Fermi National Accelerator Laboratory

FERMILAB-TM-1798

A Study of the Intel ETANN VLSI Neural Network for an Electron Isolation Trigger

Clark S. Lindsey and Bruce Denby

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

October 1992

Operated by Universities Research Association Inc. under Contract No. DE-AC02-76CHO3000 with the United States Department of Energy

Disclaimer

A Study of the Intel ETANN VLSI Neural Network for an Electron Isolation Trigger

Clark S. Lindsey*, Bruce Denby
Fermi National Accelerator Laboratory

Abstract

A neural network circuit structure was previously proposed for making an isolated electron trigger for the CDF plug calorimeter. Here we discuss a study of the isolation performance of the Intel *ETANN* (Electrically Trainable Analog Neural Network) chip. We used the PC based ETANN development system and Monte Carlo generated shower patterns to test the chip. A C++ application program was developed that runs within the development system but extends chip control and testing capabilities beyond the standard routines. With this program several configurations of the chip parameters were tested and the results are discussed here.

1. Introduction

A neural network circuit structure was proposed for the CDF plug calorimeter isolation trigger [1,2]. A Level-2 isolation cut would allow a lowering of the Level-2 trigger threshold and so increase the efficiency and acceptance for W's, top, Drell-Yan and direct photons. The circuit would be presented with 50 tower energies centered on the seed tower found by the cluster finder. The circuit examines the ratio of the energies near and including the seed tower to the surrounding tower energies to determine if the seed tower cluster is isolated.

Figure 1 shows four *templates* where the grid represents a 5x5 array of EM towers around the seed tower. Available are also 25 hadronic towers centered on the seed tower position. The hatched regions represent the four towers around and including the central seed tower. The energy sum of these four towers is called here the inner energy (E_{inner}). The four templates show the four possible patterns of the inner four towers with the seed tower at the center of the array. The energy sum of the other 21 EM towers plus the 25 hadronic towers is called the outer energy (E_{outer}). We define the isolation variable f^{isol} as:

$$f^{isol} = \frac{E_{outer}}{E_{inner}}$$

The shower is considered isolated if f^{isol} is smaller than some cut f^{isol}_{cut}. The ratio could be calculated explicitly with a digital circuit but would be slower than a simple

*Current address: 1261 Foran Ln. Aurora, Il. 60506

analog circuit. The analog circuit just compares the difference of two *weighted* values and fires if this difference is positive or negative:

$$f_{cut}^{isol} E_{inner} - E_{outer} > 0 \Longrightarrow isolated,$$

where E_{inner} is weighted by f_{cut}^{isol} and E_{outer} is weighted by 1.0.

This computation is very similar to what a neural circuit does. An electronic neuron is basically a threshold amplifier which fires if the total input signal is greater than zero. Signals sent to the neuron inputs are multiplied by a *weight* in the synapse and a sum of the weighted signals is presented to the amplifier. Here the inputs would be voltages proportional to the tower energies and the weights would be f_{cut}^{isol} for each of the four inner towers and -1.0 for the outer 46 towers:

$$N_{input} = f_{cut}^{isol} \sum_{i=1}^{4} V_{inner}^{i} - 1.0 \sum_{i=1}^{46} V_{outer}^{i}. \quad (1)$$

The threshold characteristic of the neuron response is typically (see fig. 3):

$$N_{output} = \frac{A}{1.0 + \exp(-GN_{input})}. \quad (2)$$

where constant A is the maximum output and constant G is a gain factor. If G is very large the neuron acts as a binary function, fully turning on whenever the total input slightly exceeds zero. Four such neuron circuits could provide the isolation trigger bits for the four templates of figure 1.

A discrete component circuit could be built to carry out this network (see, for example, ref. [3]). No training is involved so the weighting can be done with simple resistors. However, the Intel chip described in the next section would allow greater flexibility, e.g. one could easily experiment with different isolation cuts. Also, a B trigger system is already in place that uses the Intel chip [4,5,6]. (There is also a central calorimeter isolation trigger similar to the plug isolation trigger but it simply cuts on the sum of the outer tower energies rather than on the ratio of outer to inner[2].) The plug isolation trigger is a straightforward duplication of this system and allows for hardware and software compatibility.

2. Intel *ETANN* Chip

The Intel Electrically Trainable Analog Neural Network (ETANN) has been described previously [7,8]. The chip has 64 neurons or threshold amplifiers with sigmoidal response. Effectively, however, there are 128 neurons since the same 64 neurons are used for both the middle and output layers. A signal (analog voltage 0.0v to 3.5v) entering one of the 64 inputs is presented to a synapse. The synapse design is similar to a Gilbert multiplier circuit which produces a differential output current proportional to the multiplication of two differential voltages. Here the difference in threshold voltages (Vt) of two floating gates provides the weight value:

$$I_{output} \approx (V_{input} - V_{ref_i}) \times (Vt_1 - Vt_2). \quad (3)$$

Figure 2 shows the current outputs for various Vt_{dif} values versus the input voltage for $V_{refi} = 1.4v$. Note that the outputs are not perfectly linear, especially for the low weights and for high Vin values. The weights values are limited to approximately ± 2.5 and have about 6 bit precision. Charge can be made to tunnel onto the floating gates with large voltage pulses. The synapses are non-volatile and can remain stable for several years.

There are 64 inputs plus 16 internal constant (or bias) voltages connected to each neuron through 80 synapses. So the neurons see current sums proportional to the dot product of 80 input-synapse products. There are two sets of 80x64=5120 synapses. One set is for the first layer and the other for second layer processing where the first layer outputs are clocked back through these synapses and the neurons reused for the second layer outputs.

The neuron response, for a standard input reference voltage (V_{refi}) of 1.4v and standard output reference voltage (V_{refo}) of 1.4v, is approximated by the sigmoidal function:

$$f(x_j) = \frac{2.55v}{1 + \exp x_j} + 0.1v, \tag{4}$$

where x_j is

$$x_j = G\left[\sum_k w_{jk}(V_k - 1.4v) + \sum_b w_{jb}(V_b - 1.4v)\right]. \tag{5}$$

Here V_k is the input voltage. The w_{jk} is the weight for the connection between receiving unit j and sending unit k. Of the 16 internal biases ($V_b \simeq 4.0v$), seven are available for the user (the other 9 are reserved for the initialization of the chip by the development system described below.) The gain G can be varied with an external control voltage (V_{gain}) with a maximum of about 1.3. For V_{gain} mode the output ranges from about 0.1 to a little less than $2 \times V_{refo}$. A special binary mode (H_{gain}) with fast turn on from 0.0v to 5.0v is also available by setting a control input on. Figure 3 shows how the V_{gain} values affect the sigmoid and figure 4 shows comparison of V_{gain} and H_{gain} modes.

After signals are presented at the inputs, the first layer neuron outputs will reach final levels within about $3\mu s$ for V_{gain} mode and $0.8\mu s$ for H_{gain} . As mentioned above, a second layer mode is also available but here we only need to use the one layer mode.

A pc-based development system for the ETANN is available[9]. The Intel *iNNTS* system allows one to do such things as initialize the chip, read weights from or write weights to the chip, emulate the chip [10], and do chip-in-the-loop training (CIL). However, the standard routines may not be sufficient so also provided is a library of subroutines one may call with one's own program. Here we used the *iNNTS* system primarily for initializing the chip. It automatically sets all weights to 0.0 and uses the internal biases (7-16) to set each neuron to half maximum activation for zero input. All further testing was done with a custom program discussed in the appendix.

3. Test Procedure

To test the chip's ability to perform the isolation trigger we used the following procedure (with occasional variations as discussed later):

1. The chip was initialized with the control voltages (V_{gain} or H_{gain} mode, V_{refi} and V_{refo}) for a particular test configuration.

2. The first four neurons were chosen to represent the four templates (fig. 1). The synapses connecting a neuron to the first 50 inputs were loaded with: f_{cut}^{isol} for each of the synapses connected to the inner four towers and -1.0 for the other 46 synapses. (These weights can also be changed by some constant factor as long as the ratio of the inner and outer weights remains f_{cut}^{isol}.)

3. We want the chip to perform the sum of equation 1 and to use that sum for its output function in equation 2. However, in equations 3 and 4, we see that V_{refi} is subtracted from the outputs. So the sum is actually:

$$N_{input} = f_{cut}^{isol} \sum_{i=1}^{4} (V_{inner}^{i} - V_{refi}) - 1.0 \sum_{i=1}^{46} (V_{outer}^{i} - V_{refi}) + Bias \qquad (6)$$

or

$$N_{input} = f_{cut}^{isol} \sum_{i=1}^{4} V_{inner}^{i} - \sum_{i=1}^{46} V_{outer}^{i} + (46 - 4f_{cut}^{isol}) V_{refi} + Bias \qquad (7)$$

It is therefore necessary to use the $Bias$ (see second term in equation 5) to eliminate the term multiplied by V_{refi} . Note that for the bias we can either use 7 internal bias voltages or use the spare 14 inputs (51-64) by placing a fixed voltage on them.

4. f_{cut}^{isol} was found in reference [1] to be optimum at 0.16 so this was the target cut value.

5. To generate a shower pattern, first an E_{inner} and E_{outer} were chosen. A conversion of 1v=10GeV was used.

6. For a chosen template, the E_{inner} of the four inner towers were assigned random fractions of this value with the seed tower restricted to having at least 6GeV. The 46 outer towers were assigned random fractions of the E_{outer} with the restriction that no one tower have more than 25% of E_{outer} .

7. The pattern of voltages for a given event were presented to the chip and the output for each template neuron recorded and compared to the f^{isol} of that template.

8. Scanning f^{isol} from 0.05 to 0.30 for the chosen template, the value of f^{isol} where the transition from above a threshold to below it was recorded.

9. E_{inner} was increased from 10GeV to 28GeV in one step increments and the transition value was found at each step by the scanning of f^{isol} . The average of these transition values was calculated. The regular Level-2 electron trigger kicks in around total EM energy of 20GeV. So the most important range for the isolation trigger is roughly $10GeV < E_{inner} < 20GeV$ (a threshold cut determines the lower limit). We'll present results for that range and for $10GeV < E_{inner} < 28GeV$.

10. The average transition from high to low should occur at f_{cut}^{isol} but because of the lack of precision in the weights and to other imperfections, the transition would usually occur away from this value. So the weights connecting the neuron to the bias voltages were modified until the average transition occurred at f_{cut}^{isol} .

4. Results

4.1. Stability of Isolation Cut

The first tests used the standard reference voltage of V_{refi} =1.4. The lower 7 internal bias voltages ($\sim 4.0v$) were used to subtract out the V_{refi} term (see above). However, with $V_{refi} = 1.4$ there was not enough bias to subtract out this term if f_{cut}^{isol} of 0.16 was used. So a smaller value of 0.10 was used for the inner tower weights and 0.625 was used for the outer tower weights (the ratio is still 0.16). The upper inputs of 51-64 were set to 0.0v.

In this configuration it was found that the average transition could be readily set to 0.16 ± 0.04. However, the transition would eventually begin to move away from this value, sometimes within minutes, by several percentage points. The chip was quite hot to the touch and the transition points seem to vary if the chip cooling was altered. One possibility was that the internal bias voltages were sensitive to the heat. So we set the weights to these internal biases to zero. For substitute biases we set the upper inputs (51-64) to 3.0v and adjusted the weights to these inputs to vary the transition point. This seemed to help somewhat but there were still significant variations of the average transition point with time.

Equations 3 and 4 show that V_{refi} is subtracted from each input. In a given event most of the outer inputs are small or zero values. This means that the synapses are producing a current proportional to -1.4×-1.0 (eq. 3 and also see fig.1). For 46 inputs this produces a sizable current and therefore substantial heat. To reduce the current V_{refi} was reduced to 0.2v. (It wasn't reduced all the way to zero because of extreme non-linear behavior there.) This reduces the total current by about factor of 49. In this mode the chip became much cooler and the variation in time was reduced substantially. There is still some variation when the chip is first powered up but it goes to it's nominal value within a minute or two. We still are using the upper inputs as biases although perhaps the reduced heat would now make the internal biases more reliable.

4.2. Hgain Mode

We first discuss results with the chip in H_{gain} mode. With the smaller V_{refi} (=0.2v), the constant V_{refi} term in equation 7 was much smaller. With the available bias it was possible to have bigger weights and still eliminate this term. So instead of 0.16 and -1.0, we used 0.32 and -2.0 for the inner and outer weights, respectively. This gave more significant products when multiplying small input signals.

Figure 5a-d shows the output voltage of the 4 template neurons in H_{gain} mode as function of f^{isol} for E_{inner} from 10GeV to 20GeV. As E_{inner} is increased in 1GeV steps, f^{isol} is scanned from 0.5 to 0.30. So figure 5 shows the results of 11 such scans for each template. Figure 6 shows a simulated comparator output where the output is 1.0 if the voltage was greater than 1.4v and as 0.0 if the output is less than 1.4v. A comparator is available on the trigger board to make this threshold cut. The transitions were set to occur at 0.16. Figures 7 and 8 show similar plots for 10GeV to 28GeV range.

As f^{isol} is increased, sometimes the output goes low, say at $f^{isol} = 0.15$, and then comes back up at, say, 0.17, and falls again at perhaps 0.18. This *bouncing* usually occurs no more than once for a given E_{inner} scan, and around 4-5 times for the 11 steps between 10GeV-20GeV. In figure 6 and 8 this produces the overlap region where the outputs are sometimes high and sometimes low for same f^{isol} . Note that the input patterns (i.e. the amount of energy in each of the 50 towers) are randomized for every event. So even if two events have the same E_{inner} and E_{outer} the neuron response can be different since the patterns can be very different. If we calculate the average transition using all transition points, including these extra bounce points, the average transition has an rms of about 0.02. If we just use the position of the f^{isol} where the first transitions occurred, the rms is about 0.01.

Figure 9 shows the f^{isol} at the first transition as function of E_{inner} up to 28Gev. Figure 10 shows distribution of f^{isol} at first first transition and figure 11 shows f^{isol} for all transitions. The transitions widths are roughly within the 0.02 range.

4.3. Vgain Mode

The H_{gain} mode would seem the natural mode to implement this network since a sharp transition at threshold is desired. However, the H_{gain} mode output behavior seemed somewhat more sensitive to variations in temperature and time than the V_{gain} mode. Since a comparator was available on the trigger board, using V_{gain} output was feasible since the comparator would produce a binary output according to a threshold setting.

As seen in figure 3 and 4, the slope of the V_{gain} output is considerably smaller than for H_{gain} . However, by using $V_{gain} = 5.0$v, weights of 0.32 and -2.00, and V_{refo} = 1.7v, we obtained a suitable output slope. The isolation transition point used a threshold of 1.55v as the piace where the transition from on to off occurred.

Figure 12 shows the output voltage of the 4 template neurons in V_{gain} mode for $10GeV < E_{inner} < 20GeV$ as function of f^{isol} . Figure 13 shows the the simulated comparator output with $out = 1.0$ if the voltage is above 1.55v and $out = 0.0$ if the

voltage is below 1.55v. Figures 14-15 show similar plots for the $10GeV < E_{inner} < 28GeV$ range.

Figure 16 shows f^{isol} at first transition versus E_{inner} . Figure 17 shows distribution of f^{isol} at first transition and figure 18 shows f^{isol} for all transitions. The rms of the first transitions is about 0.01 and rms of all transitions is about 0.015.

5. Discussion

The test here shows that the ETANN can perform an isolation cut at the 0.02 precision level. The chip is primarily aimed for applications where the network is trained and so the various non-linearities, limited precision weights, etc. can be compensated for in the training process. The isolation task here tends to emphasize the weaknesses of the chip and not its strengths. However, for the plug isolation trigger the input signals are somewhat coarse so the precision of the chip is sufficient. The strategy recommended here would be to set the point of the average *first* transition at 0.16. This would avoid cutting the isolated showers that one desires. If the output bounces back up at higher f^{isol} values, then more events are accepted but, assuming the rate is not increased dramatically, these events can be cut by the Level-3 electron trigger.

Tests of time variation of the average transitions indicate that the V_{gain} mode is more stable than the H_{gain} mode. A chip in V_{gain} mode had average transitions remain within 0.01 of their values over several weeks (most of the time the chip was disconnected from power as well), whereas a H_{gain} chip showed variations at the 0.02-0.03 level.

Currently a chip in Hgain mode is installed in a Michigan neural network trigger board[4]. Test patterns sent to the chip seem to indicate similar performance there as in the trainer. A chip set in V_{gain} mode is also available and will be tried later. We hope to see plug isolated events triggered by the chip in actual data in the near future.

Acknowledgements

We thank Myron Campbell, Dolly Wu, and Kevin Burkett for their advice and help. Thanks also to Ken Johns and G. Pauletta for use of their ETANN development systems.

Appendix: A C++ Neural Network Test Program

The *iNNTS* development system comes with a package of programs that initialize and test the chip and include two neural network emulators: Dynamind [10] and Brainmaker [11]. In the past we have primarily used the Dynamind emulator for chip emulation and chip-in-the-loop training. Dynamind seems to have been written specifically for the ETANN and has worked well for training the chip. For the isolation trigger studies here, however, there were several problems with using Dynamind. For example, the inputs and outputs are defined on a scale of 0.0v to $2 \times V_{refi}$. As noted in the text, we found that having a small V_{refi} improved the

stability of the chip. We still wanted to have inputs in the 0.0v to 3.5v range, but in Dynamind we would be limited to 0.0v to 0.4v if $V_{refi} = 0.2$v.

For this reason and also for increased flexibility, we decided to develop custom programs that use the library of routines supplied with the software package. This package is written in Microsoft C. The include file *tsil.h* is available along with the library file tsil.lib. The C compiler used here was Turbo C++ and there are some compatibility problems with Microsoft C (same problem also with Turbo C). So to properly link to tsil.lib, one must obtain a special subroutine (*search.obj*) from Intel to include in the linking.

To obtain the power and flexibility of object oriented programming we chose to use C++. To link properly a C include file to C++, the function definitions must be declared as extern "C" routines. For example:

```
StatusT initialize_etann(struct InitStuct *init_data);
```

becomes

```
extern "C" { extern StatusT initialize_etann(struct InitStuct *init_data);}
```

where *StatusT* declares a binary type variable.

The tsil routines are described in the iNNTS manual. Subroutines for such things as loading weights, reading weights, setting trainer control voltages, etc. are provided. An ETANN object was created with many of these subroutines called by member functions of the object. For example, when the object is created the constructor function asks for values of chip parameter such as V_{refi} and does other setup tasks.

To do the isolation net tests here, several other objects were developed. For example a PLUG object contains the tower energies and distributions. The PLUG towers can be sent to a NET object which contains the network description. By overloading operators one can do things like

```
PLUG >= NET
```

to send the PLUG information to the NET object. Similarly, the NET information is sent to the ETANN via

```
NET >= ETANN
```

With the ETANN object we could easily read and alter a given weight. This is not available with the other iNNTS programs. In practice we would scan through the energies as discussed in the text and find the average f^{isol} at transition from high to low for a give template. If this was not at the target value, we would read a bias weight and alter it by some amount. Then we would redo the scan and see if the average f^{isol} had gotten closer to the desired value. This tweaking of the bias and scanning of the transitions continued until the target value was reached. This process was all done interactively within one program.

The +5.00v supply voltage (Vcc) to the chip from the trainer can vary by as much as 50-60mV from 5.00v. This is not a problem as long as the chip remains in the trainer. However, if the chip is to be taken from the trainer and used in an embedded system, it is very important that Vcc be the same. The chip performance can change dramatically with even a 50mV change in Vcc. Since the isolation chip is to be used in a fastbus board with exactly 5.00v input, it was necessary to make the trainer voltage also exactly 5.00v. Information on an undocumented iNNTS subroutine was obtained from Intel and this was made into an ETANN member function. With wires attached to a Vcc pin and a ground pin while the chip was connected to the trainer, we could use this routine to adjust Vcc till it was 5.00v.

The ETANN and other C++ objects mentioned here, although sufficient for the isolation analysis, are still at a early stage of development. We hope to develop the routines into an ETANN class library that would be useful for other ETANN projects (e.g. back-propagation training). The lowest level objects also should be easily adaptable to other neural network chips. The OOP structure of C++ allows one to easily build on previous work by using the inheritance feature to incorporate old objects into new ones. For further information on these routines contact C. S. Lindsey.

References

1. B. Denby, M. Franklin, S.H. Kim, J. Konigsberg, M. Timko, CDF Note 1538, *Proposal for a Level-2 Isolated Plug Electron Trigger for the 1991/2 Collider Run.*
2. D. Y. Wu and M. K. Campbell, *Nucl. Inst. & Meth.* A317 (1992) 323-334.
3. H. Haggerty, *Discrete Component Hardware Neural Net for Drift Chamber Track Finding*, submitted to NSS Conference Record, 1992 IEEE Nuclear Science Symposium.
4. W. Badgett, M. Campbell, D. Wu, *Neural-Net Based Calorimeter Trigger*, preprint Michigan Trigger Group.
5. B. Denby, M. K. Campbell, F. Bedeschi, N. Chriss, C. Bowers, F. Nesti, *Neural Networks for Triggering*, IEEE Transactions on Nuclear Science, Vol. 27, No. 2, April, 1990.
6. D. Y. Wu, M. K. Campbell, W. Badgett, *A Pattern Recognition Level-2 B Trigger at CDF in 1991*, CDF/DOC/TRIGGER/PUBLIC/1310.
7. M. Holler et al.,*Proc. Int. Joint Conf. on Neural Networks*, Washington, D.C., (1989), vol. II, IEEE Catalog 89CH2756, p. 191.
8. Data booklet for Intel 80170NX *Electrically Trainable Analog Neural Network*, Intel Corp., June. 1991.
9. Intel Corp., 2250 Mission College Boulevard, MS SC9-40, Santa Clara, Ca. 95052-8125
10. *iDynamind User's Guide*, NeuroDynamiX, Boulder Colorado, 1991.
11. *Brainmaker Professional User's Guide*, California Scientific Software, 10141 Evening Star Dr. #6, Grass Valley, Ca. 95945.

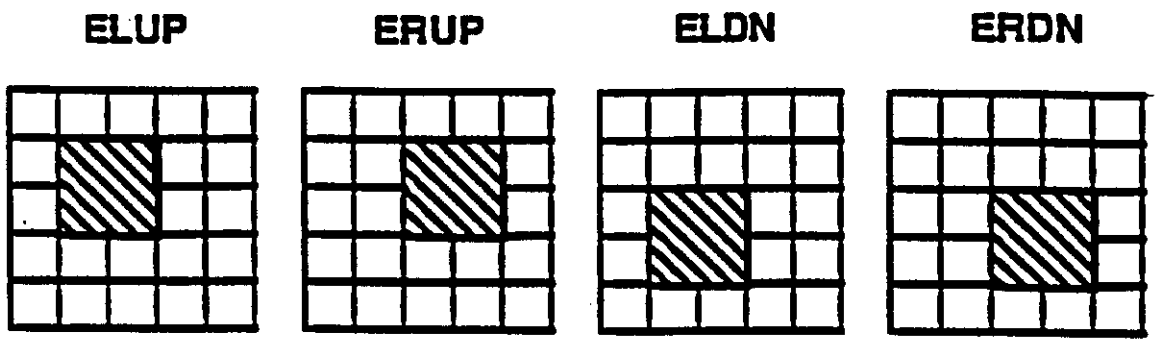

Figure 1: Isolation net templates [1].

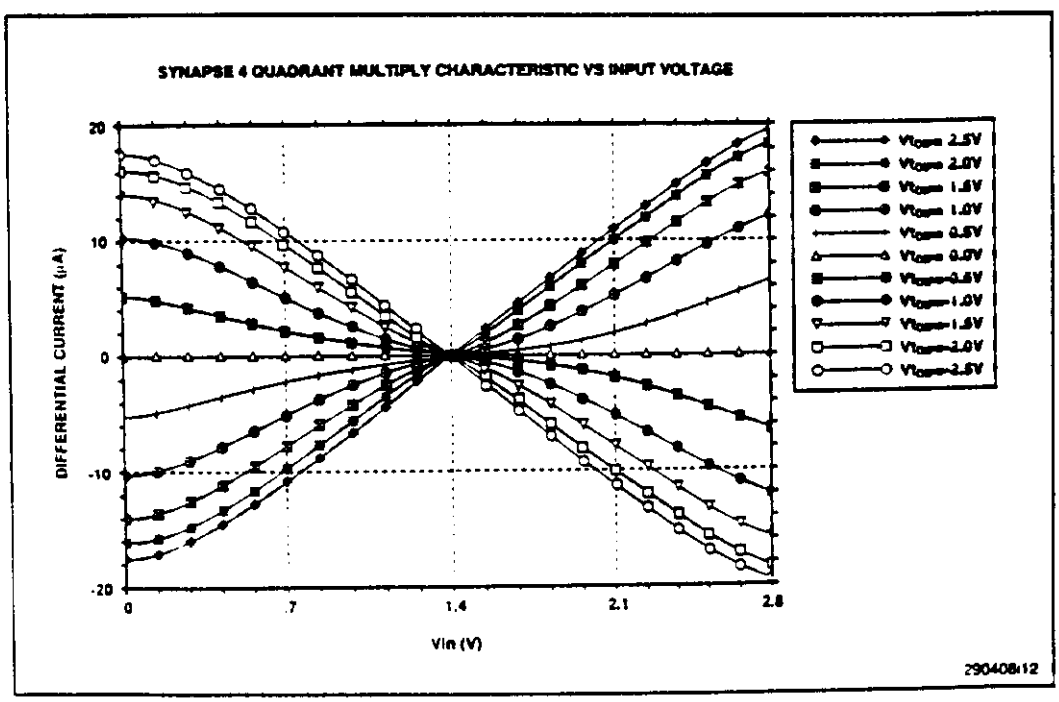

Figure 2: Synapse output current versus input voltage for various weights Vt_{dif} [8].

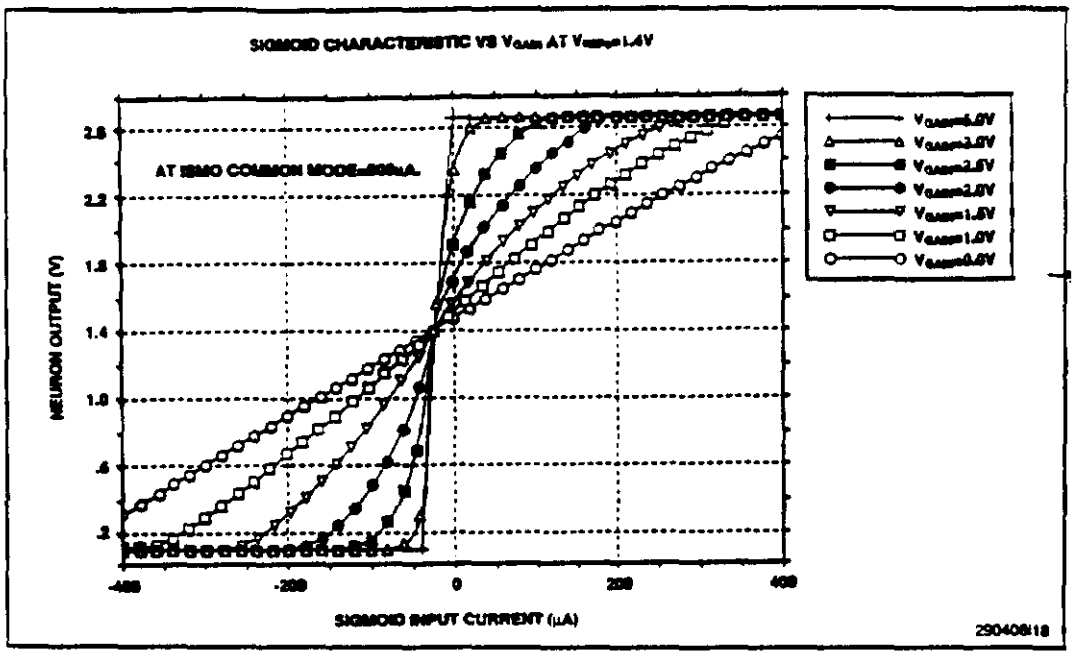

Figure 3: Neuron output distributions versus total input current for various values of V_{refi} [8].

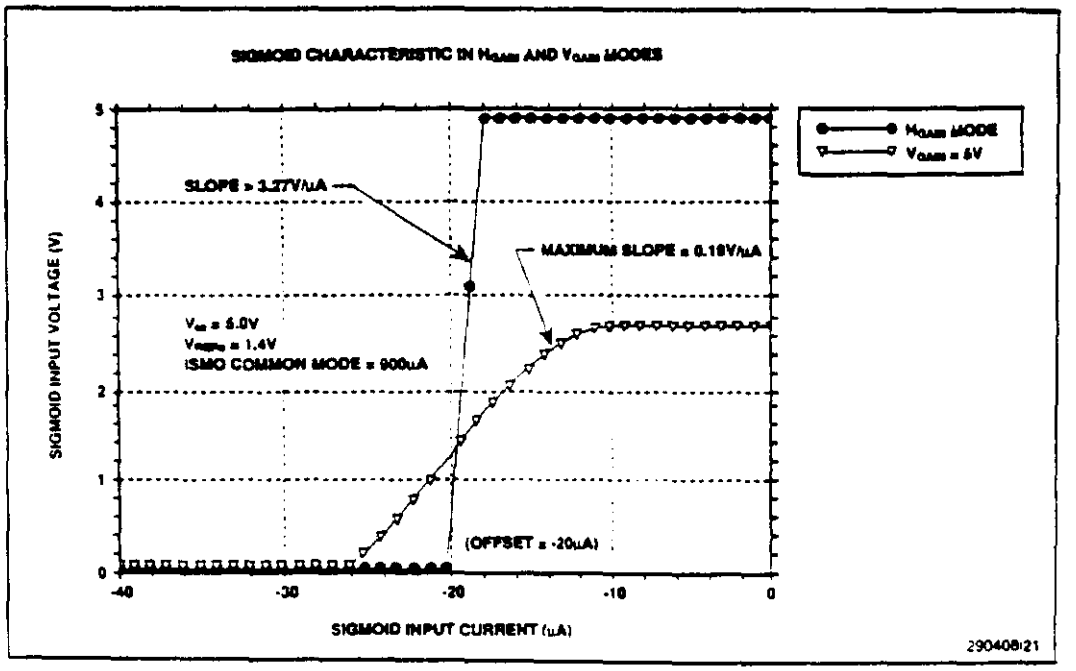

Figure 4: Neuron output versus input current for V_{gain} mode with $V_{gain} = 5.0v$ and for H_{gain} mode [8].

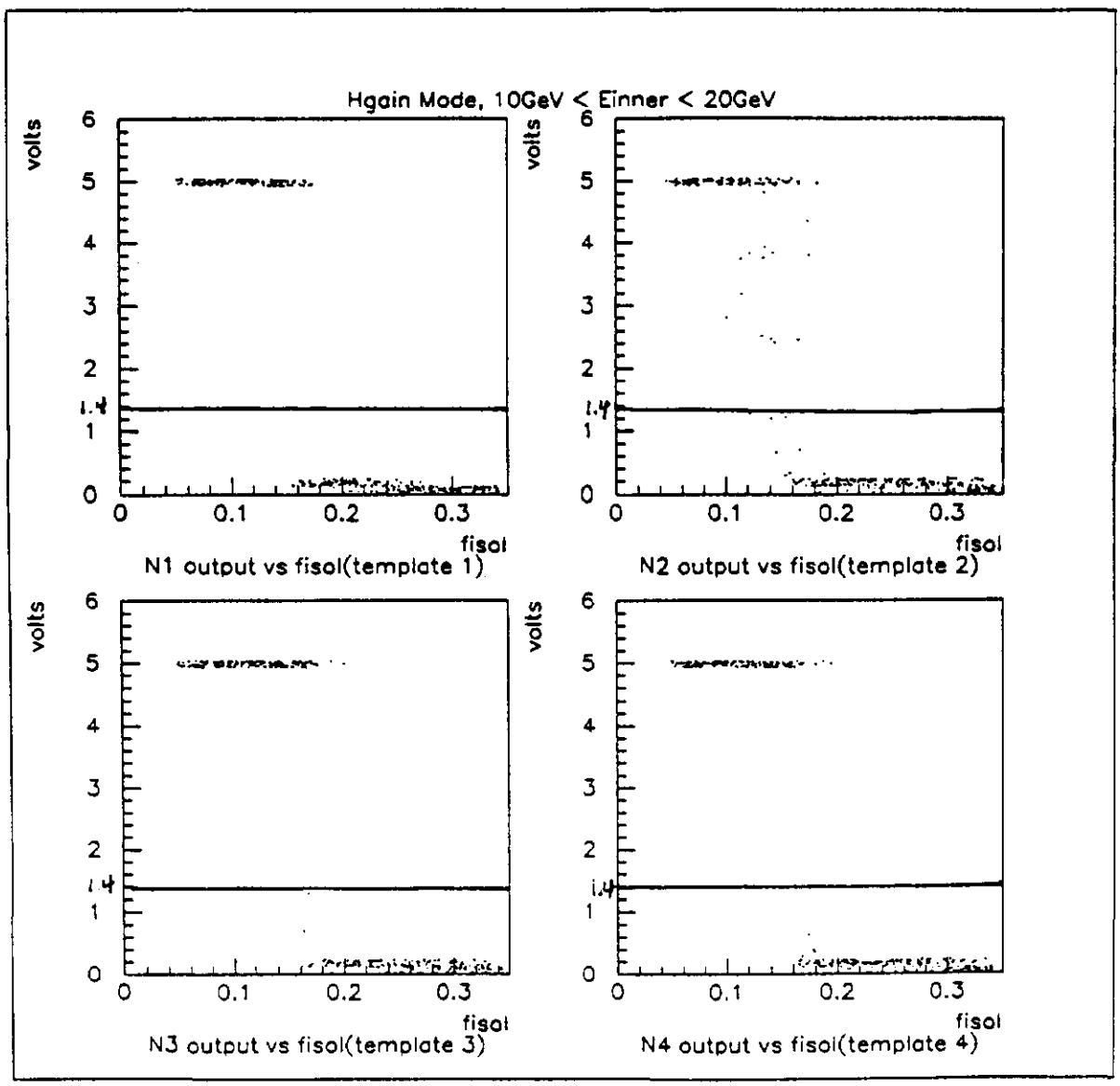

Figure 5: For the chip in H_{gain} mode, neuron voltage output versus f^{fisol} for $10GeV < E_{inner} < 20GeV$. (a)-(d) templates 1-4. Line shows threshold cut at 1.4v.

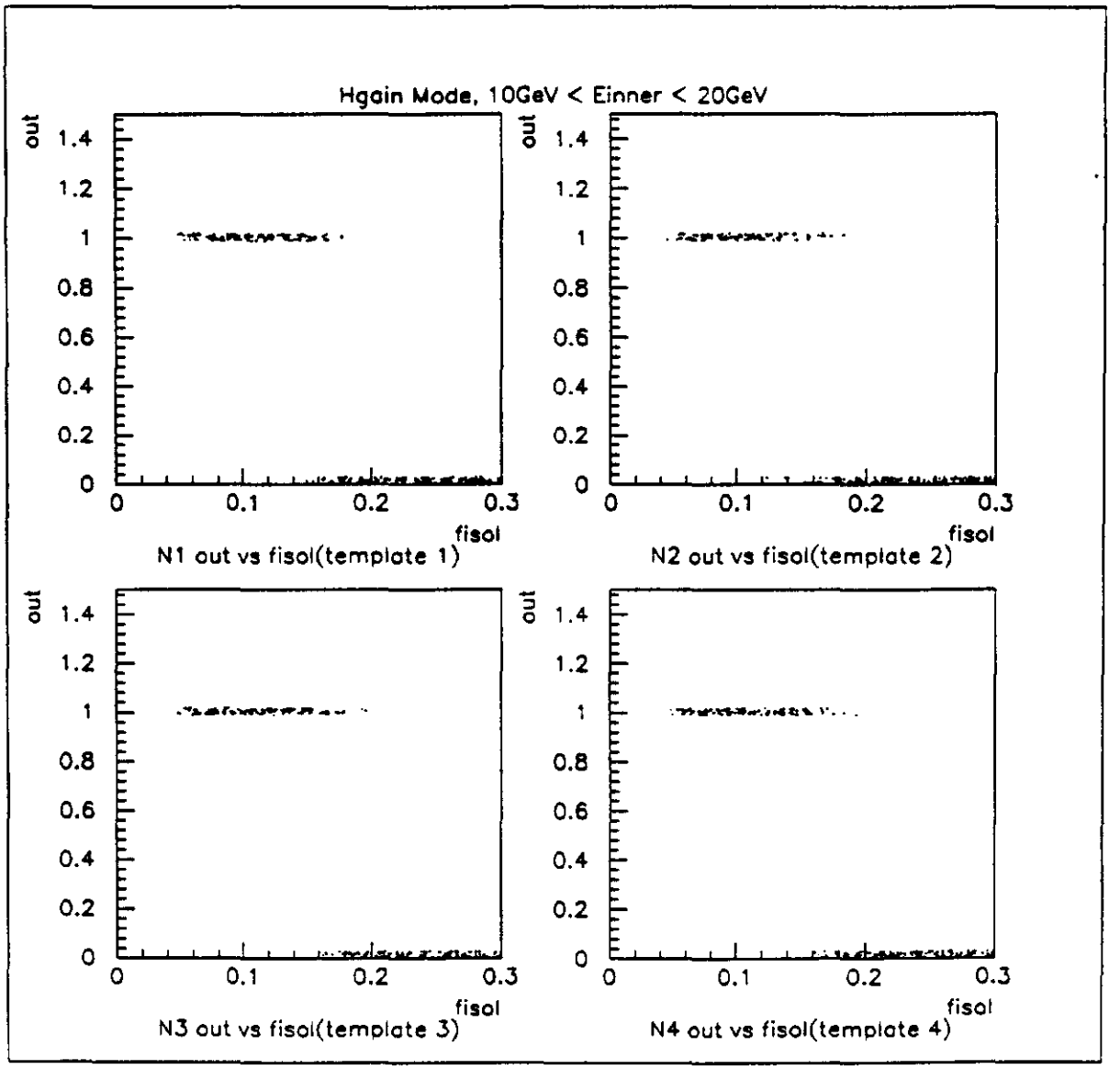

Figure 6: For the chip in H_{gain} mode, neuron output (0 or 1 using threshold cut of 1.4v as in fig. 5) versus f^{isol} for $10GeV < E_{inner} < 20GeV$. (a)-(d) templates 1-4.

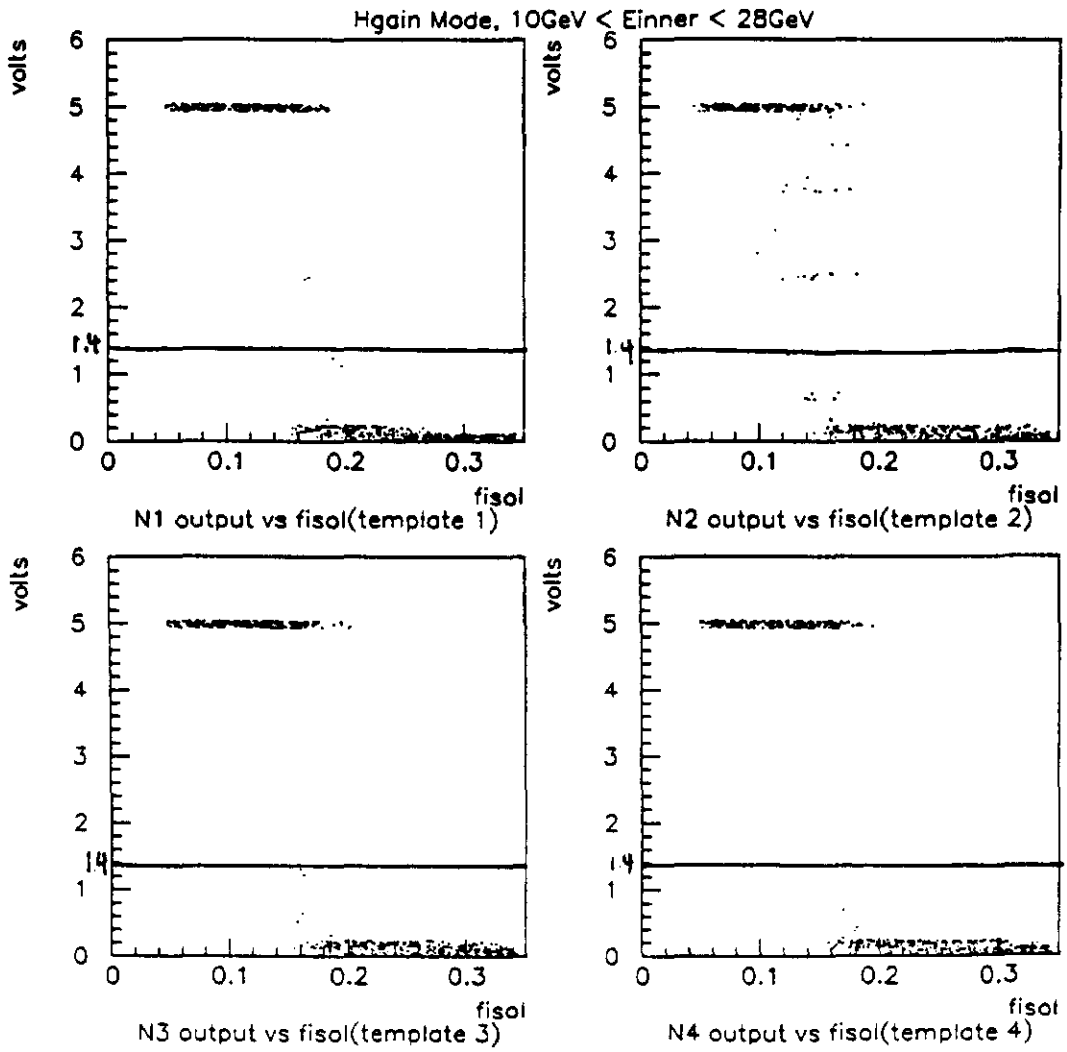

Figure 7: For the chip in H_{gain} mode, neuron voltage output versus f^{isol} for $10GeV < E_{inner} < 28GeV$. (a)-(d) templates 1-4. Line shows threshold cut at 1.4v.

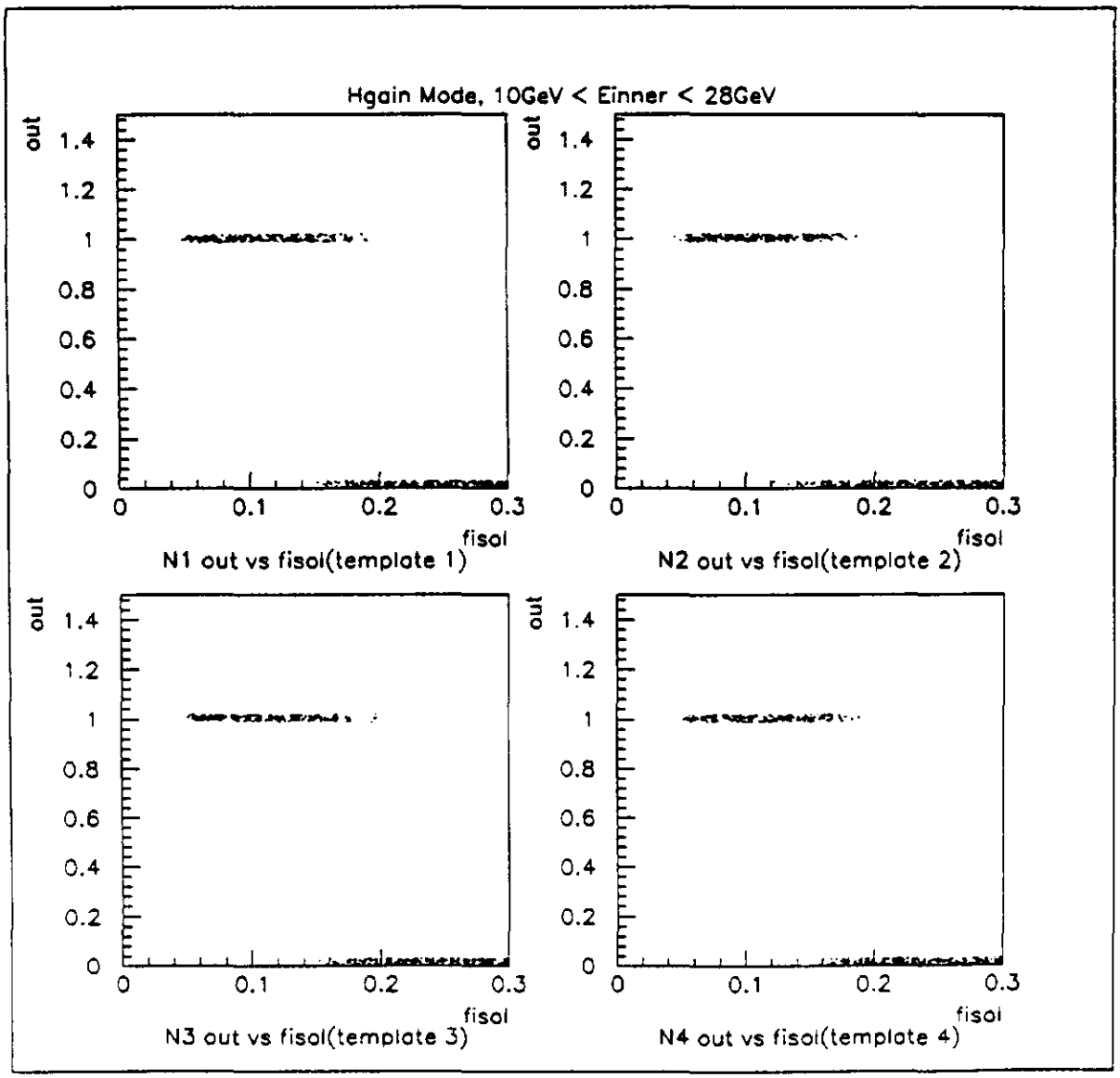

Figure 8: For the chip in H_{gain} mode, neuron output (0 or 1 using threshold cut of 1.4v as in fig. 7) versus f^{isol} for $10GeV < E_{inner} < 28GeV$. (a)-(d) templates 1-4.

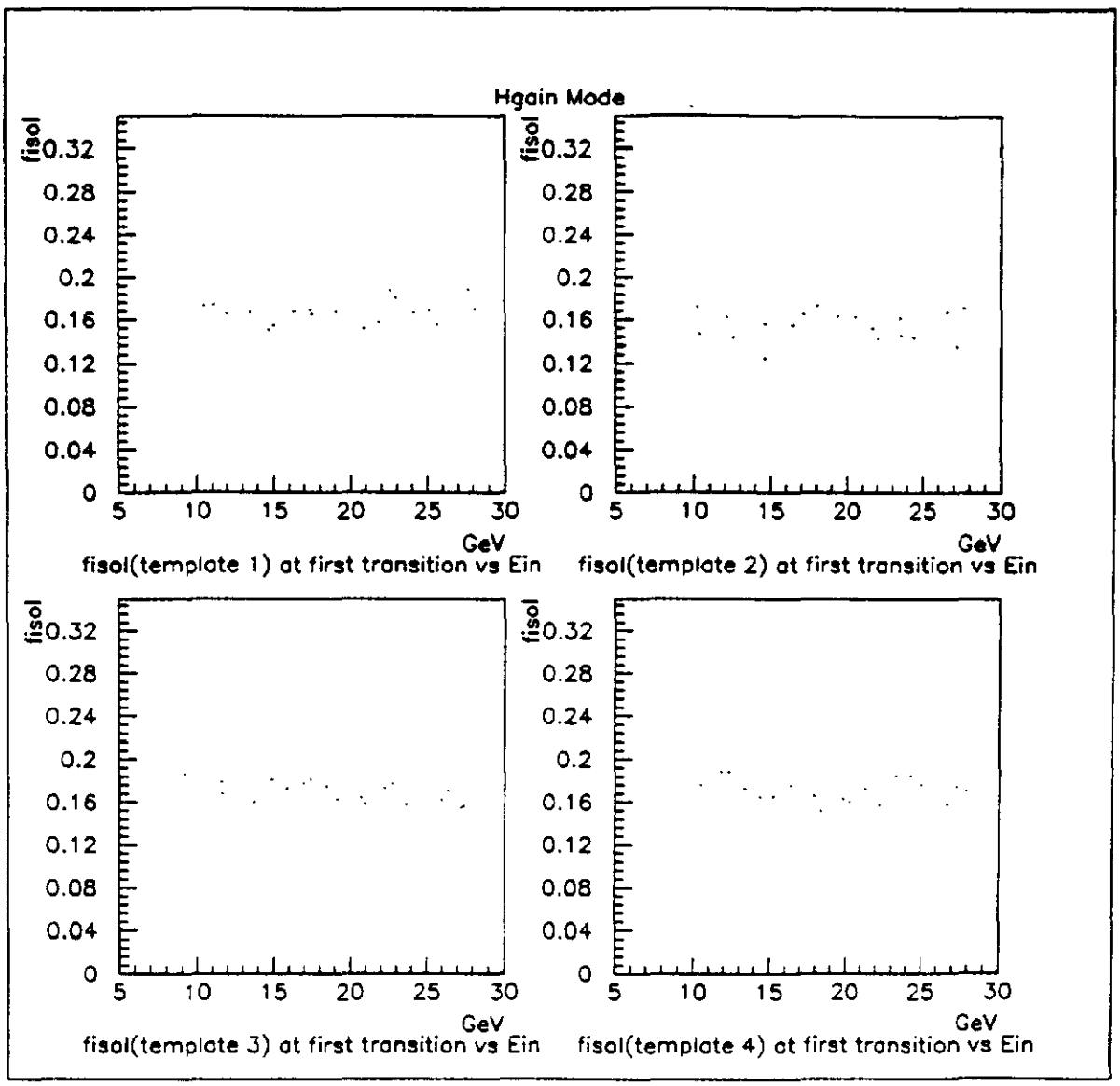

Figure 9: For the chip in H_{gain} mode, f^{isol} at first transition high to low verus E_{inner}. (a)-(d) templates 1-4.

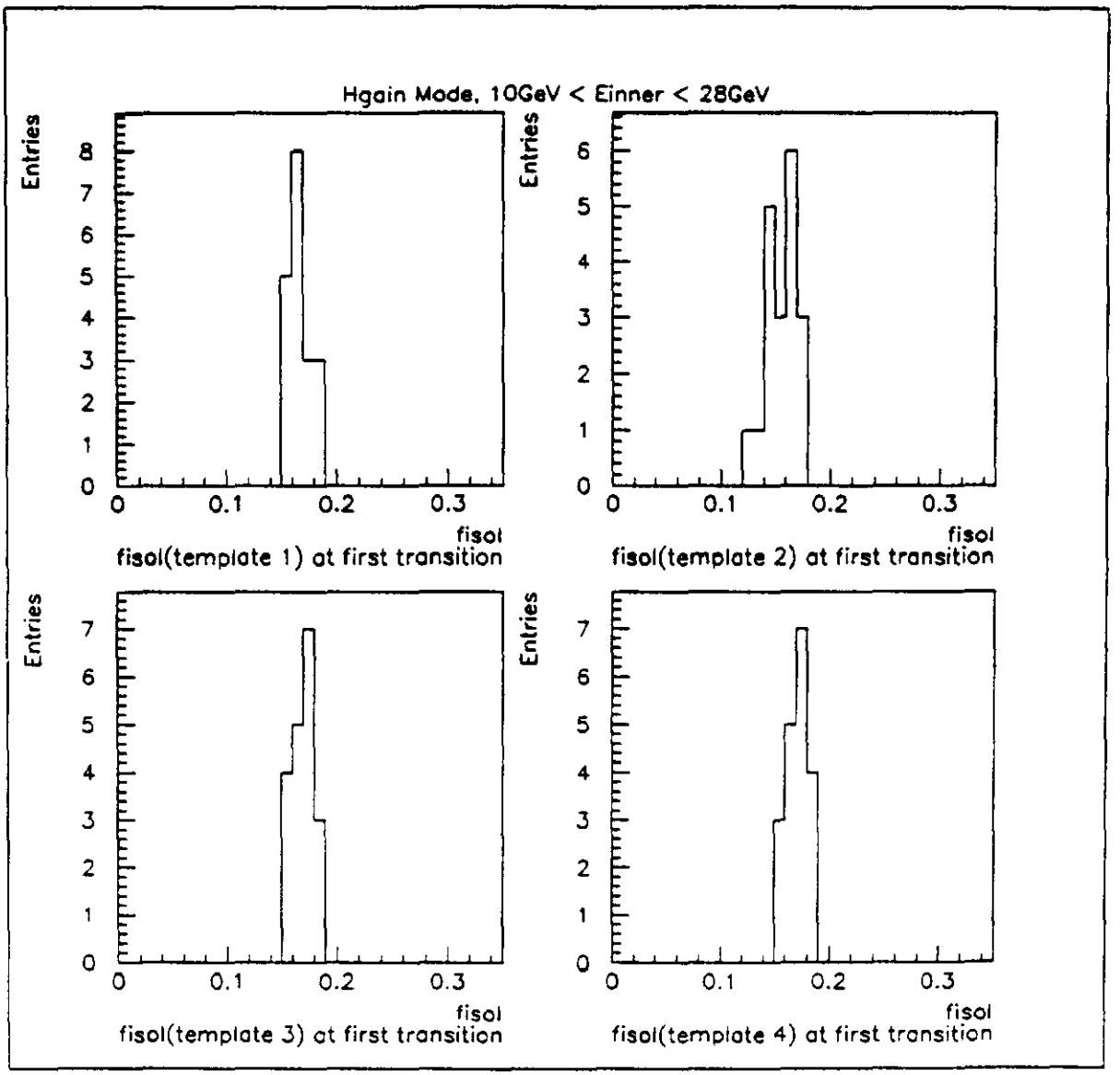

Figure 10: For the chip in H_{gain} mode, f^{isol} at first transition high to low for $10GeV < E_{inner} < 28GeV$. (a)-(d) templates 1-4.

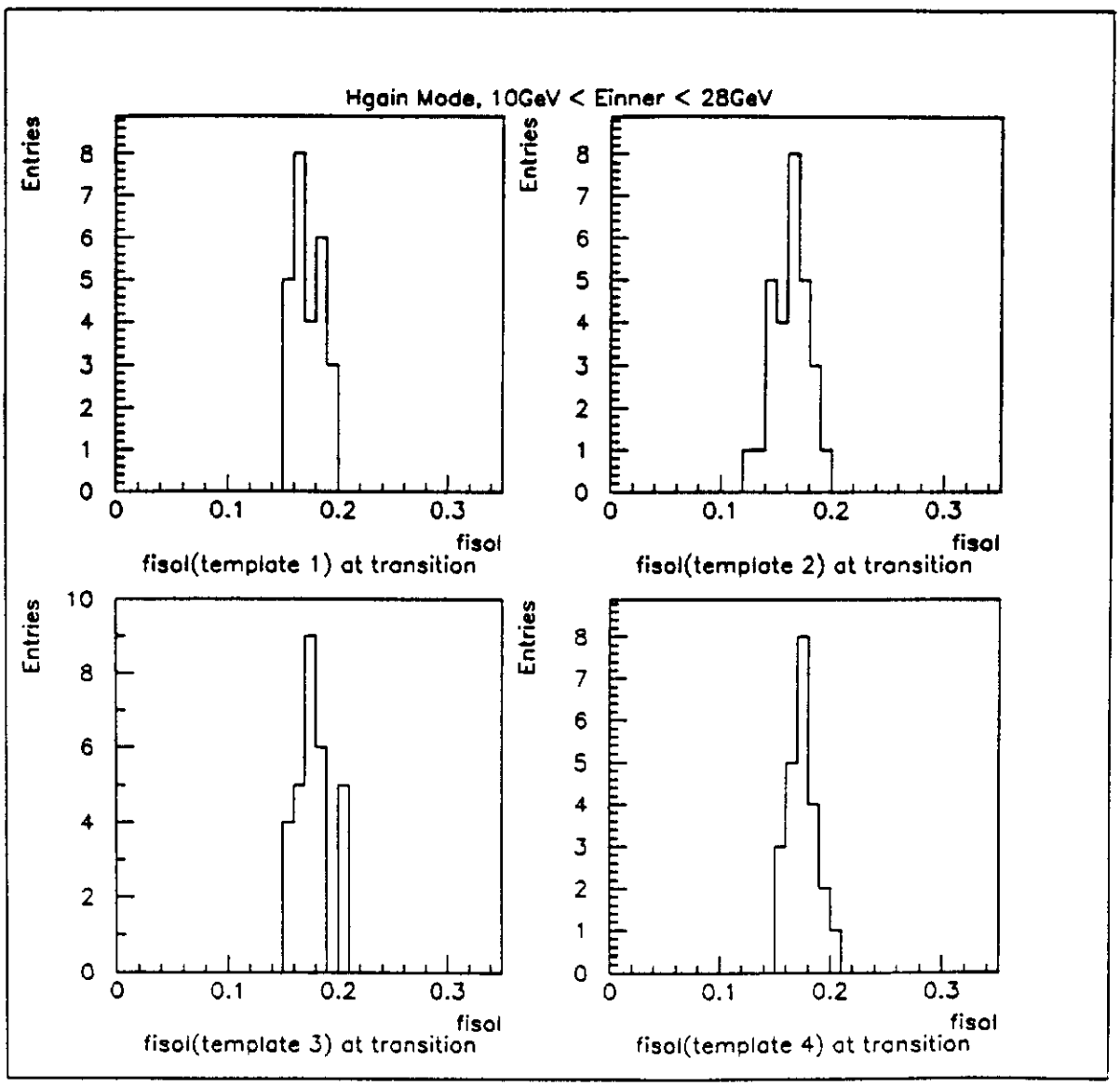

Figure 11: For the chip in H_{gain} mode, f^{isol} at all transitions high to low for $10GeV < E_{inner} < 28GeV$. (a)-(d) templates 1-4.

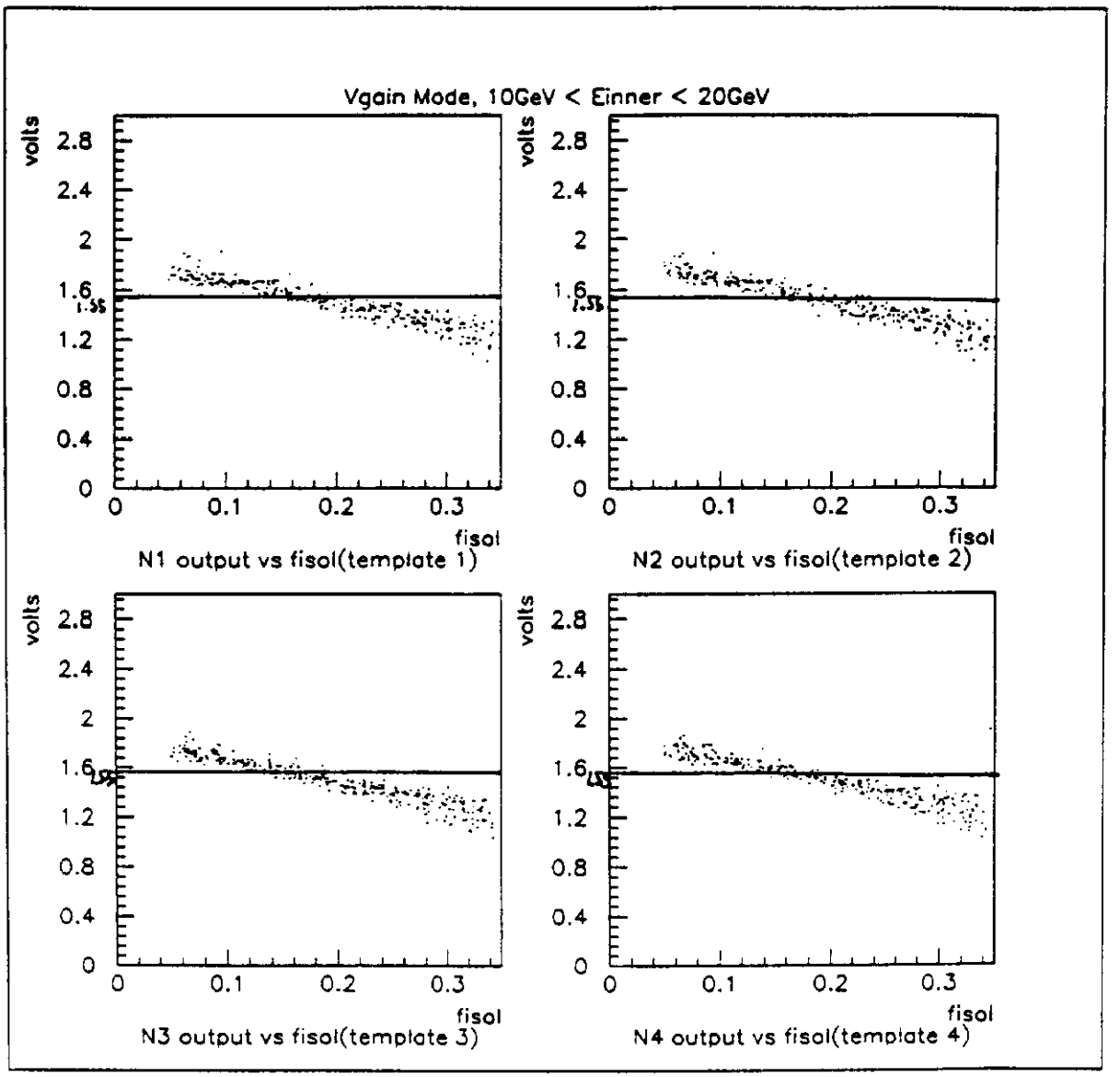

Figure 12: For the chip in V_{gain} mode, neuron voltage output versus f^{isol} for $10GeV < E_{inner} < 20GeV$. (a)-(d) templates 1-4. Line shows threshold cut at 1.55v.

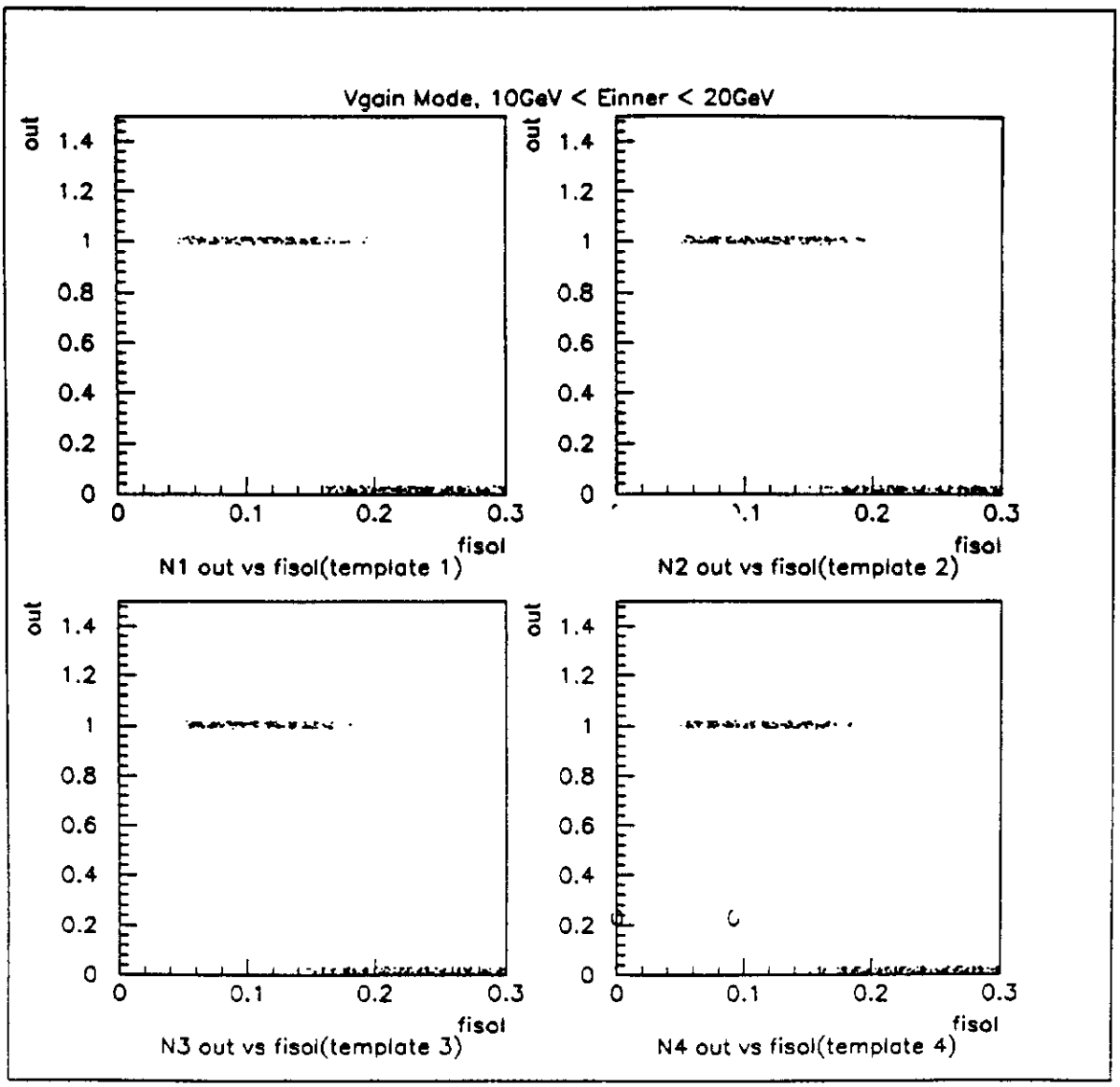

Figure 13: For the chip in V_{gain} mode, neuron output (0 or 1 using threshold cut of 1.55v as in fig. 12) versus f^{isol} for $10GeV < E_{inner} < 20GeV$. (a)-(d) templates 1-4.

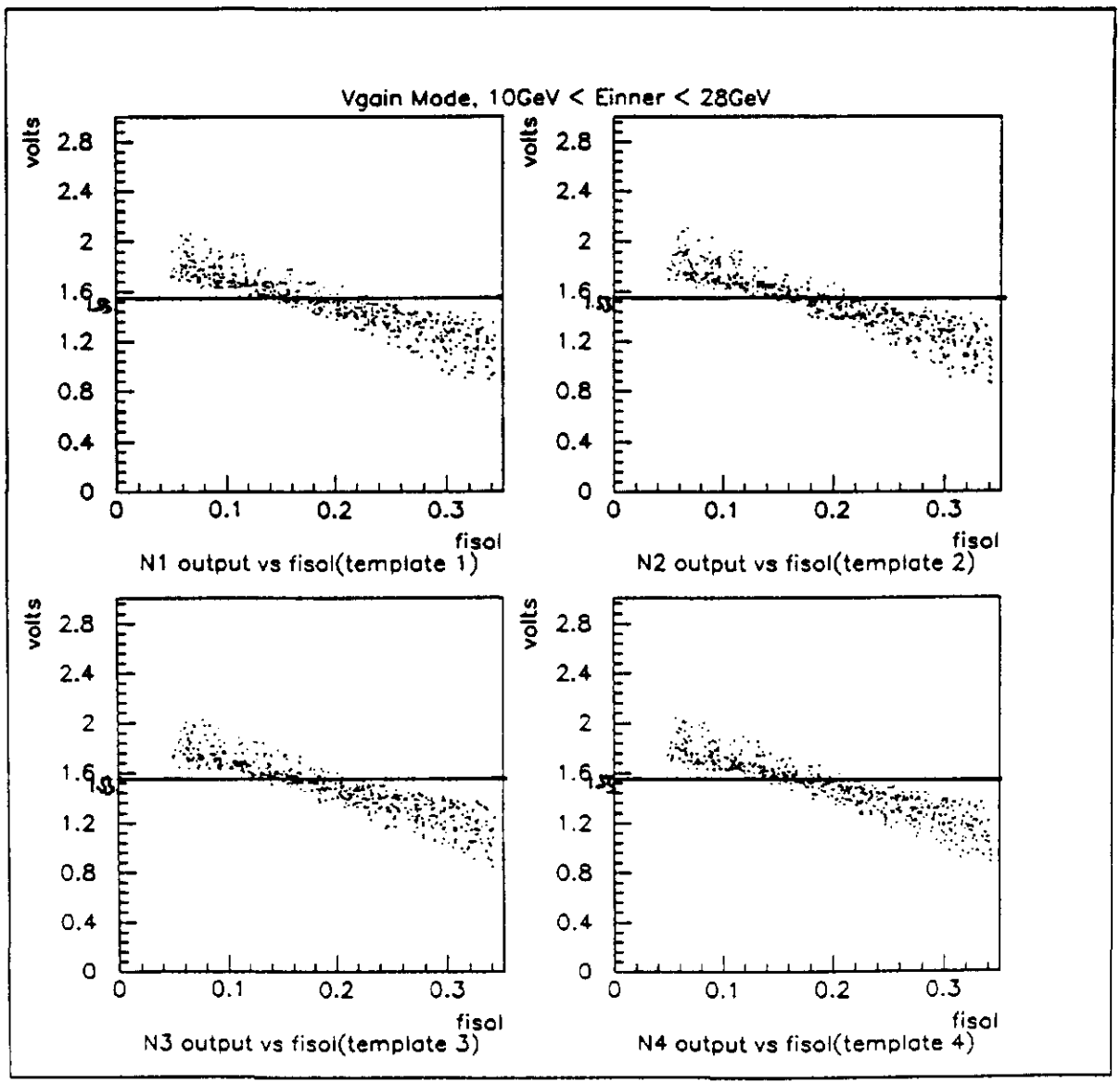

Figure 14: For the chip in V_{gain} mode, neuron voltage output versus f^{isol} for $10GeV < E_{inner} < 28GeV$. (a)-(d) templates 1-4. Line shows threshold cut at 1.55v.

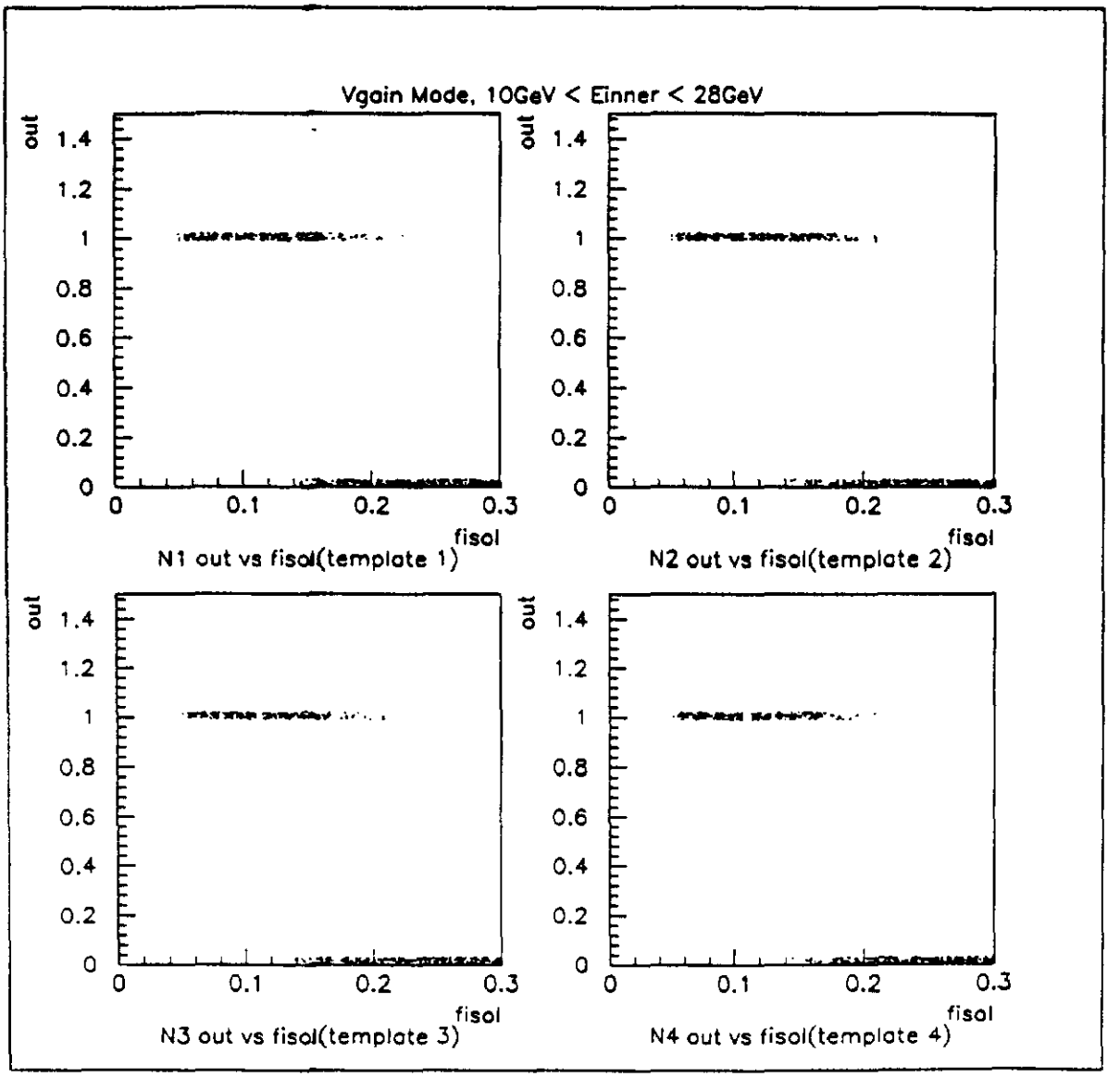

Figure 15: For the chip in V_{gain} mode, neuron output (0 or 1 using threshold cut of 1.55v as in fig. 14) versus f^{isol} for $10GeV < E_{inner} < 28GeV$. (a)-(d) templates 1-4.

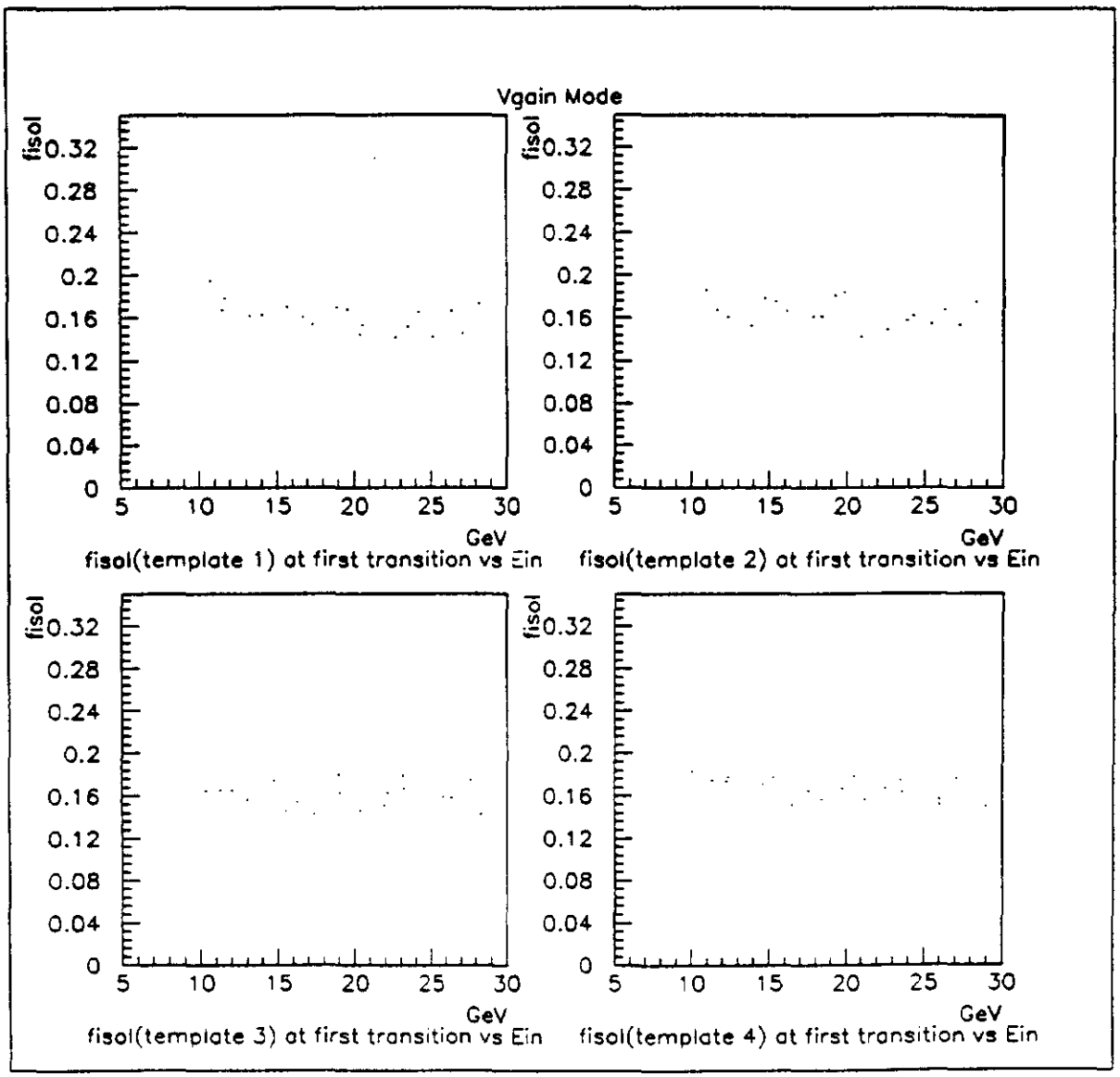

Figure 16: For the chip in V_{gain} mode, f^{fisol} at first transition high to low verus E_{inner}. (a)-(d) templates 1-4.

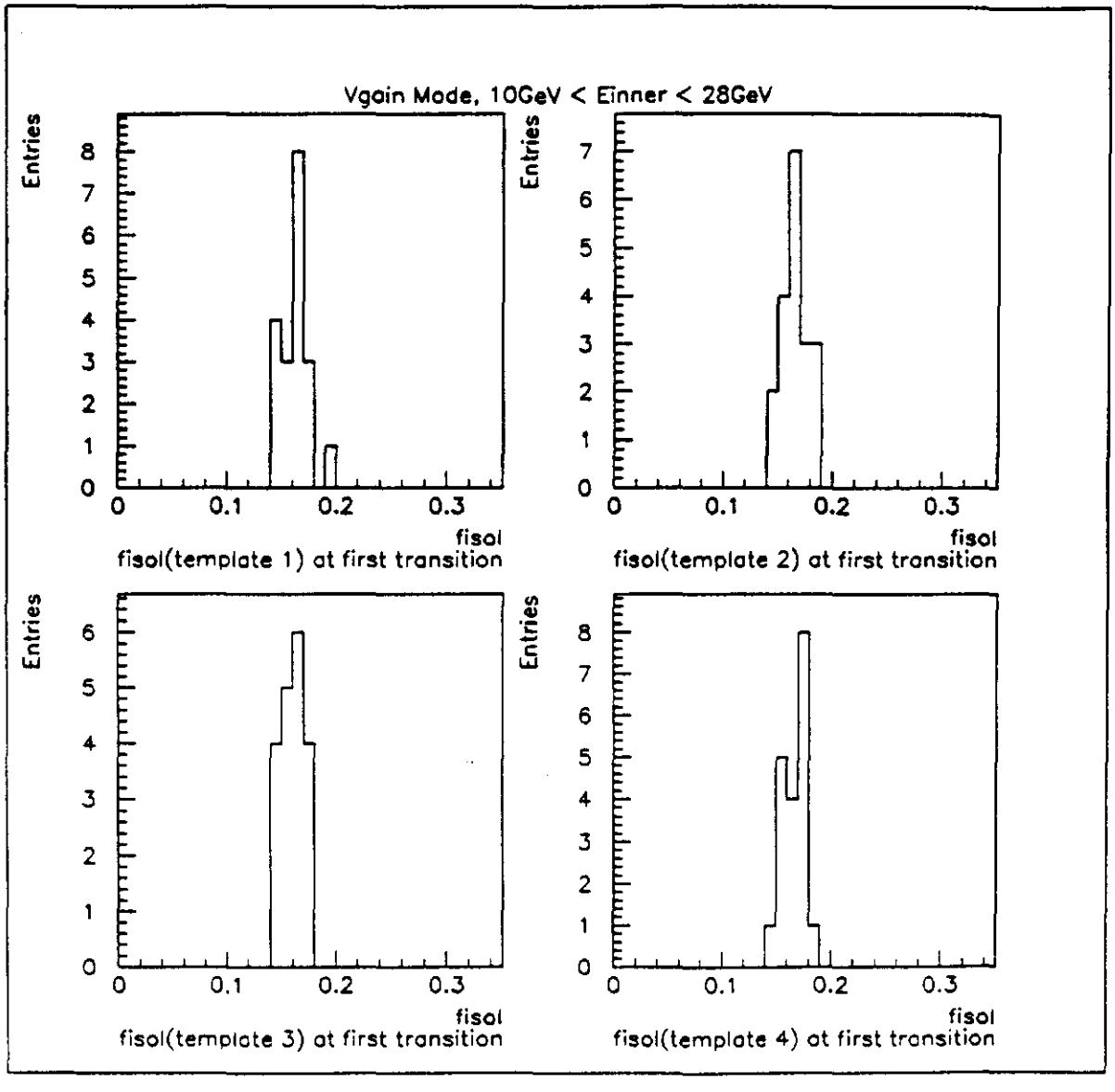

Figure 17: For the chip in V_{gain} mode, f^{isol} at first transition high to low for $10GeV < E_{inner} < 28GeV$. (a)-(d) templates 1-4.

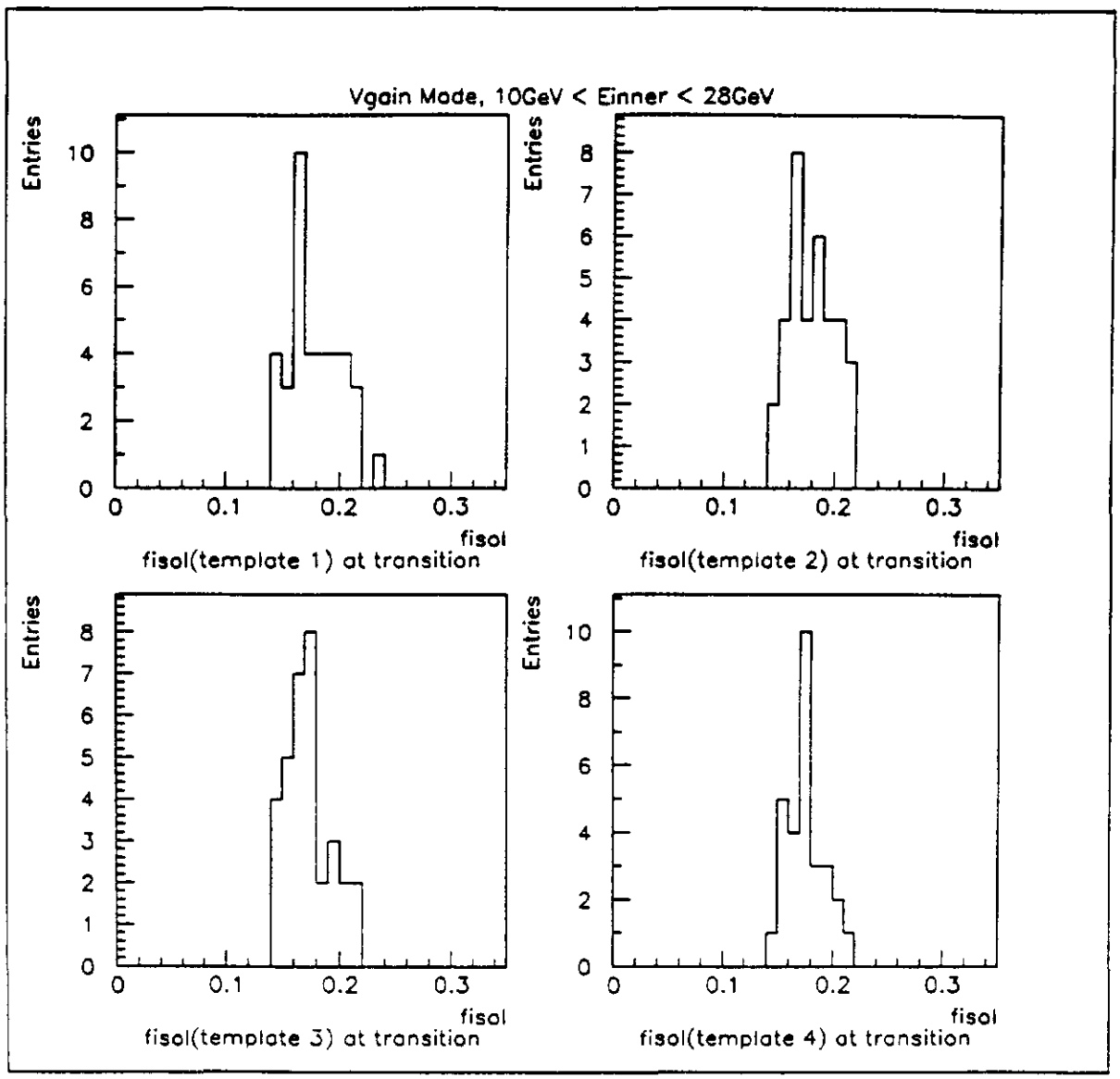

Figure 18: For the chip in V_{gain} mode, f^{isol} at all transitions high to low for $10GeV < E_{inner} < 28GeV$. (a)-(d) templates 1-4.