diff=3, different hits);mV",m,xbins);
968   ELOSSCLS2_S = new TH1F("ELOSSCLS2_S","clusterd energy loss (diff=3, different hits);mV",m,xbins);
969   ELOSSCLS1__S = new TH1F("ELOSSCLS1__S","clusterd energy loss (diff=3);mV",m,xbins);
970   ELOSSCLS2__S = new TH1F("ELOSSCLS2__S","clusterd energy loss (diff=3);mV",m,xbins);
971   RECELOSS1    = new TH1F("RECELOSS1","reconstract energy loss;mV",m,xbins);
972   RECELOSS2    = new TH1F("RECELOSS2","reconstract energy loss;mV",m,xbins);
973   originalenergyloss    = new TH1F("originalenergyloss","energyloss before clustering;mV",m,xbins);
974   ELOSSCLS1_1  ->SetLineColor(kRed);
975   ELOSSCLS2_1  ->SetLineColor(kBlue);
976   ELOSSCLS1_2  ->SetLineColor(kRed);
977   ELOSSCLS2_2  ->SetLineColor(kBlue);
978   ELOSSCLS1_3  ->SetLineColor(kRed);
979   ELOSSCLS2_3  ->SetLineColor(kBlue);
980   ELOSSCLS1_4  ->SetLineColor(kRed);
981   ELOSSCLS2_4  ->SetLineColor(kBlue);
982   ELOSSCLS1_S  ->SetLineColor(kRed);
983   ELOSSCLS2_S  ->SetLineColor(kBlue);
984   ELOSSCLS1__S ->SetLineColor(kRed);
985   ELOSSCLS2__S ->SetLineColor(kBlue);
986   RECELOSS1    ->SetLineColor(kRed);
987   RECELOSS2    ->SetLineColor(kBlue);
988   NCLS6        ->SetLineColor(kRed);
989   NCLS8        ->SetLineColor(kBlue);
990   NSTRIP6      ->SetLineColor(kRed);
991   NSTRIP8      ->SetLineColor(kBlue);
992   SumDoublehitAdc2->SetLineColor(kBlue);
993   SinglehitAdc2  ->SetLineColor(kRed);
994   SumTriplehitAdc2->SetLineColor(kGreen);
995   ELOSSCLS1_8_1  ->SetLineColor(kRed);
996   ELOSSCLS1_6_1  ->SetLineColor(kRed);
997   ELOSSCLS2_8_1  ->SetLineColor(kBlue);
998   ELOSSCLS2_6_1  ->SetLineColor(kBlue);
999   ELOSSCLS1_8_2  ->SetLineColor(kRed);
1000  ELOSSCLS1_6_2  ->SetLineColor(kRed);
1001  ELOSSCLS2_8_2  ->SetLineColor(kBlue);
1002  ELOSSCLS2_6_2  ->SetLineColor(kBlue);
1003  ELOSSCLS1_8_3  ->SetLineColor(kRed);
1004  ELOSSCLS1_6_3  ->SetLineColor(kRed);
1005  ELOSSCLS2_8_3  ->SetLineColor(kBlue);
1006  ELOSSCLS2_6_3  ->SetLineColor(kBlue);
1007  ELOSSCLS1_8_4  ->SetLineColor(kRed);
1008  ELOSSCLS1_6_4  ->SetLineColor(kRed);
1009  ELOSSCLS2_8_4  ->SetLineColor(kBlue);
1010  ELOSSCLS2_6_4  ->SetLineColor(kBlue);
1011 }
1012 void readtree::end(){
1013   m_froot->cd();
1014   singlemV       ->Write();
1015   hnhits       ->Write();
1016   SingleAdc      ->Write();
1017   DAdc       ->Write();
1018   TAdc       ->Write();
1019   singlemV       ->Write();
1020   originalenergyloss->Write();
1021   m_noiseAdc ->Write();
1022   Wadc ->Write();
1023   NHITChip ->Write();
1024   chan_1hit ->Write();
1025   chan_1hit2 ->Write();
1026   adc_0hit ->Write();
1027   multiclschip ->Write();
1028   NHITChipmV ->Write();
1029   SumDoublehitAdc2 ->Write();
1030   SinglehitAdc2 ->Write();
1031   SumTriplehitAdc2 ->Write();
1032   Adc_2hit ->Write();
1033   chan_2hit ->Write();
1034   sa_correlation ->Write();
1035   bco_interval  ->Write();
1036   bcofull_interval ->Write();
1037   adc_bcofull   ->Write();
1038   nclsnChip     ->Write();
1039   NCLS6       ->Write();
1040   NCLS8       ->Write();
1041   NSTRIP6      ->Write();
1042   NSTRIP8      ->Write();
1043   ELOSSDIFF     ->Write();
```

```
    ELOSSDIFFSA   ->Write();
    ELOSSTRAN     ->Write();
    ELOSSTRANSA   ->Write();
    ELOSSTRANS    ->Write();
    CHANDIFF      ->Write();
    CHANDIFFS     ->Write();
    CHIPDIFF      ->Write();
    ADCDIFF       ->Write();
    CHANDIFFSA1   ->Write();
    CHANDIFFSA2   ->Write();
    CHANDIFFSA3   ->Write();
    CHANDIFFSA4   ->Write();
    CHANDIFFSAh   ->Write();
    CHIPDIFFSAh   ->Write();
    CHIPDIFFSAh   ->Write();
    ELOSSDIFF1    ->Write();
    ELOSSDIFF2    ->Write();
    ELOSSDIFF3    ->Write();
    ELOSSDIFF4    ->Write();
    ELOSSCLS1_8_1 ->Write();
    ELOSSCLS1_6_1 ->Write();
    ELOSSCLS2_8_1 ->Write();
    ELOSSCLS2_6_1 ->Write();
    ELOSSCLS1_8_2 ->Write();
    ELOSSCLS1_6_2 ->Write();
    ELOSSCLS2_8_2 ->Write();
    ELOSSCLS2_6_2 ->Write();
    ELOSSCLS1_8_3 ->Write();
    ELOSSCLS1_6_3 ->Write();
    ELOSSCLS2_8_3 ->Write();
    ELOSSCLS2_6_3 ->Write();
    ELOSSCLS1_8_4 ->Write();
    ELOSSCLS1_6_4 ->Write();
    ELOSSCLS2_8_4 ->Write();
    ELOSSCLS2_6_4 ->Write();
    ELOSSTRAN1    ->Write();
    ELOSSTRAN2    ->Write();
    ELOSSTRAN3    ->Write();
    ELOSSTRAN4    ->Write();
    ELOSSCLS1_1   ->Write();
    ELOSSCLS2_1   ->Write();
    ELOSSCLS1_2   ->Write();
    ELOSSCLS2_2   ->Write();
    ELOSSCLS1_3   ->Write();
    ELOSSCLS2_3   ->Write();
    ELOSSCLS1_4   ->Write();
    ELOSSCLS2_4   ->Write();
    ELOSSCLS2_S   ->Write();
    ELOSSCLS1_S   ->Write();
    ELOSSCLS2__S  ->Write();
    ELOSSCLS1__S  ->Write();
    RECELOSS1     ->Write();
    RECELOSS2     ->Write();
}
```

Listing A.2 readtree.h

```
//////////////////////////////////////////////////////////
// This class has been automatically generated on
// Tue Nov 27 16:03:31 2018 by ROOT version 5.34/36
// from TTree tree/chip info
// found on file: fphx_raw_20191029-0947_0.root
//////////////////////////////////////////////////////////
#ifndef readtree_h
#define readtree_h
#include <TROOT.h>
#include <TChain.h>
#include <TFile.h>
#include <TH2.h>
#include <TH1.h>
// Header file for the classes stored in the TTree if any.
// Fixed size dimensions of array or collections stored in the TTree if any.
class readtree {
public :
   TTree *fChain; //!pointer to the analyzed TTree or TChain
   Int_t fCurrent; //!current Tree number in a TChain
   // Declaration of leaf types
   Int_t adc;
   Int_t ampl;
   Int_t chip_id;
   Int_t fpga_id;
   Int_t module;
   Int_t chan_id;
   Int_t fem_id;
   Int_t bco;
   Int_t bco_full;
   Int_t event;
   // List of branches
   TBranch *b_adc; //!
   TBranch *b_ampl; //!
   TBranch *b_chip_id; //!
```

```
35      TBranch *b_fpga_id; //!
36      TBranch *b_module; //!
37      TBranch *b_chan_id; //!
38      TBranch *b_fem_id; //!
39      TBranch *b_bco; //!
40      TBranch *b_bco_full; //!
41      TBranch *b_event; //!
42      TFile* m_froot;
43      TH2* m_hitmap;
44      TH1F* singlemV ;
45      TH1F* hnhits ;
46      TH2F* nhitsnChip ;
47      TH1F* nclsnChip;
48      TH1F* SingleAdc;
49      TH1F* DAdc ;
50      TH1F* TAdc ;
51      TH1F*     m_noiseAdc;
52      TH1F* originalenergyloss ;
53      TH1F* Wadc ;
54      TH2F* NHITChip ;
55      TH2F* chan_1hit ;
56      TH2F* chan_1hit2 ;
57    // TH1F* chan_1hit2 ;
58      TH2F* adc_0hit ;
59      TH2F* multiclschip ;
60      TH1F* NHITChipmV ;
61      TH1F* SumDoublehitAdc2 ;
62      TH1F* SinglehitAdc2 ;
63      TH1F* SumTriplehitAdc2 ;
64      TH2F* Adc_2hit ;
65      TH2F* chan_2hit ;
66      TH2F* sa_correlation ;
67      TH1F* bco_interval   ;
68      TH1F* bcofull_interval ;
69      TH1F* adc_bcofull   ;
70      TH2F* ELOSSDIFF    ;
71      TH1F* ELOSSDIFFSA    ;
72      TH2F* ELOSSTRAN    ;
73      TH1F* ELOSSTRANSA    ;
74      TH2F* CHANDIFF    ;
75      TH2F* CHANDIFFS    ;
76      TH2F* CHIPDIFF    ;
77      TH2F* ADCDIFF    ;
78      TH1F* CHANDIFFSA1    ;
79      TH1F* CHANDIFFSA2    ;
80      TH1F* CHANDIFFSA3    ;
81      TH1F* CHANDIFFSA4    ;
82      TH1F* CHANDIFFSAh    ;
83      TH1F* CHIPDIFFSAh    ;
84      TH2F* ELOSSDIFF1    ;
85      TH2F* ELOSSDIFF2    ;
86      TH2F* ELOSSDIFF3    ;
87      TH2F* ELOSSDIFF4    ;
88      TH1F* ELOSSCLS1_8_1 ;
89      TH1F* ELOSSCLS1_6_1 ;
90      TH1F* ELOSSCLS2_8_1 ;
91      TH1F* ELOSSCLS2_6_1 ;
92      TH1F* ELOSSCLS1_8_2 ;
93      TH1F* ELOSSCLS1_6_2 ;
94      TH1F* ELOSSCLS2_8_2 ;
95      TH1F* ELOSSCLS2_6_2 ;
96      TH1F* ELOSSCLS1_8_3 ;
97      TH1F* ELOSSCLS1_6_3 ;
98      TH1F* ELOSSCLS2_8_3 ;
99      TH1F* ELOSSCLS2_6_3 ;
100     TH1F* ELOSSCLS1_8_4 ;
101     TH1F* ELOSSCLS1_6_4 ;
102     TH1F* ELOSSCLS2_8_4 ;
103     TH1F* ELOSSCLS2_6_4 ;
104     TH2F* ELOSSTRAN1    ;
105     TH2F* ELOSSTRAN2    ;
106     TH2F* ELOSSTRAN3    ;
107     TH2F* ELOSSTRAN4    ;
108     TH2F* ELOSSTRANS    ;
109     TH1F*     ELOSSCLS1_1   ;
110     TH1F*     ELOSSCLS2_1  ;
111     TH1F*     ELOSSCLS1_2   ;
112     TH1F*     ELOSSCLS2_2   ;
113     TH1F*     ELOSSCLS1_3   ;
114     TH1F*     ELOSSCLS2_3   ;
115     TH1F*     ELOSSCLS1_4   ;
116     TH1F*     ELOSSCLS2_4   ;
117     TH1F*     ELOSSCLS1_S   ;
118     TH1F*     ELOSSCLS2_S   ;
119     TH1F*     ELOSSCLS1__S   ;
120     TH1F*     ELOSSCLS2__S   ;
121     TH1F*     NCLS6      ;
122     TH1F*     NCLS8      ;
123     TH1F*     NSTRIP6     ;
124     TH1F*     NSTRIP8     ;
125     TH1F*     RECELOSS1    ;
126     TH1F*     RECELOSS2    ;
127     TH1F*     RECELOSS3    ;
128     TH1F*     ReNhit      ;
```

```
129     int m_jentry;
130     int m_prebco, m_prebcofull, m_prenhit;
131     readtree(TTree *tree=0, const char* outfile="histo.root");
132     virtual ~readtree();
133     virtual Int_t Cut(Long64_t entry);
134     virtual Int_t GetEntry(Long64_t entry);
135     virtual Long64_t LoadTree(Long64_t entry);
136     virtual void Init(TTree *tree);
137     virtual void Loop(const int entry=-1);
138     virtual Bool_t Notify();
139     virtual void Show(Long64_t entry = -1);
140     void initHist();
141     void init_histo();
142     void process_hits(int nhit, int* chipArray, int* chanArray, int* adcArray, int* eventArray, int* bcoArray, int* bco_fullArray, double
            * summV_Ret, int* Nhits_Ret, int* chan_Ret, int* chip_Ret, int ncls_Ret, int(*adc_Ret)[10]);
143     void end();
144 };
145 #endif
146 #ifdef readtree_cxx
147 readtree::readtree(TTree *tree, const char *outfile) : fChain(0)
148 {
149 // if parameter tree is not specified (or zero), connect the file
150 // used to generate this class and read the Tree.
151     if (tree == 0) {
152         TFile *f = (TFile*)gROOT->GetListOfFiles()->FindObject("fphx_raw_20200114-all.root");
153         if (!f || !f->IsOpen()) {
154             f = new TFile("fphx_raw_20200114-all_0.root");
155         }
156         f->GetObject("tree",tree);
157     }
158     Init(tree);
159     m_froot = TFile::Open(outfile, "recreate");
160     initHist();
161     init_histo();
162     m_jentry=0;
163     m_prebco=-1;
164     m_prebcofull=-1;
165     m_prenhit=-1;
166 }
167 readtree::~readtree()
168 {
169     if (!fChain) return;
170     delete fChain->GetCurrentFile();
171 }
172 Int_t readtree::GetEntry(Long64_t entry)
173 {
174 // Read contents of entry.
175     if (!fChain) return 0;
176     return fChain->GetEntry(entry);
177 }
178 Long64_t readtree::LoadTree(Long64_t entry)
179 {
180 // Set the environment to read one entry
181     if (!fChain) return -5;
182     Long64_t centry = fChain->LoadTree(entry);
183     if (centry < 0) return centry;
184     if (fChain->GetTreeNumber() != fCurrent) {
185         fCurrent = fChain->GetTreeNumber();
186         Notify();
187     }
188     return centry;
189 }
190 void readtree::Init(TTree *tree)
191 {
192     // The Init() function is called when the selector needs to initialize
193     // a new tree or chain. Typically here the branch addresses and branch
194     // pointers of the tree will be set.
195     // It is normally not necessary to make changes to the generated
196     // code, but the routine can be extended by the user if needed.
197     // Init() will be called many times when running on PROOF
198     // (once per file to be processed).
199     // Set branch addresses and branch pointers
200     if (!tree) return;
201     fChain = tree;
202     fCurrent = -1;
203     fChain->SetMakeClass(1);
204     fChain->SetBranchAddress("adc", &adc, &b_adc);
205     fChain->SetBranchAddress("ampl", &ampl, &b_ampl);
206     fChain->SetBranchAddress("chip_id", &chip_id, &b_chip_id);
207     fChain->SetBranchAddress("fpga_id", &fpga_id, &b_fpga_id);
208     fChain->SetBranchAddress("module", &module, &b_module);
209     fChain->SetBranchAddress("chan_id", &chan_id, &b_chan_id);
210     fChain->SetBranchAddress("fem_id", &fem_id, &b_fem_id);
211     fChain->SetBranchAddress("bco", &bco, &b_bco);
212     fChain->SetBranchAddress("bco_full", &bco_full, &b_bco_full);
213     fChain->SetBranchAddress("event", &event, &b_event);
214     Notify();
215 }
216 Bool_t readtree::Notify()
217 {
218     // The Notify() function is called when a new file is opened. This
219     // can be either for a new TTree in a TChain or when when a new TTree
220     // is started when using PROOF. It is normally not necessary to make changes
221     // to the generated code, but the routine can be extended by the
```

```
222     // user if needed. The return value is currently not used.
223     return kTRUE;
224 }
225 void readtree::Show(Long64_t entry)
226 {
227 // Print contents of entry.
228 // If entry is not specified, print current entry
229     if (!fChain) return;
230     fChain->Show(entry);
231 }
232 Int_t readtree::Cut(Long64_t entry)
233 {
234 // This function may be called from Loop.
235 // returns 1 if entry is accepted.
236 // returns -1 otherwise.
237     return 1;
238 }
239 #endif // #ifdef readtree_cxx
```

Listing A.3 run_readtree.C

```
1  void run_readtree(const char *infile= "fphx_raw_20200215-large.root")
2  {
3  // gROOT->ProcessLine(".L readtree.c");
4  // const char *outfile="histo.root";
5    TString s_outfile(infile);
6    s_outfile.Prepend("histo");
7    TFile *f = TFile::Open(infile);
8    TTree *tree = (TTree*)f->Get("tree");
9    readtree t(tree, s_outfile.Data());
10   t.Loop();
11   t.end();
12 }
```

Listing A.4 plot_readtree2.C

```
1   {
2     int nhit_1[9];
3     int nhit_2[9];
4     for(int ii=1;ii<10;ii++){
5       nhit_1[ii]=0;
6       nhit_2[ii]=0;
7     }
8     TFile *rootfile = TFile::Open("histofphx_raw_20200215-large.root");
9     TCanvas *c1 = new TCanvas("c1", "energy loss", 1000, 280);
10    c1->Divide(4,1);
11    TCanvas *c2 = new TCanvas("c2", "correlation of double hits", 800, 590);
12    c2->Divide(3,2);
13    TCanvas *c3 = new TCanvas("c3", "module differense", 1400, 1300);
14    c3->Divide(4,4);
15    TCanvas *c4 = new TCanvas("c4", "cosmic", 600, 590);
16    c4->Divide(3,2);
17    TCanvas *c5 = new TCanvas("c5", "bco and bco_full intervals", 800, 300);
18    c5->Divide(3,1);
19    TCanvas *c6 = new TCanvas("c6", "energy loss per strip number", 1000, 600);
20    c6->Divide(4,2);
21    TCanvas *c7 = new TCanvas("c7", "module differense", 800, 590);
22    c7->Divide(3,2);
23    TCanvas *c9 = new TCanvas("c9", "channel differense = 3", 1200, 500);
24    c9->Divide(5,2);
25    TCanvas *c10 = new TCanvas("c10", "energy loss", 500, 500);
26
27    c1->cd(1);//before event selection
28    Wadc->Draw("hist");
29    c1->cd(2);
30    singlemV->Draw("hist");//before event selection
31    c1->cd(3);//before cluster adc
32    //SumAdc2->Draw("hist");
33    DAdc->Draw("hist");
34    SingleAdc->Draw("histsames");
35    TAdc->Draw("histsames");
36    c1->cd(4);//after cluster mV
37    SinglehitAdc2->Draw("hist");
38    SumDoublehitAdc2->Draw("histsames");
39    SumTriplehitAdc2->Draw("histsames");
40    c2->cd(1);
41    Adc_2hit->Draw("colz");
42    c2->cd(2);
43    chan_2hit->Draw("colz");
44    c2->cd(4);
45    NHITChip->Draw("colz");
46    c2->cd(5);
47    chan_1hit->Draw("colz");
48    c2->cd(6);
49    chan_1hit2->Draw();
50    c7->cd(1);//module differense
51    CHIPDIFF->Draw("colz");
52    c7->cd(2);
```

```
53  ELOSSDIFF->Draw("colz");
54  c7->cd(3);
55  ELOSSTRAN->Draw("colz");
56  c7->cd(4);
57  NCLS6->Draw("");
58  NCLS8->Draw("sames");
59  c7->cd(5);
60  NSTRIP6->Draw("");
61  NSTRIP8->Draw("sames");
62  c3->cd(1);
63  CHANDIFFSA1->Draw("hist");
64  c3->cd(2);
65  CHANDIFFSA2->Draw("hist");
66  c3->cd(3);
67  CHANDIFFSA3->Draw("hist");
68  c3->cd(4);
69  CHANDIFFSA4->Draw("hist");
70  c3->cd(5);
71  ELOSSDIFF1->Draw("colz");
72  c3->cd(6);
73  ELOSSDIFF2->Draw("colz");
74  c3->cd(7);
75  ELOSSDIFF3->Draw("colz");
76  c3->cd(8);
77  ELOSSDIFF4->Draw("colz");
78  c6->cd(1);
79  ELOSSTRAN1->Draw("colz");
80  c6->cd(2);
81  ELOSSTRAN2->Draw("colz");
82  c6->cd(3);
83  ELOSSTRAN3->Draw("colz");
84  c6->cd(4);
85  ELOSSTRAN4->Draw("colz");
86  c6->cd(5);
87  ELOSSCLS1_1->Draw("hist");
88  ELOSSCLS2_1->Draw("histsames");
89  c6->cd(6);
90  ELOSSCLS1_2->Draw("hist");
91  ELOSSCLS2_2->Draw("histsames");
92  c6->cd(7);
93  ELOSSCLS1_3->Draw("hist");
94  ELOSSCLS2_3->Draw("histsames");
95  c6->cd(8);
96  ELOSSCLS1_4->Draw("hist");
97  ELOSSCLS2_4->Draw("histsames");
98  c3->cd(9);
99  ELOSSCLS1_6_1->Draw("hist");
100 ELOSSCLS2_6_1->Draw("histsames");
101 c3->cd(10);
102 ELOSSCLS1_6_2->Draw("hist");
103 ELOSSCLS2_6_2->Draw("histsames");
104 c3->cd(11);
105 ELOSSCLS1_6_3->Draw("hist");
106 ELOSSCLS2_6_3->Draw("histsames");
107 c3->cd(12);
108 ELOSSCLS1_6_4->Draw("hist");
109 ELOSSCLS2_6_4->Draw("histsames");
110 c3->cd(13);
111 ELOSSCLS1_8_1->Draw("hist");
112 ELOSSCLS2_8_1->Draw("histsames");
113 c3->cd(14);
114 ELOSSCLS1_8_2->Draw("hist");
115 ELOSSCLS2_8_2->Draw("histsames");
116 c3->cd(15);
117 ELOSSCLS1_8_3->Draw("hist");
118 ELOSSCLS2_8_3->Draw("histsames");
119 c3->cd(16);
120 ELOSSCLS1_8_4->Draw("hist");
121 ELOSSCLS2_8_4->Draw("histsames");
122 c10->cd();
123 ReNhit->Draw("hist");
124 c9->cd(1);
125 ELOSSTRANS->Draw("colz");
126 c9->cd(2);
127 ELOSSCLS1_S->Draw("hist");
128 ELOSSCLS2_S->Draw("histsames");
129 c9->cd(3);
130 ELOSSCLS1__S->Draw("hist");
131 ELOSSCLS2__S->Draw("histsames");
132 c9->cd(4);
133 CHIPDIFFSAh->Draw("hist");
134 CHIPDIFFSAh->Fit("gaus","","",-4,11);
135 c9->cd(5);
136 CHANDIFFSAh->Draw("hist");
137 CHANDIFFSAh->Fit("gaus");
138 c9->cd(6);
139 ADCDIFF->Draw("colz");
140 c9->cd(7);
141 RECELOSS1->Draw("hist");
142 RECELOSS2->Draw("histsames");
143 RECELOSS3->Draw("histsames");
144 c9->cd(8);
145 CHANDIFFS->Draw("colz");
146 c9->cd(9);
```

```
147     ReNhit->Draw("hist");
148     gStyle->SetOptStat(1000000011);
149     for(int ii=1;ii<10;ii++){
150       nhit_1[ii]= ELOSSCLS1__S->GetBinContent(ii);
151       nhit_2[ii]= ELOSSCLS2__S->GetBinContent(ii);
152     }
153     int sum1=0;
154     int sum2=0;
155     for(int ii=1;ii<10;ii++){
156       cout<<nhit_1[ii]<<",␣";
157       sum1+=nhit_1[ii];
158     }
159     cout<<"␣sum="<<sum1<<endl;
160     for(int ii=1;ii<10;ii++){
161       cout<<sqrt(nhit_1[ii])<<",␣";
162     }
163     cout<<"␣"<<endl;
164     for(int ii=1;ii<10;ii++){
165       cout<<nhit_2[ii]<<",␣";
166       sum2+=nhit_2[ii];
167     }
168     cout<<"␣sum="<<sum2<<endl;
169     for(int ii=1;ii<10;ii++){
170       cout<<sqrt(nhit_2[ii])<<",␣";
171     }
172     cout<<"␣"<<endl;
173     c4->cd(1);//number of hits per event
174     hnhits->Draw();
175     c4->cd(2);//number of clusters per event
176     nclsnChip->Draw();
177     c4->cd(3);//multiple cluster hit chip
178     multiclschip->Draw("colz");
179     c4->cd(4);
180     NHITChipmV->Draw("hist");
181     c4->cd(5);
182     adc_0hit->Draw("colz");
183     c4->cd(6);
184     m_noiseAdc->Draw("hist");
185     c5->cd(1);//bco andbco interval
186     bco_interval->Draw();
187     c5->cd(2);
188     bcofull_interval->Draw();
189     c5->cd(3);
190     adc_bcofull->Draw("hist");
191     gStyle->SetOptStat(1000000010);
192     c1->Modified();
193     c2->Modified();
194     c3->Modified();
195     c4->Modified();
196     c5->Modified();
197     c6->Modified();
198     c7->Modified();
199     c9->Modified();
200   }
```

付録 B

MC を用いた宇宙線測定のシミュレーション

B.1　モンテカルロ（MC）シミュレーション

モンテカルロ (Monte Carlo, MC) シミュレーションは乱数を利用する確率的シミュレーションの総称である。本プログラムでは、宇宙線のシンチレーションカウンター上の入射座標と極座標、エネルギー損失のランダウ分布を乱数で発生させた。

B.1.1　シミュレーションの流れ

本シミュレーションでは、まず宇宙線測定で得られるイベントデータを MC で作成し、その後宇宙線測定と同じ解析プログラムを実行した。

測定で得られるデータの作成

1. 宇宙線がシリコンセンサーを通過
   1) 宇宙線 μ 粒子を MC で発生させる。
   2) 上下シンチレータおよびシリコンセンサーを通過させる。
2. シリコンセンサー中でエネルギー損失
   1) 宇宙線 μ 粒子が通過したシリコンセンサー上面の座標からチップ番号とチャンネル番号を取得する。
   2) シリコンセンサー中の通過ストリップ数や距離、通過チップ番号やチャンネル番号を取得する。
   3) ストリップごとにエネルギー損失を計算する。
3. 読み出し回路中で損失エネルギーが ADC に変換
   1) 損失エネルギー [eV] を電荷 [C] に変換する。

2) ゲイン (増幅係数) を用い、電荷 [C] を電圧 [mV] に変換する。

3) 電圧値 [mV] にオフセット [mV] を足す。

4) 損失エネルギーを ADC で 8 つのビンに分ける。

測定と同じ解析プログラムを実行

1) ヒットストリップをクラスター化しまとめる。

2) クラスターの ADC 分布を測定する。

B.2 シミュレーションプログラムの実行方法

宇宙線解析同様、ROOT（ルート）を用いて行った。付録 simulation.c ファイルを所定の箇所にいれ実行すれば測定可能。ゲイン値、オフセットを変更する際は l.408 と l.409 の gain と offset の値を変更、測定時間を変更する際は l.595 を変更し実行すればよい。

B.3 シミュレーションプログラムのコード

Listing B.1 simulation.c

```
1
2  // clustered hits histograms
3  // This is one of the files to get clustered hits hisotgram.
4  // You can draw dac amplitude(mV) histograms, and N hits per event hisrogram on a Canvas.
5  //
6  ////////////////////////////////////////
7  #include <TStyle.h>
8  #include <TCanvas.h>
9  #include <TH1.h>
10 #include <TH2.h>
11 #include "TNtuple.h"
12 #include "TFile.h"
13 #include <TMath.h>
14 #include <TRandom.h>
15 #include <TVector3.h>
16 #include <iostream>
17 #include <iomanip>
18 #include <fstream>
19 #include <cstdio>
20 #include "math.h"
21 #include <algorithm>
22 using namespace std;
23 double energy(double x){
24     double Z = 14.;
25     double A = 28.0855;
26     double re = 2.817*TMath::Power(10.,-13.); //cm
27     double density = 2.33;
28     double beta = 0.995428; //+-0.00379 //g cm-3
29     double Na = 6.02214*TMath::Power(10.,23.); //mol-1
30     double mec2 = 0.511; //MeVc^2
31     double length  = x;
32     double gusai = 0.307075/2.*(Z/A)*(x*0.0001*density/(beta*beta));
33     double M = 105.658;
34     double Wmax = (2.*mec2*(beta*beta/(1.-beta*beta)))/(1.+2.*(mec2/M)*TMath::Power(1.+(beta*beta/(1.-beta*beta)),1./2.)+(
           mec2/M*(mec2/M)));
35     double k2 = gusai/Wmax;
36     double gamma2 = 1./(1.-beta*beta);
37     double sigma = TMath::Power(gusai*gusai*(1.-beta*beta)/(k2*2.),1./2.);
38     double deltamp = gusai*(TMath::Log(2.*mec2*beta*beta*gamma2/0.000173)+TMath::Log(gusai/0.000173)+0.2-beta*beta-0.14);
39     double dEdx = gRandom->Landau(deltamp/(0.0001*x*density),2.*gusai);
40     return dEdx;
41 }
42 double energyloss(double gain, double x){
43     double mV   = energy(x)*x*1000000*0.0001*1.6*gain*2.33*0.0001/3.67; //energy loss per strip after channel serection
```

```
44      return mV;
45  }
46  double mV_adc(int tmp_event, int tmp_chip, int Ns, double *x,int *chan, double *SmV, int* Event_Ret, int* Chip_Ret,double *x_Ret,
          int *Schan_Ret,double *SmV_Ret,int* Adc_Ret){
47      float   mV[9] = {270., 300., 450., 602., 750., 902., 1050., 1202., 1234.};
48      int    Nhit=0;
49      for(int i=0;i<Ns;i++){
50        for(int ii=0;ii<8;ii++){
51          if( mV[ii]<=SmV[i] && SmV[i]<mV[ii+1]){
52            Chip_Ret[Nhit]=tmp_chip;
53            Event_Ret[Nhit]=tmp_event;
54            SmV_Ret[Nhit]=SmV[i];
55            Adc_Ret[Nhit]=ii;
56            x_Ret[Nhit]=x[i];
57            Schan_Ret[Nhit]=chan[i];
58            Nhit++;
59          }
60        }
61      }
62      return Nhit;//*Adc;
63  }
64  double adc_mV(int hitAdc){
65    double Dac[9] = {15., 23., 60., 98., 135., 173., 210., 248., 256.};
66    double mV = ((Dac[hitAdc+1]*4.+210.)+(Dac[hitAdc]*4.+210.))/2.-2.;
67    return mV;
68  }
69  double mV_adc2(double SmV){
70    float mV[9] = {270, 300, 450, 602, 750, 902, 1050, 1202, 1234};
71    int Adc;
72    for(int ii=0;ii<8;ii++){
73      if( mV[ii]<=SmV && SmV<mV[ii+1]){
74        Adc=ii;
75      }
76    }
77    return Adc;//*Adc;
78  }
79  double weight(int hitAdc){
80    float Dac[9] = {15, 23, 60, 98, 135, 173, 210, 248, 256};
81    float weight[8];
82    float w=1;
83    float tmpw = Dac[1]-Dac[0];
84    for(int i=0; i<8; i++){
85      weight[i] = (Dac[i+1]-Dac[i])/tmpw;
86    }
87    w = 1./weight[hitAdc];
88    return w;
89  }
90  double mV_weight(int hitAdc){
91    float mV[9] = {270, 302, 450, 602, 750, 902, 1050, 1202, 1234};
92    float weight[8];
93    float w=1;
94    float tmpw = mV[1]-mV[0];
95    for(int i=0; i<8; i++){
96      weight[i] = (mV[i+1]-mV[i])/tmpw;
97    }
98    w = 1./weight[hitAdc];
99    return w;
100 }
101 double mV_weight2(double hitmV){
102   float mV[9] = {270., 302., 450., 602., 750., 902., 1050., 1202., 1234.};
103   float weight[8];
104   float w=1;
105   float tmpw = mV[1]-mV[0];
106   for(int i=0; i<8; i++){
107     weight[i] = (mV[i+1]-mV[i])/tmpw;
108   }
109   for(int i=0;i<8;i++){
110     if(hitmV>=mV[i]&&hitmV<mV[i+1]){
111       //tmp = i;
112       w = 1./weight[i];
113     }
114   }
115   return w;
116 }
117 class Mu
118 {
119 private:
120   int  m_Nevent;
121   double m_x_silicon;  //hit point on top plane
122   double m_y_silicon;
123   double m_z_silicon;
124   double m_x_silicon_f; //hit point on each plane
125   double m_y_silicon_f;
126   double m_z_silicon_f;
127   double m_x_scinti1;  //hit point on top scinti
128   double m_y_scinti1;
129   double m_z_scinti1;
130   double m_x_scinti2;  //hit point on bottom scinti
131   double m_y_scinti2;
132   double m_z_scinti2;
133   double m_theta;
134   double m_phi;
135   double m_length;   //mu length in silicon
136   double m_yChip[28];
```

```
137     double m_xChip[2];
138     double m_dx_i;
139     int m_Chip;
140     int m_Chan;
141     TVector3 m_v_si;
142   public:
143     Mu(int nevent)
144     {
145       m_Nevent = nevent;
146       generateChip();
147     }
148     virtual ~Mu(){}
149     int   GetNevent()  {return m_Nevent;}
150     double GetX()    {return m_x_silicon;}
151     double GetY()    {return m_y_silicon;}
152     double GetZ()    {return m_z_silicon;}
153     double GetXf()    {return m_x_silicon_f;}
154     double GetYf()    {return m_y_silicon_f;}
155     double GetZf()    {return m_z_silicon_f;}
156     double GetXsc1()    {return m_x_scinti1;}
157     double GetYsc1()    {return m_y_scinti1;}
158     double GetZsc1()    {return m_z_scinti1;}
159     double GetXsc2()    {return m_x_scinti2;}
160     double GetYsc2()    {return m_y_scinti2;}
161     double GetZsc2()    {return m_z_scinti2;}
162     double GetPhi()   {return m_phi;}
163     double GetTheta()  {return m_theta;}
164     double GetLength()  {return m_length;}
165     double* GetxChipArray() {return m_xChip;}
166     double* GetyChipArray() {return m_yChip;}
167     double GetxChip(int i) {return m_xChip[i];}
168     double GetyChip(int i) {return m_yChip[i];}
169     int   GetChip() {return m_Chip;}
170     int   GetChan() {return m_Chan;}
171     double GetDX() {return m_dx_i;}
172     void generateChip(){
173       m_yChip[0] = -114000.;
174       m_yChip[14] = -114000.;
175       for(int i=1;i<6;i++){
176         m_yChip[i] = m_yChip[0]+i*20.*1000.;
177         m_yChip[i+14] = m_yChip[14]+i*20.*1000.;
178       }
179       for(int i=1;i<9;i++){
180         m_yChip[i+5+14] = m_yChip[19]+i*16.*1000.;
181         m_yChip[i+5] = m_yChip[5]+i*16.*1000.;
182       }
183       m_xChip[0] = -9.984*1000.;
184       m_xChip[1] = 9.984*1000.;
185     };
186     void checkchipchan(){
187       for(int i=0;i<13;i++){
188         if(GetyChip(i)<=GetY()&&GetyChip(i+1)>GetY()){
189           if(GetX()<0.){
190             m_Chip = 13+i+1;
191             m_dx_i  = -fmod(GetX(),78.);
192             int   tmp1  = 1;
193             if(m_dx_i==0.)tmp1= 0;
194             m_Chan = 128-(-GetX()/78+tmp1)+1; //channel number
195           }
196           if(GetX()>=0.){
197             m_Chip = i+1;
198             m_dx_i  = fmod(GetX(),78.);
199             int   tmp1  = 1;
200             if(m_dx_i==0.)tmp1= 0;
201             m_Chan = 128-(GetX()/78+tmp1)+1; //channel number
202           }
203         }
204       }
205     };
206     int planepoint(double X, double Y,double Z, int n_vec){
207       int hit = 0;
208       TVector3 v_sc1(GetXsc1(),GetYsc1(),GetZsc1());
209       TVector3 v_sc2(GetXsc2(),GetYsc2(),GetZsc2());
210       TVector3 v_mu = v_sc2 - v_sc1;
211       TVector3 v_ex(1.,0.,0.);
212       TVector3 v_ey(0.,1.,0.);
213       TVector3 v_ez(0.,0.,1.);
214       TVector3 v_n = v_ex.Cross(v_ey); //gaiseki
215       if(n_vec==1)v_n  = v_ex.Cross(v_ey); //gaiseki
216       if(n_vec==2)v_n  = v_ey.Cross(v_ez); //gaiseki
217       if(n_vec==3)v_n  = v_ez.Cross(v_ex); //gaiseki
218       TVector3 v_f(X, Y, Z);//one of the points on plane
219       double d      = -v_n*v_f;
220       double t      = -(v_sc1*v_n+d)/(v_mu*v_n);
221       m_x_silicon_f  = v_sc1.X()+t*v_mu.X(); //after rotateing
222       m_y_silicon_f  = v_sc1.Y()+t*v_mu.Y();
223       m_z_silicon_f  = v_sc1.Z()+t*v_mu.Z();
224       double  xstart;
225       int chip = GetChip();
226       if(GetChip()<14)xstart = 0.;
227       if(GetChip()>13)xstart = -9984.;
228       if(m_x_silicon_f>=xstart&&m_x_silicon_f<=xstart+9984.)hit++;
229       if(chip<14&&m_y_silicon_f>=GetyChip(chip-1)&&m_y_silicon_f<=GetyChip(chip))hit++;
230       if(chip>13&&m_y_silicon_f>=GetyChip(chip)&&m_y_silicon_f<=GetyChip(chip+1))hit++;
```

```
231       if(m_z_silicon_f>=-320.&&m_z_silicon_f<=0.)hit++;
232       return hit;
233     };
234     void generateMu(){
235       m_x_silicon = 0.;
236       m_y_silicon = 0.;
237       m_z_silicon = 0.;
238       m_x_scinti1 = 0.;
239       m_y_scinti1 = 0.;
240       m_z_scinti1 = 0.;
241       m_x_scinti2 = 0.;
242       m_y_scinti2 = 0.;
243       m_z_scinti2 = 0.;
244       m_theta   = 0.;
245       m_phi     = 0.;
246       m_Chip    = 0;
247       m_Chan    = 0;
248       int nMu=0;
249       int hitsc2=0;
250       double  x_scinti1;
251       double  y_scinti1;
252       double theta ;
253       double phi ;
254       double  z_scinti1;
255       double z_scinti2;
256       double tmpphi;
257       double x_scinti2 ;
258       double y_scinti2 ;
259       double x_silicon ;
260       double y_silicon ;
261       double z_silicon = 0.;
262       double x_silicon_f = 0.;
263       double y_silicon_f = 0.;
264       double z_silicon_f = 0.;
265       double PI=3.14159265358979323846;
266       while(nMu<1){
267         x_scinti1 = gRandom->Rndm()*24000.-12000.;
268         y_scinti1 = gRandom->Rndm()*231000.-165500.;
269         theta = TMath::ACos(gRandom->Rndm());
270         tmpphi = gRandom->Rndm()*PI*2.;
271         z_scinti1 = 33000.;
272         z_scinti2 = -29000.;
273         m_phi = tmpphi;
274         m_theta = theta;
275         x_scinti2 = x_scinti1-(-z_scinti2+z_scinti1)*TMath::Tan(theta)*TMath::Sin(tmpphi);
276         y_scinti2 = y_scinti1-(-z_scinti2+z_scinti1)*TMath::Tan(theta)*TMath::Cos(tmpphi);
277         if(x_scinti2>=-12000.&&x_scinti2<=12000.&&y_scinti2>=-165500.&&y_scinti2<=165500.){
278           hitsc2++;
279           m_x_scinti1 = x_scinti1;
280           m_y_scinti1 = y_scinti1;
281           m_z_scinti1 = z_scinti1;
282           m_x_scinti2 = x_scinti2;
283           m_y_scinti2 = y_scinti2;
284           m_z_scinti2 = z_scinti2;
285           x_silicon = x_scinti1-(z_scinti1)*TMath::Tan(theta)*TMath::Sin(tmpphi);
286           y_silicon = y_scinti1-(z_scinti1)*TMath::Tan(theta)*TMath::Cos(tmpphi);
287           z_silicon  =  0.;
288           if(x_silicon>=-9984.&&x_silicon<=9984.&&y_silicon>=-114000.&&y_silicon<=114000.){
289             int n_vec = 1; //x-y plane
290             int Hit_f1;//top plane
291             int Hit_f;//top plane
292             nMu++;
293             m_x_silicon  = x_silicon;
294             m_y_silicon  = y_silicon;
295             m_z_silicon  = z_silicon;
296             checkchipchan();
297             Hit_f1=0;
298             Hit_f=0;
299             n_vec = 1; //x-y plane
300             Hit_f1 = planepoint(0., 0., -320.,n_vec);//bottom plane
301             if(Hit_f1==3){
302               x_silicon_f  = GetXf();
303               y_silicon_f  = GetYf();
304               z_silicon_f  = GetZf();
305             }
306             Hit_f=0;
307             n_vec = 2; //y-z plane
308             Hit_f = planepoint(0., 0., 0.,n_vec);
309             if(Hit_f1!=3&&Hit_f==3){
310               x_silicon_f  = GetXf();
311               y_silicon_f  = GetYf();
312               z_silicon_f  = GetZf();
313             }
314             Hit_f=0;
315             n_vec = 3; //z-x plane
316             if(GetChip()<14)Hit_f = planepoint(0., GetyChip(GetChip()-1), 0.,n_vec);
317             if(GetChip()>13)Hit_f = planepoint(0., GetyChip(GetChip()+1), 0.,n_vec);
318             if(Hit_f1!=3&&Hit_f==3){
319               x_silicon_f  = GetXf();
320               y_silicon_f  = GetYf();
321               z_silicon_f  = GetZf();
322             }
323             Hit_f=0;
324             n_vec = 3; //z-x plane
```

```
325             Hit_f = planepoint(0., GetyChip(GetChip()), 0.,n_vec);
326             if(Hit_f1!=3&&Hit_f==3){
327               x_silicon_f  = GetXf();
328               y_silicon_f  = GetYf();
329               z_silicon_f  = GetZf();
330             }
331             Hit_f=0;
332             n_vec = 2; //y-z plane
333             Hit_f = planepoint(-9984.,0., 0.,n_vec);
334             if(Hit_f1!=3&&Hit_f==3){
335               x_silicon_f  = GetXf();
336               y_silicon_f  = GetYf();
337               z_silicon_f  = GetZf();
338             }
339             Hit_f=0;
340             n_vec = 2; //y-z plane
341             Hit_f = planepoint(9984.,0., 0.,n_vec);
342             if(Hit_f1!=3&&Hit_f==3){
343               x_silicon_f  = GetXf();
344               y_silicon_f  = GetYf();
345               z_silicon_f  = GetZf();
346             }
347             Hit_f=0;
348             m_length = TMath::Sqrt( TMath::Power(x_silicon_f-m_x_silicon,2.)
349                                     +TMath::Power(y_silicon_f-m_y_silicon,2.)
350                                     +TMath::Power(z_silicon_f-m_z_silicon,2.));
351           }
352         }
353       }
354     };
355   };
356   class GlauberCalculation
357   {
358   private:
359     Mu *m_Si;
360     double m_x[128];
361     int   m_nhit;
362     int   m_nhitorg;
363     int   m_adc[128];
364     int   m_chip[128];
365     int   m_chan[128];
366     int   m_event[128];
367     double m_Edep[128];
368     double m_SmV[128];
369   public:
370     GlauberCalculation(){}
371     virtual ~GlauberCalculation(){}
372     void operator()(Mu* Si1){
373       calculate(Si1);
374     }
375     void calculate(Mu* Si1){
376       m_nhit = 0;
377       m_Si = Si1;
378       int tmp_event = m_Si->GetNevent();
379       for(int i=0;i<128;i++){
380         m_event [i] = 0;
381         m_adc  [i] = 0;
382         m_chip [i] = 0;
383         m_chan [i] = 0;
384         m_Edep [i] = 0;
385         m_SmV  [i] = 0;
386       }
387       double xSi = m_Si->GetX();
388       double ySi = m_Si->GetY();
389       double zSi = m_Si->GetZ();
390       double theta = m_Si->GetTheta();
391       double phi  = m_Si->GetPhi();
392       double* yChip = m_Si->GetetyChipArray();
393       double* xChip = m_Si->GetetxChipArray();
394   // check chip number
395       int tmp_chip = m_Si->GetChip();
396       int tmp_chan = m_Si->GetChan();
397       double dx_i  = m_Si->GetDX();
398       int tmpchan=0;
399       if(phi<=TMath::Pi()){
400         tmpchan = -1;
401       }
402       if(phi>TMath::Pi()){
403         tmpchan = 1;
404       }
405       double length  = m_Si->GetLength();
406       double new78  = 78./(TMath::Abs(TMath::Sin(phi)*TMath::Sin(theta)));
407       double totaldEdx = energy(length);
408       double gain   = 88.;
409       double offset  = 198.;
410       int tmp2   = 0;
411       double  tmp3   = 0.;
412       double tmp5   = 0.;
413       int Nhit   = 0;
414       int Schan[1664];
415       int Adc[1664];
416       double x[1664];
417       int Ns = 0;
418       for(int i=0;i<1664;i++){
```

```
419         Schan[i]=-1;
420         Adc[i]=-1;
421         x[i]=-1.;
422       }
423   // ***********length per strip*********************
424       if(78.-dx_i>0.){
425         if(tmpchan==-1&&tmp_chip<14)  x[0]  = dx_i/(TMath::Abs(TMath::Sin(phi))*TMath::Sin(theta));
426         if(tmpchan==-1&&tmp_chip>13)  x[0]  = (78.-dx_i)/(TMath::Abs(TMath::Sin(phi))*TMath::Sin(theta));
427         if(tmpchan==1&&tmp_chip>13)  x[0]  = dx_i/(TMath::Abs(TMath::Sin(phi))*TMath::Sin(theta));
428         if(tmpchan==1&&tmp_chip<14)  x[0]  = (78.-dx_i)/(TMath::Abs(TMath::Sin(phi))*TMath::Sin(theta));
429         if(length<=x[0]){
430           x[0]=length;
431           Ns=1;
432         }
433         else if(length>x[0]){
434           tmp3 = length-x[0];
435           tmp2 = tmp3/new78;
436           tmp5 = fmod(tmp3,new78);
437           if(tmp5!=0.){
438             x[tmp2+1] = tmp5;
439             Ns =tmp2+2;
440           }
441           else if(tmp5==0.)Ns = tmp2+1;
442           if(tmp2>0.){
443             for(int iiii=0;iiii<tmp2;iiii++) {
444               x[iiii+1]=(tmp3-tmp5)/tmp2;
445             }
446           }
447         }
448       }
449       else if(78.-dx_i==0.){
450         tmp2 = length/new78;
451         tmp5 = fmod(length,new78);
452         if(tmp5!=0.){
453           x[tmp2] = tmp5;
454           Ns = tmp2+1;
455         }
456         else if(tmp5==0.)Ns = tmp2;
457         if(tmp2>0.){
458           for(int iii=0;iii<tmp2;iii++){
459             x[iii]=(length-tmp5)/tmp2;
460           }
461         }
462       }
463       Schan[0]=tmp_chan;
464       if(Schan[0]==127)Ns=1;
465       if(tmpchan==1&&tmp_chip<14){
466         for(int ii=1;ii<Ns;ii++){
467           Schan[ii]=Schan[ii-1]-1;
468           if(Ns>128)cout<<"x["<<ii<<"]="<<x[ii]<<"␣chan="<<Schan[ii]<<endl;
469         }
470       }
471       if(tmpchan==1&&tmp_chip>13){
472         for(int ii=1;ii<Ns;ii++){
473           Schan[ii]=Schan[ii-1]+1;
474           if(Ns>128)cout<<"x["<<ii<<"]="<<x[ii]<<"␣chan="<<Schan[ii]<<endl;
475         }
476       }
477       if(tmpchan==-1&&tmp_chip<14){
478         for(int ii=1;ii<Ns;ii++){
479           Schan[ii]=Schan[ii-1]+1;
480           if(Ns>128)cout<<"x["<<ii<<"]="<<x[ii]<<"␣chan="<<Schan[ii]<<endl;
481         }
482       }
483       if(tmpchan==-1&&tmp_chip>13){
484         for(int ii=1;ii<Ns;ii++){
485           Schan[ii]=Schan[ii-1]-1;
486           if(Ns>128)cout<<"x["<<ii<<"]="<<x[ii]<<"␣chan="<<Schan[ii]<<endl;
487         }
488       }
489       int   Nhit_Re=0; // check channel number & number of strips after channel serection
490       double  new_x[128];
491       int   new_chan[128];
492       for(int i=0;i<Ns;i++){
493         if(Schan[i]>=0&&Schan[i]<=127){
494           new_x[Nhit_Re]=x[i];
495           new_chan[Nhit_Re]=Schan[i];
496           Nhit_Re++;
497         }
498       }
499       for(int i=0;i<Nhit_Re;i++){
500         m_SmV[i] = energyloss(gain,new_x[i])+offset; //energy loss per strip after channel serection
501       }
502       m_nhit = mV_adc(tmp_event, tmp_chip, Nhit_Re,new_x,new_chan,m_SmV,m_event,m_chip,m_x,m_chan, m_Edep, m_adc);//xpos, chan,
                ADC, Eloss
503       m_nhitorg = Nhit_Re;
504
505       for(int i=0;i<m_nhit;i++){
506       }
507     }
508     int   GetNhitorg()  {return m_nhitorg;}
509     int   GetNhit ()   {return m_nhit;}
510     int   GetChan (int i) {return m_chan[i];}
511     int   GetChip (int i) {return m_chip[i];}
```

```
512     int   GetAdc (int i) {return m_adc[i];}
513     double  GetEdep (int i) {return m_Edep[i];}
514     int   GetEvent(int i) {return m_event[i];}
515     int   *GetChanArray() {return m_chan;}
516     int   *GetChipArray() {return m_chip;}
517     int   *GetAdcArray() {return m_adc;}
518     double *GetEdepArray() {return m_Edep;}
519     double *GetSmVArray() {return m_SmV;}
520     int   *GetEventArray(){return m_event;}
521   };
522   void simulation(){
523     TH1F* hnhits    ;
524     TH1F*       nclsnChip  ;
525     TH1F* SingleAdc   ;
526     TH1F* DAdc    ;
527     TH1F* TAdc    ;
528     TH2F* multiclschip ;
529     TH1F* SumDoublehitAdc2 ;
530     TH1F* SinglehitAdc2 ;
531     TH1F* SumTriplehitAdc2 ;
532     TH2F* Adc_2hit ;
533     TH2F* chan_2hit ;
534     TH1F*       ELOSScls1    ;
535     TH1F*       ELOSScls2    ;
536     TH1F*       ELOSScls3    ;
537     TH1F*       ELOSSORG1    ;
538     TH1F*       ELOSSORG2    ;
539     TH1F*       ELOSSORG3    ;
540     TH2F* HITCORE ;
541     TH2F* ELOSSchange1 ;
542     TH2F* ELOSSchange2 ;
543     TH2F* ELOSSchange3 ;
544     TH1F*       KAIELOSSC1    ;
545     TH1F*       KAIELOSSC2    ;
546     TH1F*       KAIELOSSS1    ;
547     TH1F*       KAIELOSSS2    ;
548     int m=9;
549     double xbins[] = {0, 270, 302, 450, 602, 750, 902, 1050, 1202, 1234};
550     hnhits = new TH1F("hnhits","N␣hits␣per␣event",5,0,5);
551     SingleAdc = new TH1F("SingleAdc","energyloss␣before␣clustering",m,xbins);
552     DAdc = new TH1F("DAdc",  "energyloss␣before␣clustering",m,xbins);
553     TAdc = new TH1F("TAdc",  "energyloss␣before␣clustering",m,xbins);
554     SingleAdc->SetLineColor(kRed);
555     DAdc->SetLineColor(kBlue);
556     TAdc->SetLineColor(kGreen);
557     multiclschip = new TH2F("multiclschip","chip␣vs␣channel␣in␣multiple␣clsters␣event;chan_id;chip_id",128,0,128,26,0,26);
558     nclsnChip  = new TH1F("nclsnChip","N␣clusters␣per␣event;number␣of␣clusters",5,0,5);
559     SumDoublehitAdc2 = new TH1F("SumDoublehitAdc2","energy␣loss␣per␣cluster;mV",m,xbins);
560     SinglehitAdc2 = new TH1F("SinglehitAdc2","energy␣loss␣per␣cluster;mV",m,xbins);
561     SumTriplehitAdc2 = new TH1F("SumTriplehitAdc2","energy␣loss␣per␣cluster;mV",m,xbins);
562     Adc_2hit = new TH2F("Adc_2hit","adc␣correlation␣of␣double␣hits;ch1␣adc;ch2␣adc",12,-2.,10.,12,-2.,10.);
563     chan_2hit = new TH2F("chan_2hit","channel␣correlation␣of␣double␣hits;ch1␣channel;ch2␣channel",130,0.,130.,130,0.,130.);
564     HITCORE  = new TH2F("HITCORE","N␣strips␣(clustered␣vs␣original);clustered␣strips;original␣strips",5,0.,5.,5,0.,5.);
565     ELOSScls1    = new TH1F("ELOSScls1","original␣energyloss␣(clusterd␣hit␣number=1);mV",50,0.,1234.);
566     ELOSScls2    = new TH1F("ELOSScls2","original␣energyloss␣(clusterd␣hit␣number=2);mV",50,0.,1234.);
567     ELOSScls3    = new TH1F("ELOSScls3","original␣energyloss␣(clusterd␣hit␣number=3);mV",50,0.,1234.);
568     ELOSSORG1    = new TH1F("ELOSSORG1","original␣energyloss␣(original␣hit␣number=1);mV",50,0.,1234.);
569     ELOSSORG2    = new TH1F("ELOSSORG2","original␣energyloss␣(original␣hit␣number=2);mV",50,0.,1234.);
570     ELOSSORG3    = new TH1F("ELOSSORG3","original␣energyloss␣(original␣hit␣number=3);mV",50,0.,1234.);
571     ELOSSchange1 = new TH2F("ELOSSchange1","energyloss␣differense␣per␣clustered␣1␣strip;clusterd␣energyloss␣[mV];orginal␣energyloss␣[mV]",m,xbins,m,
            xbins);
572     ELOSSchange2 = new TH2F("ELOSSchange2","energyloss␣differense␣per␣clustered␣2␣strips;clusterd␣energyloss␣[mV];orginal␣energyloss␣[mV]",m,xbins,
            m,xbins);
573     ELOSSchange3 = new TH2F("ELOSSchange3","energyloss␣differense␣per␣clustered␣3␣strips;clusterd␣energyloss␣[mV];orginal␣energyloss␣[mV]",m,xbins,
            m,xbins);
574     KAIELOSSC1    = new TH1F("KAIELOSSC1","energyloss␣(simulation␣and␣cosmic);mV",m,xbins);
575     KAIELOSSC2    = new TH1F("KAIELOSSC2","energyloss␣(simulation␣and␣cosmic);mV",m,xbins);
576     KAIELOSSS1    = new TH1F("KAIELOSSS1","energyloss␣(simulation␣and␣cosmic);mV",m,xbins);
577     KAIELOSSS2    = new TH1F("KAIELOSSS2","energyloss␣(simulation␣and␣cosmic);mV",m,xbins);
578     SumDoublehitAdc2 ->SetLineColor(kBlue);
579     SinglehitAdc2   ->SetLineColor(kRed);
580     SumTriplehitAdc2 ->SetLineColor(kGreen);
581     ELOSScls1    ->SetLineColor(kRed);
582     ELOSScls2    ->SetLineColor(kBlue);
583     ELOSScls3    ->SetLineColor(kGreen);
584     ELOSSORG1    ->SetLineColor(kRed);
585     ELOSSORG2    ->SetLineColor(kBlue);
586     ELOSSORG3    ->SetLineColor(kGreen);
587     KAIELOSSC1    ->SetLineColor(kRed);
588     KAIELOSSC2    ->SetLineColor(kBlue);
589     KAIELOSSS1    ->SetLineColor(kOrange);
590     KAIELOSSS2    ->SetLineColor(kGreen);
591     gRandom->SetSeed();
592     double Vchip  = 9984.*(16000.*16.+20000.*10.); //26chip, micrometer
593     double Hchip  = 320.;          //micrometer
594     double Nevent = Vchip/100000000.; //26chips, N event per minute
595     double event  = 60.*10.*Nevent; //26chips, N event per 10 hours
596     for(int nevent = 0.; nevent<event; nevent++){ //N particle come
597       Mu mu1(nevent);
598       mu1.generateMu();
599       GlauberCalculation glauber;
600       glauber.calculate(&mu1);
601       double  lengthorg  = mu1.GetLength();
602       double elossorg  = energyloss(90., lengthorg);
```

```
603     int  hitorg   = glauber.GetNhitorg();
604     int   nhit    = glauber.GetNhit();
605     int*  chipArray  = glauber.GetChipArray();
606     int*  chanArray  = glauber.GetChanArray();
607     int*  adcArray  = glauber.GetAdcArray();
608     int*  eventArray = glauber.GetEventArray();
609     double*  EdepArray  = glauber.GetEdepArray();
610     double*  SmVArray  = glauber.GetSmVArray();
611     double summV_Ret[128];
612     int   Nhits_Ret[128];
613     int   chan_Ret[128];
614     int   chip_Ret[128];
615     double SUMORG=0;
616     for(int i=0;i<hitorg;i++){
617       if(hitorg==1){ELOSSORG1->Fill(SmVArray[i], mV_weight2(SmVArray[i]));}
618       if(hitorg==2){ELOSSORG2->Fill(SmVArray[i], mV_weight2(SmVArray[i]));}
619       if(hitorg==3){ELOSSORG3->Fill(SmVArray[i], mV_weight2(SmVArray[i]));}
620       SUMORG+=SmVArray[i];
621     }
622     double SUMEDEP=0.;
623     for(int i=0;i<nhit;i++){
624       SUMEDEP+=EdepArray[i];
625     }
626     //process_hits
627     int hitChip[27][800];
628     int hitChan[27][800];
629     int hitAdc [27][800];
630     int hitEvent[27][800];
631     int nchip=0;
632     int nHitChip[27];
633     for(int i=0;i<27;i++){
634       nHitChip[i] = 0;
635     }
636     int tmp1, tmp2, tmp3, tmp4, tmp5, tmp6;
637     int ii=0;
638     for(int ihit=0; ihit<nhit; ihit++){
639       int ichip = chipArray[ihit];
640       ii = nHitChip[ichip];
641       nHitChip[ichip]++;
642       hitChip[ichip][ii] = chipArray[ihit];
643       hitChan[ichip][ii] = chanArray[ihit];
644       hitAdc [ichip][ii] = adcArray[ihit];
645       hitEvent[ichip][ii]= eventArray[ihit];
646     }
647     double nhit_1[128], nhit_2[128], nhit_3[128];
648     for(int ii=0; ii<128;ii++){
649       nhit_1[ii]=0.;
650       nhit_2[ii]=0.;
651       nhit_3[ii]=0.;
652     }
653     double nhit_adc1[128], nhit_adc2[128], nhit_adc3[128];
654     for(int ii=0; ii<128;ii++){
655       nhit_adc1[ii]=0.;
656       nhit_adc2[ii]=0.;
657       nhit_adc3[ii]=0.;
658     }
659 // /////// sort channel order/////////////
660     for(int ichip=0; ichip<26; ichip++){
661       if(nHitChip[ichip]>0){
662         for(int ihit=0; ihit<nHitChip[ichip]; ihit++){
663           for(int jhit=ihit+1;jhit<nHitChip[ichip];jhit++){
664             if(hitChan[ichip][ihit]>hitChan[ichip][jhit]){
665               tmp1 = hitChan[ichip][ihit];
666               tmp2 = hitChip[ichip][ihit];
667               tmp3 = hitAdc[ichip][ihit];
668               tmp4 = hitEvent[ichip][ihit];
669               hitChan[ichip][ihit] = hitChan[ichip][jhit];
670               hitChip[ichip][ihit] = hitChip[ichip][jhit];
671               hitAdc[ichip][ihit] = hitAdc[ichip][jhit];
672               hitEvent[ichip][ihit] = hitEvent[ichip][jhit];
673               hitChan[ichip][jhit] = tmp1;
674               hitChip[ichip][jhit] = tmp2;
675               hitAdc[ichip][jhit] = tmp3;
676               hitEvent[ichip][jhit] = tmp4;
677             }
678           }
679         }
680       }
681     }
682     for(int ichip=0; ichip<27; ichip++){
683       if(nHitChip[ichip]>1){
684         int tmpchip[26];
685         int tmpchan[128];
686         for(int i=0;i<127;i++)tmpchan[i]=-1;
687         for(int i=0;i<26;i++)tmpchip[i]=-1;
688         int noise =1;
689         int tmptmp=0;
690         for(int ihit=0; ihit<nHitChip[ichip]; ihit++){
691           if(hitChip[ichip][ihit]==hitChip[ichip][ihit+1]&&hitChan[ichip][ihit+1]==hitChan[ichip][ihit]){
692             noise++;
693             if(tmpchan[tmptmp]!=hitChan[ichip][ihit]&&tmpchip[tmptmp]!=hitChip[ichip][ihit]){
694               tmptmp++;
695             }
696             tmpchan[tmptmp]=hitChan[ichip][ihit];
```

```
697             tmpchip[tmptmp]=hitChip[ichip][ihit];
698           }
699         }
700         for(int ttt=0;ttt<tmptmp+1;ttt++){
701           for(int ihit=0; ihit<nHitChip[ichip]+1; ihit++){
702             if(hitChip[ichip][ihit]==tmpchip[ttt]&&hitChan[ichip][ihit]==tmpchan[ttt]){
703             }
704           }
705         }
706       }
707     }
708     int nhitmap=0;
709     double summV;
710     int sumChan;
711     int ncls = 0;
712     for(int ichip=0; ichip<27; ichip++){
713       if(nHitChip[ichip]>0){
714         int start = 0;
715         int Nhits =1;
716         int end;
717         for(int ihit=0; ihit<nHitChip[ichip]; ihit++){
718           if(Nhits==nHitChip[ichip]){
719             end = ihit;
720             summV=0;
721             sumChan =0;
722             for(int i=start; i<end+1; i++){
723               summV += adc_mV(hitAdc[ichip][i]);
724               sumChan += hitChan[ichip][i];
725             }
726             if(Nhits==1){
727               for(int ii=0; ii<8;ii++){
728                 if(mV_adc2(summV)==ii)nhit_1[ii]++;
729               }
730             }
731             else if(Nhits==2){
732               for(int ii=0; ii<8;ii++){
733                 if(mV_adc2(summV)==ii)nhit_2[ii]++;
734               }
735             }
736             else if(Nhits==3){
737               for(int ii=0; ii<8;ii++){
738                 if(mV_adc2(summV)==ii)nhit_3[ii]++;
739               }
740             }
741             start = end+1;
742             ncls++;
743             Nhits = 1;
744           }
745           else if(hitChan[ichip][ihit+1]-hitChan[ichip][ihit]==1){
746             Nhits++;
747           }
748           else{
749             end = ihit;
750             summV=0;
751             sumChan =0;
752             for(int i=start; i<end+1; i++){
753               summV += adc_mV(hitAdc[ichip][i]);
754               sumChan += hitChan[ichip][i];
755             }
756             if(Nhits==1){
757               for(int ii=0; ii<8;ii++){
758                 if(mV_adc2(summV)==ii)nhit_1[ii]++;
759               }
760             }
761             else if(Nhits==2){
762               for(int ii=0; ii<8;ii++){
763                 if(mV_adc2(summV)==ii)nhit_2[ii]++;
764               }
765             }
766             else if(Nhits==3){
767               for(int ii=0; ii<8;ii++){
768                 if(mV_adc2(summV)==ii)nhit_3[ii]++;
769               }
770             }
771             start = end+1;
772             ncls++;
773             Nhits = 1;
774           }
775         }
776       }
777     }
778 // ++++++++++++++++++++++++++++++++++
779     if(ncls>1){
780       for(int ichip=0; ichip<27; ichip++){
781         if(nHitChip[ichip]>0){
782           for(int ihit=0; ihit<nHitChip[ichip]; ihit++){
783             multiclschip->Fill(hitChan[ichip][ihit],hitChip[ichip][ihit]);
784           }
785         }
786       }
787     }
788     if(ncls>0){                    //cluster number for analysis
789       nhitmap=0;
790       int Ncls = 0;                  //number of strips (same as ncls)
```

```
791         int Nhits =0;
792         for(int ichip=0; ichip<27; ichip++){
793         int NHITCHIP=0;  //number of single hits per chip per event
794         int single_ihit[20];
795         int single_0_ihit[20];
796           if(nHitChip[ichip]>0){
797             int start = 0;
798             Nhits =1;
799             int end;
800             for(int ihit=0; ihit<nHitChip[ichip]; ihit++){
801               if(Nhits==nHitChip[ichip]){
802                 end = ihit;
803                 summV=0;
804                 sumChan =0;
805                 for(int i=start; i<end+1; i++){
806                   summV += adc_mV(hitAdc[ichip][i]);
807                   sumChan += hitChan[ichip][i];
808                 }
809                 summV_Ret[NHITCHIP] = summV;
810                 Nhits_Ret[NHITCHIP] = Nhits;
811                 chip_Ret[Ncls] = hitChip[ichip][ihit];
812                 chan_Ret[Ncls] = sumChan/Nhits;
813                 hnhits->Fill(Nhits);
814 // +++++++++ Fill single hit ++++++++++++
815                 if(Nhits==1){
816                   HITCORE->Fill(Nhits, hitorg);
817                   if(Nhits==1)ELOSScls1->Fill(SUMEDEP, mV_weight2(SUMEDEP));
818                   if(Nhits==1)ELOSSchange1->Fill(summV,elossorg );
819                   single_ihit[NHITCHIP]=start;
820                   NHITCHIP++;
821                   if(hitAdc[ichip][start]!=-1){      //ADC value for analysis
822                     SinglehitAdc2->Fill(summV,mV_weight2(summV));
823                     SingleAdc->Fill(adc_mV(hitAdc[ichip][start]), mV_weight2(adc_mV(hitAdc[ichip][start])));
824                     for(int ii=0; ii<8;ii++){
825                       if(mV_adc2(summV)==ii)nhit_1[ii]++;
826                     }
827                   }
828                   else if(hitAdc[ichip][start]==0)single_0_ihit[NHITCHIP]=start;
829                 }
830 // +++++++++ Fill double hits ++++++++++++
831                 else if(Nhits==2){
832                   HITCORE->Fill(Nhits, hitorg);
833                   if(Nhits==2)ELOSScls2->Fill(SUMEDEP, mV_weight2(SUMEDEP));
834                   if(Nhits==2)ELOSSchange2->Fill(summV,elossorg );
835                   summV = adc_mV(hitAdc[ichip][start])+adc_mV(hitAdc[ichip][start+1]);
836                   SumDoublehitAdc2->Fill(summV,mV_weight2(summV));
837                   DAdc->Fill(adc_mV(hitAdc[ichip][start]), mV_weight2(adc_mV(hitAdc[ichip][start])));
838                   DAdc->Fill(adc_mV(hitAdc[ichip][start+1]), mV_weight2(adc_mV(hitAdc[ichip][start+1])));
839                   Adc_2hit->Fill(hitAdc[ichip][start],hitAdc[ichip][start+1]);
840                   chan_2hit->Fill(hitChan[ichip][start],hitChan[ichip][start+1]);
841                   if(hitAdc[ichip][start]==0&&hitAdc[ichip][start+1]==0){
842                   }
843                   for(int ii=0; ii<8;ii++){
844                     if(mV_adc2(summV)==ii)nhit_2[ii]++;
845                   }
846                 }
847 // +++++++++ Fill triple hits ++++++++++++
848                 else if(Nhits==3){
849                   HITCORE->Fill(Nhits, hitorg);
850                   if(Nhits==3)ELOSScls3->Fill(SUMEDEP, mV_weight2(SUMEDEP));
851                   if(Nhits==3)ELOSSchange3->Fill(summV,elossorg );
852                   SumTriplehitAdc2->Fill(summV,mV_weight2(summV));
853                   TAdc->Fill(adc_mV(hitAdc[ichip][start]),  mV_weight2(adc_mV(hitAdc[ichip][start])));
854                   TAdc->Fill(adc_mV(hitAdc[ichip][start+1]), mV_weight2(adc_mV(hitAdc[ichip][start+1])));
855                   TAdc->Fill(adc_mV(hitAdc[ichip][start+2]), mV_weight2(adc_mV(hitAdc[ichip][start+2])));
856                   for(int ii=0; ii<8;ii++){
857                     if(mV_adc2(summV)==ii)nhit_3[ii]++;
858                   }
859                 }
860                 start = end+1;
861                 Ncls++;
862                 Nhits = 1;
863               }
864               else if(hitChan[ichip][ihit+1]-hitChan[ichip][ihit]==1){
865                 Nhits++;
866               }
867               else{
868                 end = ihit;
869                 summV=0;
870                 sumChan =0;
871                 for(int i=start; i<end+1; i++){
872                   summV += adc_mV(hitAdc[ichip][i]);
873                   sumChan += hitChan[ichip][i];
874                 }
875                 summV_Ret[NHITCHIP] = summV;
876                 Nhits_Ret[NHITCHIP] = Nhits;
877                 chip_Ret[Ncls] = hitChip[ichip][ihit];
878                 chan_Ret[Ncls] = sumChan/Nhits;
879                 hnhits->Fill(Nhits);
880 // +++++++++ Fill single hit ++++++++++++
881                 if(Nhits==1){
882                   HITCORE->Fill(Nhits, hitorg);
883                   if(Nhits==1)ELOSScls1->Fill(SUMEDEP, mV_weight2(SUMEDEP));
884                   if(Nhits==1)ELOSSchange1->Fill(summV,elossorg );
```

```
885                 single_ihit[NHITCHIP]=start;
886                 NHITCHIP++;
887                 if(hitAdc[ichip][start]!=-1){
888                   SinglehitAdc2->Fill(summV,mV_weight2(summV));
889                   SingleAdc->Fill(adc_mV(hitAdc[ichip][start]), mV_weight2(adc_mV(hitAdc[ichip][start])));
890                   for(int ii=0; ii<8;ii++){
891                     if(mV_adc2(summV)==ii)nhit_1[ii]++;
892                   }
893                 }
894                 else if(hitAdc[ichip][start]==0)single_0_ihit[NHITCHIP]=start;
895               }
896 // ++++++++ Fill double hits ++++++++++++
897               else if(Nhits==2){
898                 HITCORE->Fill(Nhits, hitorg);
899                 if(Nhits==2)ELOSScls2->Fill(SUMEDEP, mV_weight2(SUMEDEP));
900                 if(Nhits==2)ELOSSchange2->Fill(summV,elossorg );
901                 SumDoublehitAdc2->Fill(summV,mV_weight2(summV));
902                 DAdc->Fill(adc_mV(hitAdc[ichip][start]),  mV_weight2(adc_mV(hitAdc[ichip][start])));
903                 DAdc->Fill(adc_mV(hitAdc[ichip][start+1]), mV_weight2(adc_mV(hitAdc[ichip][start+1])));
904                 Adc_2hit->Fill(hitAdc[ichip][start],hitAdc[ichip][start+1]);
905                 chan_2hit->Fill(hitChan[ichip][start],hitChan[ichip][start+1]);
906                 if(hitAdc[ichip][start]==0&&hitAdc[ichip][start+1]==0){
907                 }
908                 for(int ii=0; ii<8;ii++){
909                   if(mV_adc2(summV)==ii)nhit_2[ii]++;
910                 }
911               }
912 // ++++++++ Fill triple hits ++++++++++++
913               else if(Nhits==3){
914                 HITCORE->Fill(Nhits, hitorg);
915                 if(Nhits==3)ELOSScls3->Fill(SUMEDEP, mV_weight2(SUMEDEP));
916                 if(Nhits==3)ELOSSchange3->Fill(summV,elossorg );
917                 SumTriplehitAdc2->Fill(summV,mV_weight2(summV));
918                 TAdc->Fill(adc_mV(hitAdc[ichip][start]) , mV_weight2(adc_mV(hitAdc[ichip][start])));
919                 TAdc->Fill(adc_mV(hitAdc[ichip][start+1]), mV_weight2(adc_mV(hitAdc[ichip][start+1])));
920                 TAdc->Fill(adc_mV(hitAdc[ichip][start+2]), mV_weight2(adc_mV(hitAdc[ichip][start+2])));
921                 for(int ii=0; ii<8;ii++){
922                   if(mV_adc2(summV)==ii)nhit_3[ii]++;
923                 }
924               }
925               start = end+1;
926               Ncls++;
927               Nhits = 1;
928             }
929           }
930           nclsnChip->Fill(Ncls);
931         }
932       }
933     }
934   }
935   TCanvas *c1 = new TCanvas("c1", "energy loss", 1000, 500);
936   c1->Divide(2,1);
937   TCanvas *c2 = new TCanvas("c2", "correlation of double hits", 1000, 500);
938   c2->Divide(2,1);
939   TCanvas *c4 = new TCanvas("c4", "cosmic", 1500, 500);
940   c4->Divide(3,1);
941   TCanvas *c7 = new TCanvas("c7", "energyloss and cluster change", 1500,1000);
942   c7->Divide(3,2);
943   TCanvas *c8 = new TCanvas("c8", "energylos(simulation and cosmic)", 500, 500);
944   TCanvas *c9 = new TCanvas("c9", "energyloss", 500, 500);
945   c1->cd(1);//before cluster adc
946   DAdc->Draw("hist");
947   SingleAdc->Draw("histsames");
948   c1->cd(2);//after cluster mV
949   SinglehitAdc2  ->Draw("hist");
950   SumDoublehitAdc2->Draw("histsames");
951   c9->cd();//after cluster mV
952   SinglehitAdc2  ->Draw("hist");
953   SumDoublehitAdc2->Draw("histsames");
954 //+++++for Kai2++++++++++++++++++++++++
955   int   nhit_1[10];
956   int   nhit_2[10];
957   double  sum1=0;
958   double  sum2=0;
959   double  W1=0.;
960   double  Kai2=0;
961   double  cosmic1[10]={0.,0.,55., 38., 384., 169., 31., 13., 3., 6.};
962   double  cosmic2[10]={0.,0.,0., 0., 0., 87., 124., 35., 17., 0.};
963   for(int ii=0;ii<10;ii++){
964     nhit_1[ii]= 0.;
965     nhit_2[ii]= 0.;
966   }
967   for(int ii=2;ii<10;ii++){
968     nhit_1[ii]= SinglehitAdc2  ->GetBinContent(ii);
969     nhit_2[ii]= SumDoublehitAdc2->GetBinContent(ii);
970   }
971   for(int ii=3;ii<10;ii++){
972     cout<<nhit_1[ii]<<", ";
973     sum1+=nhit_1[ii];
974   }
975   cout<<" "<<endl;
976   for(int ii=3;ii<10;ii++){
977     cout<<nhit_2[ii]<<", ";
978     sum2+=nhit_2[ii];
```

```
979     }
980     W1=(699.-55.+263.)/(sum1+sum2);
981     cout<<"␣W1="<<W1<<endl;
982     cout<<"␣sum="<<sum1+sum2<<endl;
983     for(int ii=3;ii<10;ii++){
984       if(nhit_1[ii]!=0)Kai2+=(cosmic1[ii]-nhit_1[ii]*W1)*(cosmic1[ii]-nhit_1[ii]*W1)/(nhit_1[ii]*W1);
985       cout<<"Kai2["<<ii<<"]=␣"<<(cosmic1[ii]-nhit_1[ii]*W1)*(cosmic1[ii]-nhit_1[ii]*W1)/(nhit_1[ii]*W1)<<endl;
986     }
987     for(int ii=3;ii<10;ii++){
988       if(nhit_2[ii]!=0)Kai2+=(cosmic2[ii]-nhit_2[ii]*W1)*(cosmic2[ii]-nhit_2[ii]*W1)/(nhit_2[ii]*W1);
989       cout<<"Kai2["<<ii<<"]=␣"<<(cosmic2[ii]-nhit_2[ii]*W1)*(cosmic2[ii]-nhit_2[ii]*W1)/(nhit_2[ii]*W1)<<endl;
990     }
991     cout<<"Kai2="<<Kai2<<endl;
992     float mV[9] = {270, 302, 450, 602, 750, 902, 1050, 1202, 1234};
993     for(int ii=2;ii<10;ii++){
994       for(int i=0;i<cosmic1[ii];i++){
995         if(cosmic1[ii]!=0)KAIELOSSC1->Fill(adc_mV(ii-2));
996       }
997       for(int i=0;i<cosmic2[ii];i++){
998         if(cosmic2[ii]!=0)KAIELOSSC2->Fill(adc_mV(ii-2));
999       }
1000      for(int i=0;i<W1*nhit_1[ii];i++){
1001        if(nhit_1[ii]!=0) KAIELOSSS1->Fill(adc_mV(ii-2));
1002      }
1003      for(int i=0;i<W1*nhit_2[ii];i++){
1004        if(nhit_2[ii]!=0) KAIELOSSS2->Fill(adc_mV(ii-2));
1005      }
1006    }
1007    c8->cd();
1008    KAIELOSSC1->Draw("hist");
1009    KAIELOSSC2->Draw("histsames");
1010    KAIELOSSS1->Draw("histsames");
1011    KAIELOSSS2->Draw("histsames");
1012  //+++++++++++++++++++++++++++++++++++++
1013    c2->cd(1);
1014    Adc_2hit->Draw("colz");
1015    c2->cd(2);
1016    chan_2hit->Draw("colz");
1017    c4->cd(1);//number of hits per event
1018    hnhits->Draw();
1019    c4->cd(2);//number of clusters per event
1020    nclsnChip->Draw();
1021    c4->cd(3);//multiple cluster hit chip
1022    multiclschip->Draw("colz");
1023    //only for simulation
1024    c7->cd(1);//number of hits per event
1025    HITCORE  ->Draw("colz");
1026    c7->cd(2);//number of hits per event
1027    ELOSSORG1 ->Draw("hist") ;
1028    ELOSSORG2 ->Draw("histsames");
1029    ELOSSORG3 ->Draw("histsames");
1030    c7->cd(3);//number of hits per event
1031    ELOSScls1 ->Draw("hist") ;
1032    ELOSScls2 ->Draw("histsames");
1033    ELOSScls3 ->Draw("histsames");
1034    c7->cd(4);//number of hits per event
1035    ELOSSchange1->Draw("colz");
1036    c7->cd(5);//number of hits per event
1037    ELOSSchange2->Draw("colz");
1038    c7->cd(6);//number of hits per event
1039    ELOSSchange3->Draw("colz");
1040    gStyle->SetOptStat(1000000010);
1041  };
```